

IC Compiler Commands

Version C-2009.06-SP4, December 2009

SYNOPSYS®

Copyright Notice and Proprietary Information

Copyright © 2009 Synopsys, Inc. All rights reserved. This software and documentation contain confidential and proprietary information that is the property of Synopsys, Inc. The software and documentation are furnished under a license agreement and may be used or copied only in accordance with the terms of the license agreement. No part of the software and documentation may be reproduced, transmitted, or translated, in any form or by any means, electronic, mechanical, manual, optical, or otherwise, without prior written permission of Synopsys, Inc., or as expressly provided by the license agreement.

Right to Copy Documentation

The license agreement with Synopsys permits licensee to make copies of the documentation for its internal use only. Each copy shall include all copyrights, trademarks, service marks, and proprietary rights notices, if any. Licensee must assign sequential numbers to all copies. These copies shall contain the following legend on the cover page:

"This document is duplicated with the permission of Synopsys, Inc., for the exclusive use of _____ and its employees. This is copy number _____."

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Registered Trademarks (®)

Synopsys, AMPS, Astro, Behavior Extracting Synthesis Technology, Cadabra, CATS, Certify, CHIPit, Design Compiler, DesignWare, Formality, HDL Analyst, HSIM, HSPICE, Identify, Leda, MAST, ModelTools, NanoSim, OpenVera, PathMill, Physical Compiler, PrimeTime, SCOPE, Simply Better Results, SiVL, SNUG, SolvNet, Syndicated, Synplicity, Synplify, Synplify Pro, Synthesis Constraints Optimization Environment, TetraMAX, the Synplicity logo, UMRBus, VCS, Vera, and YIELDirector are registered trademarks of Synopsys, Inc.

Trademarks (™)

AFGen, Apollo, Astro-Rail, Astro-Xtalk, Aurora, AvanWaves, BEST, Columbia, Columbia-CE, Confirma, Cosmos, CosmosLE, CosmosScope, CRITIC, CustomExplorer, CustomSim, DC Expert, DC Professional, DC Ultra, Design Analyzer, Design Vision, DesignerHDL, DesignPower, DFTMAX, Direct Silicon Access, Discovery, Eclipse, Encore, EPIC, Galaxy, Galaxy Custom Designer, HANEX, HAPS, HapsTrak, HDL Compiler, Hercules, Hierarchical Optimization Technology, High-performance ASIC Prototyping System, HSIM^{plus}, i-Virtual Stepper, IICE, in-Sync, iN-Tandem, Jupiter, Jupiter-DP, JupiterXT, JupiterXT-ASIC, Liberty, Libra-Passport, Library Compiler, Magellan, Mars, Mars-Rail, Mars-Xtalk, Milkyway, ModelSource, Module Compiler, MultiPoint, Physical Analyst, Planet, Planet-PL, Polaris, Power Compiler, Raphael, Saturn, Scirocco, Scirocco-i, Star-RCXT, Star-SimXT, StarRC, System Compiler, System Designer, Taurus, TotalRecall, TSUPREM-4, VCS Express, VCSi, VHDL Compiler, VirSim, and VMC are trademarks of Synopsys, Inc.

Service Marks (SM)

MAP-in, SVP Café, and TAP-in are service marks of Synopsys, Inc.

SystemC is a trademark of the Open SystemC Initiative and is used under license.

ARM and AMBA are registered trademarks of ARM Limited.

Saber is a registered trademark of SabreMark Limited Partnership and is used under license.

All other product or company names may be trademarks of their respective owners.

Table of Contents

add_clock_drivers.	1
add_distributed_hosts	11
add_drc_error_detail	14
add_end_cap	16
add_open_drc_error_detail	18
add_pg_pin_to_db	20
add_pg_pin_to_lib	22
add_port_state	24
add_pst_state	26
add_row	28
add_tap_cell_array	30
add_to_collection	36
add_to_rp_group	39
adjust_fp_floorplan	43
adjust_fp_io_placement	48
adjust_premesh_connection.	49
alias	51
align_fp_pins	53
align_objects.	57
all_active_scenarios	60
all_ao_cellsfp	61
all_bounds_of_cell	62
all_cells_in_bound	64
all_clocks	66

all_connected	67
all_connectivity_fanin	69
all_connectivity_fanout	71
all_critical_cells	73
all_critical_pins	75
all_designs	77
all_dont_touch	78
all_drc_violated_nets	80
all_fanin	82
all_fanout	84
all_fixed_placement	86
all_high_fanout	88
all_ideal_nets	90
all_inputs	92
all_isolation_cellsfp	94
all_level_shiftersfp	95
all_macro_cells	96
all_mtcmos_cellsfp	98
all_objects_in_bounding_box	99
all_outputs	101
all_physical_only_cells	103
all_physical_only_nets	105
all_physical_only_ports	107
all_registers	109
all_rp_groups	112
all_rp_hierarchicals	114
all_rp_inclusions	116
all_rp_instantiations	118
all_rp_references	120
all_scenarios	122
all_size_only_cells	123
all_spare_cells	125
all_threestate	127
all_tieoff_cells	129
allocate_fp_budgets	131
analyze_fp_rail	134
analyze_fp_routing	138
analyze_rail	142

analyze_subcircuit	146
append_to_collection	157
apropos.	159
archive_design	161
assign_flip_chip_nets	163
balance_inter_clock_delay	167
calculate_caa_based_yield2db	169
calculate_hier_antenna_property	171
cd	175
change_fp_soft_macro_to_black_box	177
change_link	178
change_macro_view.	180
change_names.	182
change_selection	188
change_tie_connection.	191
characterize	193
check_ccs_lib	198
check_clock_tree	201
check_database	203
check_design	205
check_error.	208
check_fp_budget_result	210
check_fp_pin_alignment.	212
check_fp_pin_assignment	215
check_fp_rail	219
check_fp_timing_environment	221
check_isolation_cellsfp.	225
check_legality.	227
check_level_shifters.	228
check_library	230
check_license.	236
check_mpc	237
check_mv_design.	240
check_noise	245
check_physical_constraints	247
check_physical_design.	249
check_route	254
check_routeability.	255

check_rp_groups	256
check_scan_chain	258
check_scenarios.	260
check_target_library_subset.	261
check_timing.	262
check_tlu_plus_files	267
clock_opt	269
close_distributed_route	272
close_mw_cel.	273
close_mw_lib	276
Collections_and_Querying	277
commit_fp_group_block_ring	282
commit_fp_plan_groups	283
commit_fp_rail	285
commit_skew_group.	286
compare_collections.	288
compare_delay_calculation	290
compare_interface_timing	292
compare_lib	296
compare_rc.	298
compile_clock_tree.	302
compile_fp_clock_plan	305
compile_premesh_tree.	307
compute_polygons	309
connect_net	314
connect_pin	316
connect_power_domain	318
connect_power_net_info	320
connect_power_switch.	322
connect_spare_diode	324
connect_supply_net	326
connect_tie_cells	328
convert_from_polygon	331
convert_mw_lib.	333
convert_to_polygon	335
convert_wire_ends	336
convert_wire_to_pin	337
copy_collection.	339

copy_floorplan	341
copy_mim	343
copy_mw_cel	345
copy_mw_lib	348
copy_objects	349
count_drc_violations	351
cputime	353
create_auto_shield	355
create_base_array	358
create_boundary	360
create_bounds	363
create_buffer_tree	366
create_cell	368
create_clock	371
create_clock_mesh	374
create_command_group	381
create_connview	382
create_custom_wire	385
create_differential_group	389
create_drc_error	390
create_drc_error_type	393
create_edit_group	396
create_fp_block_shielding	397
create_fp_blockages_for_child_hardmacro	401
create_fp_group_block_ring	402
create_fp_pins	405
create_fp_placement	407
create_fp_plan_group_padding	410
create_fp_virtual_pad	412
create_generated_clock	413
create_ilm	417
create_ilm_models	425
create_lib_track	429
create_macro_fram	431
create_mw_cel	439
create_mw_lib	441
create_net	443
create_net_shape	445

create_open_drc_error	449
create_open_locator_drc_error	452
create_operating_conditions	455
create_pad_rings	458
create_partition	460
create_pg_network	462
create_physical_bus	465
create_physical_buses_from_patterns	467
create_pin_guide	469
create_placement	472
create_placement_blockage	474
create_plan_groups	477
create_port	480
create_power_domain	482
create_power_net_info	485
create_power_straps	487
create_power_switch	496
create_power_switch_array	499
create_power_switch_ring	504
create_preroute_vias	508
create_pst	514
create_qor_snapshot	516
create_qtm_clock	518
create_qtm_constraint_arc	520
create_qtm_delay_arc	523
create_qtm_drive_type	526
create_qtm_generated_clock	529
create_qtm_insertion_delay	531
create_qtm_load_type	533
create_qtm_model	535
create_qtm_path_type	537
create_qtm_port	540
create_rail_setup	542
create_rectangular_rings	545
create_rectilinear_rings	551
create_route_guide	555
create_routing_blockage	557
create_rp_group	559

create_scenario	564
create_short_drc_error	565
create_site_row	567
create_spacing_drc_error	569
create_stack_via_on_pad_pin	572
create_supply_net	575
create_supply_port	577
create_terminal	579
create_text	581
create_track	584
create_user_shape	586
create_via	589
create_voltage_area	592
create_zrt_shield	596
current_design	601
current_design_name	603
current_instance	604
current_mw_cel	608
current_mw_lib	610
current_scenario	611
cut_objects	612
cut_row	615
date	616
define_antenna_accumulation_mode	617
define_antenna_layer_ratio_scale	618
define_antenna_layer_rule	620
define_antenna_rule	622
define_bus	625
define_io_antenna_area	627
define_io_diode_protection	629
define_io_gate_size	631
define_name_rules	633
define_proc_attributes	647
define_routing_rule	651
define_scaling_lib_group	655
define_user_attribute	657
define_via	660
define_zrt_redundant_vias	661

delete_operating_conditions	663
derive_constraints	664
derive_mpc_macro_options	666
derive_mpc_options	669
derive_mpc_port_options	673
derive_pg_connection	676
detect_flcc_hotspot	680
detect_lcc_hotspot	682
disconnect_net	684
disconnect_power_net_info	686
display_flip_chip_route_flylines	687
distance	689
distribute_objects	691
drive_of	695
echo	697
eco_netlist	698
end_fp_trace_mode	702
error_info	703
estimate_fp_area	704
estimate_fp_black_boxes	708
estimate_rc	711
exit	713
expand_flip_chip_cell_locations	714
expand_objects	716
explore_power_switch	719
extract_blockage_pin_via	724
extract_flcc_hotspot	730
extract_fp_rail_to_constraints	732
extract_fp_relative_location	735
extract_hier_antenna_property	737
extract_rc	739
extract_rp_group	741
extract_zrt_hier_antenna_property	744
filter_collection	746
fix_flcc_hotspot	749
fix_isolated_via	750
fix_lcc_hotspot	754
flatten_clock_gating	756

flatten_fp_black_boxes	758
flatten_fp_hierarchy	759
flip_mim	763
flip_objects	765
focal_opt.	769
foreach_in_collection	773
format_lib	775
generate_qtm_model	779
get_adjusted_endpoints	781
get_alternative_lib_cells	783
get_always_on_logic	786
get_attribute	787
get_bounds.	789
get_buffers	791
get_cell_sites	794
get_cells	796
get_clocks.	802
get_command_option_values.	804
get_core_area	806
get_coupling_capacitors.	808
get_cts_scenario	809
get_design_lib_path	810
get_designs	811
get_die_area.	814
get_dominant_scenarios	816
get_drc_errors	819
get_edit_groups	821
get_error_view_property.	823
get_flat_cells.	825
get_flat_nets.	829
get_flat_pins.	833
get_floorplan_data	837
get_fp_trace_mode.	839
get_fp_wirelength.	840
get_generated_clocks	842
get_ilm_objects.	844
get_ilms	846
get_layer_attribute	848

get_layers	849
get_lib_attribute	851
get_lib_cells	853
get_lib_pins	856
get_libs	859
get_license	862
get_location	863
get_magnet_cells	865
get_message_info	867
get_mw_cels.	869
get_net_shapes	872
get_nets	876
get_new_bounds	880
get_object_fixed_edit	882
get_object_name	883
get_object_snap_type	885
get_path_groups.	887
get_physical_buses	889
get_physical_lib_cells.	892
get_physical_lib_pins	895
get_physical_libs	898
get_pin_guides	901
get_pin_shapes	903
get_pins	907
get_placement_area.	911
get_placement_blockages	912
get_plan_groups.	915
get_polygon_area.	917
get_ports.	918
get_power_domains	921
get_power_switches.	924
get_route_guides	927
get_route_mode_options	929
get_route_zrt_common_options.	930
get_route_zrt_detail_options	931
get_route_zrt_global_options	932
get_route_zrt_track_options.	933
get_routing_blockages	934

get_rp_group_keepouts	938
get_rp_groups	940
get_scan_cells_of_chain	943
get_scan_chains.	944
get_selection	945
get_si_xtalk_bumps	948
get_site_rows	949
get_supply_nets	953
get_supply_ports	956
get_terminals	959
get_text.	963
get_timing_paths	966
get_tracks.	973
get_user_grid	977
get_user_shapes	978
get_via_masters.	982
get_vias	984
get_voltage_areas	987
get_zero_interconnect_delay_mode	989
get_zrt_net_properties	990
getenv.	991
group	993
group_path	998
gui_clear_filter_errors.	1002
gui_clear_selected_errors	1003
gui_create_tk_palette_type	1004
gui_error_browser	1006
gui_filter_errors.	1007
gui_get_error_browser_option	1009
gui_get_errors	1011
gui_get_routes_between_objects.	1013
gui_inspect_violations	1015
gui_load_area_net_connection_vm.	1017
gui_load_cell_density_mm	1019
gui_load_cell_slack_vm.	1020
gui_load_clock_delay_vm	1023
gui_load_clock_tree_vm	1026
gui_load_delta_delay_vm	1028

gui_load_hierarchy_vm	1030
gui_load_illegal_cell_placement_vm	1032
gui_load_imported_path_pins_vm	1033
gui_load_net_capacitance_vm	1035
gui_load_path_slack_vm	1038
gui_load_pin_density_mm	1041
gui_load_power_density_mm	1042
gui_load_relative_placement_vm	1044
gui_load_scan_chain_vm	1046
gui_load_static_noise_vm	1048
gui_load_voltage_area_vm	1050
gui_page_errors	1051
gui_report_errors	1052
gui_set_current_errors	1053
gui_set_error_browser_option	1055
gui_set_error_fixed	1057
gui_set_selected_errors	1059
gui_show_form	1061
gui_toggle_fixed_selected_errors	1063
gui_unload_error_cel	1064
gui_violation_schematic_add_objects	1065
gui_wave_add_signal	1067
gui_write_layout_image	1069
help	1071
history	1073
hookup_power_gating_ports	1076
hookup_retention_register	1078
identify_clock_gating	1079
ignore_site_row	1082
import_designs	1084
import_fp_black_boxes	1086
index_collection	1088
infer_power_domains	1090
initialize_floorplan	1092
initialize_rectilinear_block	1096
insert_buffer	1100
insert_diode	1108
insert_isolation_cell	1112

insert_level_shifters	1115
insert_metal_filler	1117
insert_ng_filler	1122
insert_pad_filler	1125
insert_port_protection_diodes	1127
insert_redundant_vias	1129
insert_spare_cells	1132
insert_stdcell_filler	1135
insert_tap_cells_by_rules	1140
insert_well_filler	1145
insert_zrt_redundant_vias	1148
is_false	1150
is_true	1152
is_zrt_routed_design	1154
legalize_fp_placement	1155
legalize_placement	1156
lib2saif	1158
license_users	1160
link	1162
link_physical_library	1168
list_attributes	1169
list_designs	1172
list_drc_error_types	1174
list_files	1175
list_floorplan_data	1176
list_instances	1178
list_libs	1181
list_licenses	1183
list_mw_cels	1184
list_partition_data	1186
lminus	1190
load_fp_rail_map	1192
load_of	1195
load_upf	1196
magnet_placement	1201
man	1204
map_freeze_silicon	1205
map_isolation_cell	1207

map_level_shifter_cell	1209
map_power_switch.	1211
map_retention_cell	1213
mark_clock_tree	1215
mem	1218
merge_clock_gates	1220
merge_flip_chip_nets	1222
merge_fp_hierarchy	1224
merge_net_shapes.	1227
merge_saif	1229
move_mw_cel_origin	1233
move_objects	1235
move_pins_on_edge	1238
mw_cel_collection	1240
name_format	1242
open_mw_cel	1244
open_mw_lib	1246
optimize_clock_tree	1248
optimize_dft	1252
optimize_flip_chip_route.	1254
optimize_fp_timing	1256
optimize_netlist_hierarchy	1258
optimize_power_switch	1262
optimize_pre_cts_power.	1265
optimize_wire_via.	1267
order_rp_groups.	1269
pack_fp_macro_in_area.	1271
place_flip_chip_array	1273
place_flip_chip_ring	1275
place_fp_pins	1278
place_freeze_silicon.	1281
place_io_pads	1282
place_opt	1284
place_opt_feasibility	1286
preroute_instances.	1288
preroute_standard_cells.	1295
print_message_info	1302
print_proc_new_vars	1304

print_suppressed_messages	1306
printenv	1307
printvar	1308
proc_args	1310
proc_body	1311
process_particle_probability_file	1312
propagate_constraints	1314
propagate_ilm	1320
propagate_switching_activity	1322
psynopt	1325
push_down_fp_objects	1328
push_flip_chip_route	1333
push_up_fp_objects	1336
pwd	1340
query_objects	1341
quit	1344
read_antenna_violation	1345
read_aocvm	1346
read_ddc	1348
read_def	1349
read_drc_error_file	1352
read_file	1354
read_flip_chip_bumps	1358
read_floorplan	1361
read_io_constraints	1363
read_lib	1365
read_mw_eco_list	1369
read_parasitics	1372
read_partition	1374
read_rail_maps	1375
read_saif	1377
read_sdc	1381
read_sdf	1386
read_tdf_ports	1389
read_verilog	1391
read_zrt_route_guidance	1393
rebuild_mw_lib	1395
recover_tie_connection	1397

redirect	1399
redo	1403
reduce_fp_rail_stacked_via	1404
refine_fp_macro_channels	1409
refine_placement	1410
remove_all_spacing_rules	1412
remove_annotated_check	1413
remove_annotated_delay	1415
remove_annotated_transition	1418
remove_annotations	1419
remove_antenna_rules	1421
remove_aocvm	1422
remove_attribute	1424
remove_base_arrays	1426
remove_bounds	1427
remove_buffer	1429
remove_buffer_tree	1432
remove_bus	1434
remove_case_analysis	1436
remove_cell	1438
remove_cell_degradation	1440
remove_cell_sites	1441
remove_cell_vt_type	1442
remove_checkpoint_designs	1443
remove_clock	1444
remove_clock_gates	1445
remove_clock_gating_check	1447
remove_clock_groups	1449
remove_clock_latency	1451
remove_clock_mesh	1453
remove_clock_sense	1455
remove_clock_transition	1457
remove_clock_tree	1458
remove_clock_tree_exceptions	1460
remove_clock_tree_options	1463
remove_clock_uncertainty	1465
remove_congestion_options	1468
remove_cts_scenario	1469

remove_dangling_wires	1470
remove_data_check	1473
remove_design	1475
remove_diode	1477
remove_disable_clock_gating_check	1479
remove_disable_timing	1480
remove_distributed_hosts	1482
remove_distributed_route	1483
remove_dont_touch_placement	1484
remove_dp_int_round	1485
remove_drc_error	1486
remove_driving_cell	1487
remove_edit_groups	1489
remove_filler_with_violation	1491
remove_flip_chip_route	1492
remove_fp_block_shielding	1494
remove_fp_feedthroughs	1496
remove_fp_pin_constraints	1499
remove_fp_pin_overlaps	1501
remove_fp_plan_group_padding	1503
remove_fp_rail_stacked_via	1504
remove_fp_rail_voltage_area_constraints	1505
remove_fp_relative_location	1506
remove_fp_virtual_pad	1507
remove_fp_voltage_area_constraints	1509
remove_from_collection	1510
remove_from_rp_group	1512
remove_generated_clock	1514
remove_host_options	1516
remove_ideal_latency	1517
remove_ideal_net	1519
remove_ideal_network	1521
remove_ideal_transition	1523
remove_ignore_cell_timing	1525
remove_ignored_layers	1526
remove_input_delay	1528
remove_io_antenna_properties	1531
remove_io_constraints	1532

remove_io_pin_overlap	1534
remove_isolate_ports	1535
remove_isolation_cell	1536
remove_keepout_margin	1537
remove_left_right_filler_rule	1539
remove_level_shifters	1540
remove_license	1541
remove_map_power_switch	1542
remove_mim_property	1544
remove_mw_cel	1546
remove_net	1549
remove_net_routing	1551
remove_net_routing_layer_constraints	1552
remove_net_shape	1553
remove_net_timing_spacing	1555
remove_objects	1556
remove_output_delay	1557
remove_partition	1559
remove_pg_network	1560
remove_physical_bus	1562
remove_pin_guides	1564
remove_pin_name_synonym	1566
remove_placement	1569
remove_placement_blockage	1570
remove_plan_groups	1572
remove_pnet_options	1573
remove_port	1574
remove_power_domain	1576
remove_power_net_info	1579
remove_power_switch	1580
remove_preferred_routing_direction	1582
remove_propagated_clock	1583
remove_qor_snapshot	1585
remove_rail_maps	1586
remove_route_by_type	1587
remove_route_guide	1589
remove_routing_blockage	1591
remove_routing_rules	1592

remove_row_type	1593
remove_rp_group_options	1594
remove_rp_groups	1596
remove_scaling_lib_group	1598
remove_scan_def	1599
remove_scan_pin_type	1600
remove_scenario	1601
remove_sdc	1602
remove_site_row	1603
remove_skew_group	1604
remove_stdcell_filler	1605
remove_supply_net	1606
remove_supply_port	1608
remove_target_library_subset	1610
remove_terminal	1611
remove_text	1612
remove_tie_cells	1614
remove_track	1616
remove_unconnected_ports	1618
remove_user_shape	1620
remove_via	1621
remove_voltage_area	1623
remove_vt_filler_rule	1625
remove_well_filler	1626
remove_xtalk_prop	1627
remove_zrt_filler_with_violation	1628
remove_zrt_redundant_shapes	1630
rename	1632
rename_mw_cel	1633
rename_mw_lib	1635
replace_power_switch	1636
report_adjusted_endpoints	1639
report_ahfs_options	1641
report_annotated_check	1642
report_annotated_delay	1644
report_annotated_transition	1646
report_antenna_ratio	1647
report_antenna_rules	1648

report_aocvm	1649
report_area	1653
report_attribute	1656
report_bounds	1659
report_buffer_tree	1661
report_buffer_tree_qor	1663
report_bus	1665
report_case_analysis	1667
report_cbt_options	1669
report_cell	1670
report_cell_physical	1674
report_cell_vt_type	1677
report_change_list	1678
report_check_library_options	1679
report_clock	1681
report_clock_gating	1684
report_clock_gating_check	1693
report_clock_timing	1696
report_clock_tree	1708
report_clock_tree_optimization_options	1714
report_clock_tree_power	1716
report_congestion	1722
report_congestion_options	1726
report_constraint	1728
report_critical_area	1736
report_crpr	1739
report_cts_batch_mode	1744
report_delay_calculation	1745
report_delay_estimation_options	1750
report_design	1752
report_design_lib	1755
report_design_physical	1757
report_direct_power_rail_tie	1761
report_disable_timing	1762
report_distributed_hosts	1764
report_distributed_route	1766
report_drc_error_type	1767
report_droute_options	1769

report_eco_history	1771
report_edit_groups	1773
report_error_coordinates	1775
report_extraction_options	1777
report_fast_mode	1779
report_feasibility_options	1780
report_filler_placement	1782
report_flip_chip_bump_attributes	1784
report_flip_chip_driver_bump	1785
report_flip_chip_type	1787
report_floorplan_data	1789
report_fp_clock_plan_options	1792
report_fp_macro_array	1793
report_fp_macro_options	1794
report_fp_pin_constraints	1796
report_fp_placement	1798
report_fp_placement_strategy	1800
report_fp_rail_constraints	1802
report_fp_rail_strategy	1804
report_fp_rail_voltage_area_constraints	1807
report_fp_relative_location	1809
report_fp_shaping_strategy	1810
report_fp_voltage_area_constraints	1812
report_groute_options	1813
report_hierarchy	1814
report_host_options	1816
report_ideal_network	1817
report_ignored_layers	1821
report_ilm	1822
report_interclock_relation	1826
report_internal_loads	1828
report_io_antenna_properties	1830
report_io_constraints	1832
report_isolate_ports	1834
report_isolated_via	1835
report_isolation_cell	1839
report_keepout_margin	1842
report_latency_adjustment_options	1844

report_lcc_hotspot	1845
report_left_right_filler_rule	1847
report_level_shifter	1848
report_lib	1852
report_milkyway_version	1867
report_mim	1869
report_mode	1871
report_mpc_macro_array	1873
report_mpc_macro_options	1875
report_mpc_options	1877
report_mpc_pnet_options	1879
report_mpc_port_options	1880
report_mpc_rectilinear_outline	1882
report_mpc_ring_options	1883
report_mtcmos_pna_strategy	1884
report_mw_design_ecos	1886
report_mw_lib	1888
report_name_rules	1890
report_names	1893
report_net	1895
report_net_changes	1900
report_net_fanout	1902
report_net_routing_layer_constraints	1905
report_net_routing_rules	1906
report_noise	1909
report_noise_calculation	1912
report_operating_conditions	1915
report_optimize_dft_options	1917
report_optimize_pre_cts_power_options	1918
report_parameter	1919
report_path_group	1921
report_pg_net	1923
report_physical_bus	1925
report_physical_signoff_options	1927
report_pin_guides	1929
report_pin_name_synonym	1931
report_pin_shape	1933
report_placement_utilization	1934

report_pnet_options	1936
report_port	1938
report_port_protection_diodes	1944
report_power	1945
report_power_calculation	1953
report_power_domain.	1959
report_power_gating.	1964
report_power_guidefp.	1966
report_power_net_info	1967
report_power_options.	1969
report_power_pin_info	1970
report_power_switch	1975
report_preferred_routing_direction.	1977
report_preroute_drc_strategy.	1978
report_primetime_options.	1980
report_pst	1981
report_qor	1983
report_qor_snapshot	1987
report_qtm_model	1989
report_rail_options	1992
report_reference	1994
report_retention_cell.	1997
report_route_opt_strategy	2000
report_route_options	2001
report_route_zrt_common_options.	2002
report_route_zrt_detail_options	2003
report_route_zrt_global_options.	2004
report_route_zrt_track_options.	2005
report_routing_rules	2006
report_rp_group_options	2007
report_saif.	2009
report_scan_chain	2013
report_scenario_options.	2015
report_scenarios.	2018
report_si_options	2020
report_signal_em	2022
report_skew_group.	2024
report_spacing_rules	2026

report_split_clock_gates_options	2028
report_starrcxt_options	2029
report_supply_net	2030
report_supply_port	2032
report_target_library_subset	2034
report_threshold_voltage_group	2035
report_tie_nets	2037
report_timing	2038
report_timing_derate	2051
report_timing_requirements	2053
report_tlu_plus_files	2057
report_track	2058
report_transitive_fanin	2060
report_transitive_fanout	2062
report_units	2065
report_voltage_area	2066
report_vt_filler_rule	2068
report_write_stream_options	2069
report_xtalk_route_options	2070
report_zrt_net_properties	2071
reset_clock_tree_optimization_options	2072
reset_clock_tree_options	2075
reset_clock_tree_references	2080
reset_cts_batch_mode	2083
reset_design	2084
reset_fp_clock_plan_options	2087
reset_latency_adjustment_options	2089
reset_mode	2090
reset_path	2092
reset_split_clock_gates_options	2096
reset_switching_activity	2097
reset_timing_derate	2099
reset_upf	2100
resize_objects	2102
resize_polygon	2105
restore_design_settings	2108
rotate_objects	2109
route_area	2112

route_auto	2114
route_detail.....	2116
route_differential.....	2118
route_eco	2120
route_flip_chip	2123
route_fp_proto	2126
route_global	2128
route_group	2130
route_htree.....	2132
route_mesh_net	2134
route_opt	2136
route_rc_reduction	2140
route_search_repair	2142
route_spreadwires	2144
route_track	2146
route_widen_wire	2147
route_zrt_auto	2149
route_zrt_clock_tree.....	2151
route_zrt_detail.....	2152
route_zrt_eco	2155
route_zrt_global	2157
route_zrt_group	2159
route_zrt_track	2161
route_zrt_with_route_guidance	2162
rp_group_inclusions	2163
rp_group_instantiations	2165
rp_group_references	2167
run_parallel_jobs	2169
run_signoff	2174
save_design_settings.....	2177
save_mw_cel	2178
save_qtm_model	2182
save_upf.....	2184
select_mim_master_instance.....	2185
send_flow_status	2187
set_active_scenarios	2189
set_ahfs_options	2191
set_always_on_cell	2196

set_always_on_strategy	2197
set_annotated_check	2199
set_annotated_delay	2202
set_annotated_transition	2205
set_aocvm_coefficient	2207
set_attribute	2209
set_auto_disable_drc_nets	2211
set_buffer_opt_strategy	2214
set_case_analysis	2215
set_cbt_options	2217
set_cell_degradation	2218
set_cell_internal_power	2220
set_cell_location	2222
set_cell_row_type	2223
set_cell_type	2224
set_cell_vt_type	2226
set_check_library_options	2227
set_checkpoint_strategy	2238
set_child_terminal	2241
set_chiplevel_pad_physical_constraints	2244
set_clock_gating_check	2246
set_clock_gating_registers	2249
set_clock_groups	2251
set_clock_latency	2254
set_clock_sense	2257
set_clock_transition	2259
set_clock_tree_exceptions	2261
set_clock_tree_optimization_options	2268
set_clock_tree_options	2273
set_clock_tree_references	2280
set_clock_uncertainty	2282
set_combinational_type	2286
set_congestion_options	2288
set_connection_class	2290
set_context_margin	2292
set_cost_priority	2294
set_critical_range	2296
set_cts_batch_mode	2298

set_cts_scenario.	2299
set_current_command_mode.	2301
set_data_check	2303
set_default_drive	2306
set_default_driving_cell	2308
set_default_fanout_load.	2311
set_default_input_delay	2313
set_default_load	2314
set_default_output_delay	2316
set_delay_calculation	2317
set_delay_estimation_options	2319
set_design_license.	2322
set_design_top	2324
set_die_area.	2325
set_direct_power_rail_tie	2326
set_disable_clock_gating_check	2327
set_disable_timing	2329
set_distributed_route	2331
set_domain_supply_net	2334
set_dont_touch.	2336
set_dont_touch_network	2338
set_dont_touch_placement	2340
set_dont_use	2341
set_dp_int_round	2343
set_drive.	2346
set_driving_cell.	2348
set_droute_options.	2353
set_equal	2355
set_error_view_property.	2356
set_extraction_options	2358
set_false_path	2363
set_fanout_load	2369
set_fast_mode	2371
set_feasibility_options	2372
set_fix_hold	2377
set_fix_hold_options.	2379
set_fix_multiple_port_nets	2381
set_flip_chip_bump_attributes	2383

set_flip_chip_cell_site.	2385
set_flip_chip_driver_array	2387
set_flip_chip_driver_island.....	2389
set_flip_chip_driver_ring.....	2392
set_flip_chip_grid	2395
set_flip_chip_options	2397
set_flip_chip_type.....	2399
set_fp_base_gate.....	2401
set_fp_black_boxes_estimated	2402
set_fp_black_boxes_unestimated	2403
set_fp_block_ring_constraints	2404
set_fp_clock_plan_options.....	2407
set_fp_flow_strategy.....	2409
set_fp_macro_array	2412
set_fp_macro_options	2415
set_fp_pin_constraints	2418
set_fp_placement_strategy	2426
set_fp_power_pad_constraints.....	2429
set_fp_rail_constraints	2431
set_fp_rail_region_constraints	2437
set_fp_rail_strategy	2439
set_fp_rail_voltage_area_constraints.....	2447
set_fp_relative_location	2456
set_fp_shaping_strategy	2459
set_fp_trace_mode.....	2461
set_fp_voltage_area_constraints	2463
set_groute_options.....	2466
set_hierarchy_color	2468
set_host_options	2470
set_ideal_latency	2474
set_ideal_net	2476
set_ideal_network.....	2478
set_ideal_transition.....	2481
set_ignore_cell_timing	2483
set_ignored_layers.....	2484
set_input_delay	2486
set_input_transition	2490
set_inter_clock_delay_options.....	2492

set_isolate_ports	2495
set_isolation	2497
set_isolation_cell	2499
set_isolation_control	2501
set_keepout_margin	2503
set_latency_adjustment_options	2506
set_left_right_filler_rule	2507
set_level_shifter	2509
set_level_shifter_cell	2512
set_level_shifter_strategy	2514
set_level_shifter_threshold	2516
set_lib_attribute	2518
set_lib_cell_spacing_label	2520
set_load	2522
set_local_link_library	2525
set_logic_dc	2527
set_logic_one	2529
set_logic_zero	2531
set_macro_cell_bound_spot	2533
set_max_area	2535
set_max_capacitance	2537
set_max_delay	2539
set_max_dynamic_power	2545
set_max_fanout	2547
set_max_leakage_power	2549
set_max_lvth_percentage	2551
set_max_net_length	2553
set_max_time_borrow	2555
set_max_total_power	2557
set_max_transition	2559
set_mcm_job_options	2561
set_message_info	2563
set_min_capacitance	2565
set_min_delay	2567
set_min_library	2572
set_mode	2574
set_mpc_macro_array	2575
set_mpc_macro_options	2578

set_mpc_options	2582
set_mpc_pnet_options	2587
set_mpc_port_options	2590
set_mpc_rectilinear_outline	2594
set_mpc_ring_options	2596
set_mtcmos_pna_strategy	2598
set_multicycle_path	2601
set_mw_lib_reference	2608
set_mw_technology_file	2610
set_name	2612
set_net_aggressors	2614
set_net_routing_layer_constraints	2615
set_net_routing_rule	2617
set_object_boundary	2619
set_object_fixed_edit	2621
set_object_shape	2622
set_object_snap_type	2626
set_operand_isolation_scope	2629
set_operating_conditions	2631
set_opposite	2635
set_optimize_dft_options	2636
set_optimize_pre_cts_power_options	2637
set_output_delay	2639
set_pad_physical_constraints	2644
set_parameter	2647
set_partition_data	2649
set_pg_pin_model	2652
set_physical_signoff_options	2654
set_physopt_cpulimit_options	2657
set_pin_model	2660
set_pin_name_synonym	2662
set_pin_physical_constraints	2664
set_place_opt_cts_strategy	2667
set_pnet_options	2669
set_port_fanout_number	2675
set_port_location	2677
set_power_gating_signal	2679
set_power_gating_style	2681

set_power_guide	2684
set_power_net_to_voltage_area	2686
set_power_options	2687
set_power_switch_cell	2689
set_prefer	2691
set_preferred_routing_direction	2693
set_preroute_advanced_via_rule	2694
set_preroute_drc_strategy	2697
set_preroute_special_rules	2701
set_primetime_options	2711
set_propagated_clock	2713
set_pulse_clock_cell	2715
set_qtm_global_parameter	2717
set_qtm_port_drive	2720
set_qtm_port_load	2722
set_qtm_technology	2724
set_rail_options	2727
set_register_type	2734
set_related_supply_net	2737
set_relative_always_on	2739
set_resistance	2741
set_retention	2743
set_retention_cell	2745
set_retention_control	2747
set_route_flip_chip_options	2749
set_route_mode_options	2752
set_route_opt_strategy	2753
set_route_options	2756
set_route_type	2760
set_route_zrt_common_options	2761
set_route_zrt_detail_options	2770
set_route_zrt_global_options	2778
set_route_zrt_track_options	2781
set_row_type	2782
set_rp_group_options	2784
set_scaling_lib_group	2788
set_scan_pin_type	2790
set_scenario_options	2792

set_scope	2796
set_si_options	2798
set_size_only	2800
set_skew_group	2802
set_spacing_label_rule	2803
set_split_clock_gates_options	2805
set_starrcxt_options	2806
set_switching_activity	2808
set_synlib_dont_get_license	2813
set_target_library_subset	2815
set_timing_derate	2817
set_timing_ranges	2820
set_tlu_plus_files	2823
set_true_delay_case_analysis	2826
set_unconnected	2828
set_undoable_attribute	2830
set_ungroup	2832
set_units	2834
set_user_grid	2836
set_via_array_size	2839
set_voltage	2840
set_voltage_model	2842
set_vt_filler_rule	2843
set_write_stream_options	2845
set_xtalk_route_options	2851
set_zero_interconnect_delay_mode	2853
set_zrt_net_properties	2854
setenv	2855
shape_fp_blocks	2857
shell_is_in_upf_mode	2860
signoff_drc	2861
signoff_metal_fill	2865
signoff_opt	2871
size_cell	2875
sizeof_collection	2877
skew_opt	2879
slot_wire	2888
snap_objects	2893

sort_collection.	2895
sort_fp_pins	2897
source.	2898
split_clock_gates	2900
split_clock_net	2902
split_mw_lib	2904
split_net	2906
split_objects	2909
spread_spare_cells	2911
spread_zrt_wires	2913
stretch_wire	2915
sub_designs_of	2917
sub_instances_of	2919
suppress_message	2921
swap_cell_locations	2922
syntax_check	2924
synthesize_fp_rail.	2926
trace_scan_chain	2934
translate_zrt_parameters	2936
trim_fill_eco	2937
unalias	2939
uncommit_fp_soft_macros	2940
undefine_bus	2942
undo	2944
undo_config	2945
undo_mark	2947
ungroup	2948
uniquify	2952
uniquify_fp_mw_cel	2955
unset_hierarchy_color	2957
unset_power_guide	2958
unsuppress_message	2959
update_bounds.	2960
update_clock_latency.	2962
update_flip_chip_pin_locations	2963
update_lib.	2964
update_lib_model.	2967
update_lib_pg_pin_model	2969

update_lib_pin_model	2971
update_lib_voltage_model	2973
update_physical_bus	2974
update_timing	2977
update_voltage_area	2978
verify_drc	2980
verify_lvs	2985
verify_pg_nets	2988
verify_route	2990
verify_zrt_route	2993
which	2995
widen_zrt_wires	2997
window_stretch	2999
write	3001
write_def	3005
write_design_lib_paths	3009
write_design_settings	3011
write_environment	3013
write_flip_chip_bumps	3015
write_flip_chip_nets	3017
write_floorplan	3018
write_interface_timing	3022
write_io_constraints	3024
write_lib	3027
write_lib_specification_model	3031
write_link_library	3032
write_mw_lib_files	3034
write_parasitics	3036
write_physical_constraints	3039
write_physical_script	3041
write_plib	3043
write_qtm_model	3045
write_route	3047
write_rp_groups	3049
write_saif	3052
write_script	3055
write_sdc	3058
write_sdf	3062

write_stream.....	3064
write_verilog.....	3066

add_clock_drivers

Adds multiple levels of drivers into your design and connects them together. These drivers are suitable for driving a clock mesh.

SYNTAX

```
status add_clock_drivers
-load net_or_pins -prefix prefix_string
  | -remove_display
[-configuration list_of_groups
  | -external_configuration config_file]
[-driver_type library_cell]
[-avoid cells]
[-max_displacement distance]
[-offset {x_distance y_distance}]
[-use_common_bbox]
[-check_only]
[-verbose]
```

Data Types

<i>net_or_pins</i>	collection
<i>prefix_string</i>	string
<i>list_of_groups</i>	list
<i>config_file</i>	string
<i>library_cell</i>	string
<i>cells</i>	collection
<i>x_distance</i>	float
<i>y_distance</i>	float

ARGUMENTS

-load *net_or_pins*

Specifies the overall driver pin and the set of overall load pins. When you specify a net, it must have one driver pin and typically has many input (load) pins.

If you only want to work on a subset of the pins of a net, use the driving pin of the net and the pins that you want to process as the value of this option.

This option and the **-remove_display** option are mutually exclusive. You must specify one of them.

-prefix *prefix_string*

Specifies the prefix for all newly created nets and cells. All newly created nets and cells also have a suffix that denotes their level and unique ID. You should not use this prefix for any other nets or cells in the design. This is a global option: it impacts all created drivers.

There is no default prefix. If you specify the **-load** option, you must also specify this option.

-remove_display

Removes the previously annotated tentative mesh drivers from the GUI. This

option works only when the GUI is open.

-configuration *list_of_groups*

Defines groups of drivers to be added.

This option accepts a list of any length. Each element in the list defines a group of drivers to be added, and each element must itself be a Tcl list. For details about the configuration syntax, see the CONFIGURATION STRUCTURE section below.

This option and the **-external_configuration** option are mutually exclusive; you can specify only one.

-external_configuration *config_file*

Specifies the configuration in a Tcl text file.

Each line of text in the file is considered a Tcl command, and must begin with the **add_clock_driver_group** command name. The syntax and semantics of this file is identical to the **-configuration** option. For details about the configuration syntax, see the CONFIGURATION STRUCTURE section below.

(Note that the command **add_clock_drivers_group** is not available unless processed through the **-external_configuration** option.)

This option and the **-configuration** option are mutually exclusive; you can specify only one.

-driver_type *library_cell*

Specifies the default driver cell to use. The specified driver cell must have exactly one input and one output.

The configuration can override the default driver cell for a specific group of drivers. If you do not specify this option, each group in the configuration must explicitly specify a driver type.

-avoid *cells*

Prevents drivers from being inserted at the top of a large macro block. You can specify any number of cells as a list or a collection.

-max_displacement *distance*

Specifies the maximum allowed displacement in the x- and y-directions from the position specified in the configuration. The distance value is in library units.

When the tool inserts a driver, it starts at the position specified in the configuration and tries to find an unblocked, legal location within the distance specified by this option. If the tool cannot find an unblocked location within this distance, it does not insert that driver.

-offset {*x_distance y_distance*}

Specifies the offset to be added to each driver location after positioning the driver, but before searching for an unblocked, legal location. The distance values are in library units.

You can use this option to place drivers on a grid so that they are in the center of each horizontal wire or in the middle of each gap in a mesh.

If you use both this option and the **-offset** option for a specific group in the configuration, the two offsets are added together.

-use_common_bbox

Specifies a single bounding box of load pins for all groups of drivers.

Normally each group of drivers in the configuration uses the geometry of the group below as the bounding box for placement. The higher the hierarchy level

is, the smaller the bounding box area becomes. For some layouts, it is more convenient to have one common bounding box. You should use this option for H-tree layouts.

-check_only

Displays a short report of the number of drivers to be created at each level and their fanouts (without creating drivers). The report is done before the legalization of cell locations, so the values are approximate. Use this option to test and tune your configuration without actually inserting drivers.

When the GUI is open, this option displays the locations of tentative mesh drivers in the GUI. Use the **-remove_display** option to undo the display.

-verbose

Displays additional information about the way drivers are being created and positioned.

DESCRIPTION

If you need to drive a large load of many pins with minimum skew, you generally need to create multiple levels of drivers. You can either manually create such a hierarchy of drivers by using **insert_buffer** repeatedly, or let command like **compile_clock_tree** automatically create them. However, if the load pins are uniformly distributed, **add_clock_drivers** provides some advantages. This command is especially suitable for driving a large clock mesh net.

This command accepts one net, usually a high-fanout net. The net must have exactly one driver pin. When the command is done, the clock net is still connected to its original driver pin, but the original load pins are connected to one or more new nets.

If you use the command to drive a clock mesh (as opposed to a collection of unconnected input pins), you must use the **-short_outputs** option at the lowest level. In a mesh, all load pins are connected by wire straps. To design a mesh, you must add the mesh straps before running this command. If you do not use **-short_outputs**, the loads are connected to the newly created drivers, not to each other. Running this command without this option is appropriate for generating conventional clock tree circuitry where the driver outputs are not shorted together.

The specified command options apply globally to the entire hierarchy of drivers. The configuration lets you specify options in more detail. The two sets of options complement each other. The global options specify the root and loads of the driver tree to be created and affect all drivers in the tree, while the configuration specifies how small a group of drivers or individual drivers should be created.

You must specify a prefix for naming new nets and cells created by this command. The prefix must not be used by other nets or cells in the design. The names of the newly created cells have the following format:

`<prefix>_L<level_number>_<group_number>_<column>_<row>`

For example,

`clk_a_L1_1_2_3`

driv_L4_3_7_2

Using a unique prefix enables you to perform operations such as legalizing, routing, or removing the nets and cells that are created by this command.

You should use a short prefix because the command creates many cells and nets.

When nets are created, they share the same name as the cell that drives them. However, if you are driving a mesh, you must specify the name of the large mesh net separately.

You can run this command multiple times with the **-check_only** option to verify that your circuitry has valid inputs and a reasonable structure with legal fanouts and balancing before running the command to insert the drivers and to connect the nets.

You can use this command to insert the cells that drive a mesh net (the mesh drivers) and then use a separate command to create the cells that drive the mesh drivers (the premesh tree). For example, use the **compile_premesh_tree** command after the **add_clock_drivers** command inserts the mesh drivers. This two-step approach provides the advantage of analyzing the mesh driver circuitry and mesh straps before adding the premesh tree. If the mesh drivers and straps do not meet performance requirement, they should be redesigned. A premesh tree cannot improve a poorly designed mesh or set of premesh drivers.

CONFIGURATION STRUCTURE

The configuration is comprised of one or more driver group specifications.

Each group of drivers has a "level" and is specified by a list of options. The options all have the form **-option_name value**, just like a Tcl command. The order of the group specifications is not important. For example, you can specify the level 2 groupings before or after the level 1 groupings. Also, the order of the options with each grouping is not important. For example, you can specify the **-boxes** option before or after the **-driver_type** option. More importantly, you can specify multiple groupings at a single level. For example, you could have three groupings at level 5 with different geometries to fit unbalanced loads or avoid obstacles. Each grouping of drivers must be assigned to a level in the driver hierarchy. Level 0 is the original drive pin or port specified with the **-load** command option. You must not try to define any grouping at level 0. The input pins on the first level (level 1) are connected to the original driving net. The outputs of level 1 drivers are connected to the input pins on level 2, and so forth. Drivers in the highest numbered level drive the original load pins. At least one grouping must appear in the configuration, so there must be at least one grouping at level 1.

The clock tree is built from the bottom (the load pins) up. To add drivers, the tool must know what the drivers should drive and where to place them. The level number determines the default set of pins that should be driven. For each level, the default driven pins are all the pins on the level below, and the default driven region for each group is the bounding box of the driven pins. You can change the set of load pins for a group on the lowest level by specifying an explicit **-drives** option.

The default geometric boundary for placing the the new drivers is the same as the driven pin region. You can override this boundary for a group by using the **-boundary**

add_clock_drivers

option. You would typically override these specifications only when the design has obstacles or other complex geometry. If you want all the groups in the configuration to use the same boundary, use the **-use_common_bbox** option at the command level.

When you specify geometries, note that you specify a point as a Tcl list of two distances and a rectangle as a Tcl list of two points.

An example of a simple configuration is

```
{  {-level 1 -boxes {3 2}}
  {-level 2 -boxes {6 4}}
}
```

This creates two groups of drivers. The first group, on level 1, comprises six drivers (three columns and two rows) connected to the top-level driver.

The second group, on level 2, comprises 24 drivers, which are driven by the six drivers on level 1; thus, each of the level 1 drivers has a fanout of 4. The level 2 drivers are arrayed in six columns and four rows. They are spread uniformly over the region defined by the load pins specified in the **-load** command option, because these load pins are implicitly on level 3.

By default, each driver drives those pins that are closest to it.

The configuration file provides various options to help create drivers suitable for a variety of common situations.

For each driver group, you can specify the following configuration options:

```
-level level_number
[-boxes {columns rows}
 | -auto {columns rows}
 | -interior {columns rows}
 | -grid {columns rows}
 | -single_driver point
 | -coincident collection
 | -replicate]
[-drives rectangle | -load collection_of_pins]
[-boundary rectangle]
[-driver_type lib_cell]
[-offset {x_offset y_offset}]
[-short_outputs
  [-output_net_name string [-transfer_wires_from net]]]
```

-level *level_number*

Specifies where in the hierarchy to connect the group of drivers. You must specify this option for each grouping. Level 1 is the highest level. The input pins of level 1 drivers are driven by the overall driver pin and are therefore connected directly to the overall clock net. Level 2 drivers with are driven by level 1 drivers. The driver groupings with the highest level number drive the loads specified by the **-load** command option.

There must not be any gaps in the level numbers. Groups must exist for each level from level 1 to your highest level number. If there is a gap, the configuration is rejected and the command returns an error.

You can have any number of groups with the same level. This is useful if you

have complex geometry and want to specify particular drivers for particular parts of your design. Note that it would be very rare, but not illegal, to have a level number more than about 10, because each level typically has a fanout of approximately 4 to 8, and 10 levels could drive at least 262144 loads.

-boxes {columns rows}

Specifies an array of drivers that are spread across the rectangle that is the boundary of the group. When you use this option, the drivers are placed at the centers of the boxes that define the loads they serve.

For example, to place six drivers at {50 50}, {50 150}, {150 50}, {150 150}, {250 50}, and {250 150}, specify the following options:

-level 2 -boundary {{0 0} {300 200}} -boxes {3 2}

To picture the result, draw four vertical and three horizontal lines that form six boxes. The drivers are placed at the center of each box.

To specify a column of drivers, set the *columns* argument to 1 and specify the number of drivers that you want in the *rows* argument. To specify a row of drivers, set the *rows* argument to 1 and specify the number of drivers that you want in the *columns* argument. To specify a single driver, set the option argument {1 1}.

-auto {columns rows}

Specifies an array of drivers that are spread across the rectangle that is the boundary of the group. When you use this option, the tool automatically adjusts the driver positions to even out the distance from a driver to its loads.

To picture the result, represent the loads by drawing a pattern of 6 dots wide by 4 dots tall, which are spaced 100 units apart horizontally and vertically, starting at {0 0} and having the upper-right dot at {500 300}. The **-auto {3 2}** option adds six drivers centered in the middle of symmetrical clusters of 4 dots. The drivers are located at {50 50}, {250 50}, {450 50}, {50 250}, {250 250}, and {450 250}. Each driver drives four loads, with an equal distance to each of them.

To specify a column of drivers, set the *columns* argument to 1 and specify the number of drivers that you want in the *rows* argument. To specify a row of drivers, set the *rows* argument to 1 and specify the number of drivers that you want in the *columns* argument. To specify a single driver, set the option argument {1 1}.

-grid {columns rows}

Specifies an array of drivers that are spread on the edges and interior of the rectangle that is the boundary of the group. When you use this option, the drivers are placed on the perimeter of the boundary rectangle.

For example, to place six drivers at {0 0}, {150 0}, {300 0}, {0 200}, {150 200}, and {300 200}, specify the following options:

-boundary {{0 0} {300 200}} -grid {3 2}

To picture the result, draw three vertical and two horizontal lines that form two boxes. Each box is 150 units wide by 200 units tall. The drivers are placed at the intersections of the horizontal and vertical lines.

To specify a column of drivers, set the *columns* argument to 1 and specify the number of drivers that you want in the *rows* argument. To specify a row of drivers, set the *rows* argument to 1 and specify the number of drivers that you want in the *columns* argument. To specify a single driver, set the option argument {1 1}.

This option provides more detailed manual specification than the **-boxes** or **-**

auto options.

The **-grid** option might be more convenient than the **-boxes** option for placing drivers directly under specific wire segments in the clock wiring mesh. This would be useful for the lowest level that drives the externally specified load.

-interior {*columns rows*}

Specifies a regular array of drivers that are spread on the interior of the rectangle that is the boundary of the group. When you use this option, the drivers are placed only in the interior and not along the edges.

For example, to place two drivers at {100 100} and {200 100}, enter the following options:

```
-boundary {{0 0} {300 200}} -interior {2 1}
```

To picture the result, draw four vertical lines and three horizontal lines that form six boxes. Each box is 100 units wide by 100 units tall. The two drivers are placed at the four-way intersections of the horizontal and vertical lines in the middle.

To specify a column of drivers, set the *columns* argument to 1 and specify the number of drivers that you want in the *rows* argument. To specify a row of drivers, set the *rows* argument to 1 and specify the number of drivers that you want in the *columns* argument. To specify a single driver, set the option argument {1 1}.

The **-interior** option can be more convenient than the **-grid** option for placing drivers directly under specific wire intersections in the clock wiring mesh.

-single_driver *point*

Inserts a single driver at the specified point.

-coincident *collection*

Specifies that a new set of drivers should be allocated on a one-to-one basis with members of the specified collection. Typically, this is a collection of vias that are located at the intersections of the clock mesh. If the clock mesh has already been allocated, you can get the vias with the following command:

```
set vias [get_vias -of clock_net]
```

To specify this configuration to the **add_clock_drivers** command, enter **add_clock_drivers** -load clk \

```
-driver_type buffd7 -use_common_bbox -verbose -prefix ccc \
```

```
-config [list \
```

```
{-level 1 -boxes {1 1}} \
```

```
{-level 2 -boxes {2 2}} \
```

```
{-level 3 -boxes {4 4}} \
```

```
[list -level 4 -coincident $vias -short_outputs \
```

```
-output_net_name clk_mesh]]
```

-replicate

Specifies that a new set of drivers should be allocated on a one-to-one basis with the drivers on the level below the current level. Typically, this is used for allocating pairs of inverters.

-drives *rectangle*

Specifies that the group drives the loads in the specified rectangle.

By default, the group drives the loads in the entire bounding box of the load pins on the level below.

-load *collection_of_pins*

Specifies that the group drives the specified load pins.

You can use this configuration option only for groups at the lowest level that drive a subset of pins specified in the **-load** command option.

By default, the group drives all load pins on the level below.

-boundary *rectangle*

Specifies the boundary for the group of drivers.

By default, the boundary is determined by the set of load pins that the group serves.

For example, if you want to drive loads on the left side of your design with drivers located near the middle of your design, you can use the **-drives** configuration option to select the loads and the **-boundary** configuration option to specify where to put the drivers.

The **-boundary** configuration option directly impacts the operation of the **-boxes**, **-grid**, and **-auto** configuration options.

-driver_type *lib_cell*

Specifies the driver cell to use for the group of drivers.

By default, the group uses the driver cell specified by the **-driver_type** command option. If you do not specify the **-driver_type** command option, you must specify this option for each group in the configuration.

-offset {*x_distance y_distance*}

Specifies the distance, in library units, to shift each driver from the location calculated by the tool.

The tool calculates a preferred location for each driver in the group. When you use this option, the tool first shifts each location by the specified horizontal and vertical offset before searching for an unblocked location. You can use this configuration option with the **-boxes**, **-single_driver**, **-replicate**, and **-grid** configuration options. For example, you can use this configuration option with the **-replicate** configuration option to displace the newly created driver a small distance from the inverter that it drives.

If you specify both the **-offset** configuration option and the **-offset** command option, the drivers are shifted by the sum of the two offsets.

-short_outputs

Specifies that the outputs of all the drivers in this group are connected to a single net. This option is typically used when the load pins are joined together by a grid of wires in a mesh.

-output_net_name *string*

Specifies the name of the net driven by the driver group. This option is typically used when the load pins are joined together by a grid of wires in a mesh and you want to specify the name of the mesh net.

This option can be used only in conjunction with the **-short_outputs** configuration option.

-transfer_wires_from *from_net*

Specifies that all of the wires and vias on the specified net are transferred to the net specified on the **-output_net_name** option.

This option enables you to use the following flow:

```
create_clock_mesh -load clk -net clk -layers {METAL5 METAL4} \  
-num_straps {16 16}
```

echo Clock mesh wires are created now

```
add_clock_drivers -load clk -driver_type buffd7 -prefix ccc \  
  -config {      {-level 1 -boxes {1 1}}  
    {-level 2 -boxes {2 2} -short_outputs \  
      -output_net_name clk_mesh -transfer_wires_from clk}  
  }
```

This sequence creates the wires on the original clock net, `clk`, and then creates a 2-level hierarchy of clock drivers. The second level drives the newly created `clk_mesh` net and all of the wires and vias that were once on the `clk` net are now on the `clk_mesh` net. Note that you can save the design and exit the tool at the point marked "Clock mesh wires are created now". Because the wires are attached to the `clk` net, which is part of the design with a driver and load, the wires will not be lost during a save and restore cycle.

This option can be used only in conjunction with the **-output_net_name** configuration option.

LIMITATIONS

The quality of results is primarily determined by your specification. If your grouping provides too many or too few drivers or puts the drivers in the wrong location, the tree might be imbalanced or even illegal.

Although you can specify the driver groups and levels in any order, the actual construction of the clock tree is bottom up. That is, it starts with the load pins and works toward the root pin. This command does not optimize the overall geometry.

The placement legalization process is done after the tree is constructed internally, so the relationship between the drivers and their loads might not be as symmetric as simple geometry would suggest.

The presence of large obstructions or irregular geometry can limit the utility of this command.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example generates a hierarchy of drivers connected to a mesh net. You should run the **legalize_placement** command after running the **add_clock_drivers** command.

```
prompt> add_clock_drivers -load clk1 \  
  -prefix clkd_ -driver_type buf2 \  
  -configuration {      {-level 1 -boxes {6 4} -driver_type buff2}  
    {-level 2 -boxes {3 2} -driver_type buff2 \  
      -short_outputs -output_net_name clk1mesh}  
  }  
1
```

Report:

```
Level 0 (overall driver) was port "clk"
Level 1 had 6 cells, max fanout= 4, min fanout=4
Level 2 had 24 cells, max fanout= 6, min fanout=4
Level 3 (overall load) had 100 input pins
```

1

The following example generates a hierarchy of drivers connected to a large number of input pins, but only reports the result. In this case, two columns of drivers comprise level 1 and one column comprises level 2.

The grouping list is created with the Tcl **list** command, which allows the use of Tcl variables and other tool functions.

```
prompt> set loads [get_pins "fflop*/CP"]
prompt> set l2_bound_box [get_bounding_box [get_pins clk2*/CP]

prompt> set config [list
    {-level 1 -grid {2 1} -boundary {20 400 20 600}}
    [list -level 2 -grid {6 1} -boundary $l2_bound_box]
    {-level 2 -boxes {3 1}}
]

prompt> add_clock_drivers -load [get_net clkd] -prefix clkd_ \
    -driver_type buf2 -check_only -configuration $config
```

Report:

```
Level 0 (overall driver) was port "inv2/Y"
Level 1 had 2 cells, max fanout= 5, min fanout=4
Level 2 had 9 cells, max fanout= 5, min fanout=3
Level 3 (overall load) had 34 input pins
```

1

SEE ALSO

```
create_clock_mesh(2)
route_mesh_net(2)
compile_premesh_tree(2)
analyze_subcircuit(2)
compile_clock_tree(2)
```

add_distributed_hosts

Adds one or more distributed hosts for distributed jobs.

SYNTAX

```
status add_distributed_hosts
[-target PrimeTime | StarRCXT | all]
[-32bit]
[-enable_ssh]
-farm lsf | grd | generic | now
[-setup_path setup_path]
[-num_of_hosts count]
[-options string]
[-submission_script submission_script]
[work_station_name]
```

ARGUMENTS

- target PrimeTime | StarRCXT | all
Specifies the target application for this configuration. Valid values are PrimeTime, StarRCXT, all. Default is all.
- 32bit
This switch is currently not supported in **IC Compiler**.
- enable_ssh
By default, remote processes are run with **rsh**. Enable this switch will ask **IC Compiler** to use **ssh** to launch remote processes.
- farm lsf | grd | generic | now
This option is used to specify the type of the farm from which the distributed hosts are to be acquired.
- setup_path path
This option specifies the path to the submission scripts/executables for farms of type lsf/grd/generic. The path should contain the following scripts/executables.
-farm lsf the setup_path should contain "qsub", "qdel" -farm grd the setup_path should contain "bsub", "bkill" -farm generic the setup_path should contain "gsub", "gdel"
The -setup_path option is not valid with -farm now
- [num_of_hosts count]
The user can specify the number of hosts to use in an lsf, grd, generic, or now compute environment. The number must be at least 1. If this option is not specified then the number of hosts is 1.
- [options string]
This option is used only in conjunction with -farm lsf | grd | generic. It is a list of arguments to be passed off to the submission_script used for job submission to a lsf, grd or generic queue.

-[submission_script]

The user can explicitly specify the script to call when submitting jobs to farm types grd, lsf or generic using this option. When both the -setup_path and -submission_script options are used simultaneously, the path and submission script specification are combined to define the script used when submit jobs to the compute resource being specified.

The -submission_script option is not valid with -farm now

work_station_name

Specifies a single UNIX machine. This option is used only in conjunction with -farm now.

DESCRIPTION

The options for **add_distributed_hosts** depend on whether the user has an on-site load balancing capability and whether they wish to use it or not.

If on-site load balanced facilities are available these should be specified using the -farm lsf, -farm grd or -farm generic options, depending on the type of resource available.

If no on-site load balancing capabilities are available or there are other compute resources available that are not under load balancing control, these should be specified using the -farm now hostname method.

All combinations of resources are concurrently supported by the command, however, each separate type must be specified independently with a separate invocation of the **add_distributed_hosts** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

In the following example, two hosts are added for a distributed command. The first host, platinum1, is specified to have 2 hosts and the second, platinum2, is specified to have 4 hosts. The processes on platinum1 will be 64 bit where as the processes on platinum2 will be 32 bit. Notice that the actual number of processors in platinum1 and platinum2 is irrelevant but for optimal performance the number or processes allocated should match the processors in each machine i.e. 2 processors in platinum1 and 4 processors in platinum2.

```
prompt> add_distributed_hosts -farm now -num_of_hosts 2 platinum1
1
prompt> add_distributed_hosts -farm now -num_of_hosts 4 platinum2 -32bit
1
```

To use this command where on-site load balancing capabilities (LSF/GRD/GENERIC) are

add_distributed_hosts

in place, use options `-farm grd`, `-farm lsf`, or `-farm generic`. In the example below, the distributed hosts list is populated by defining four 32 bit hosts from an lsf farm, and two 64 bit hosts from a grd farm.

```
prompt> add_distributed_hosts -farm lsf -setup_path "/bin/lsf/"  
-options {-R "tmp>300" -q} -num_of_hosts slaves 4 -32bit  
1  
prompt> add_distributed_hosts -farm grd -setup_path "/bin/grd/"  
-options {-cwd -V -r y -P bnormal -l} -num_of_hosts slaves 2  
1
```

add_drc_error_detail

Adds additional details to an existing error object that is associated with the default error type class.

SYNTAX

```
status add_drc_error_detail  
-error_view mw_error_view  
-drc_error drc_error  
-rectangles rectangles  
[-net net]  
[-layer layer]
```

Data Types

<i>mw_error_view</i>	list or collection of one item
<i>drc_error</i>	list or collection of one item
<i>net</i>	list or collection of one item
<i>layer</i>	list or collection of one item

ARGUMENTS

-error_view *mw_error_view*

Specifies the error view that contains the error record to which the details are added. You may specify only a single error view; specifying more than one error view causes an error.
This is a required option.

-drc_error *drc_error*

Specifies the error object to which to add these details. You can specify a single error object; specifying more than one error object causes an error. You can add error details only to existing error objects. To specify the error object, use the collection returned by the **create_drc_error** command or the **get_drc_errors** command. The specified error object must be associated with the default error type class. For more information about error type classes, see the man page for the **create_drc_error_type** command.
This is a required option.

-rectangles *rectangles*

Specifies the rectangle shapes to add to the error object. You must specify one or more rectangles, which are specified in the following format:
{numRectangles *rectangle*+}
where each rectangle is specified as a list of four points, {llx lly urx ury}, which represent the lower-left x- and y-coordinates and the upper-right x- and y-coordinates.
This is a required option.

-net *net*

Specifies the net associated with the shapes. You can specify the net by providing a collection containing the net or by specifying the net name. You cannot use wildcards in the net name.

add_drc_error_detail

-layer layer

Specifies the layer associated with the shapes. You can specify the layer by using the layer name from the technology file or by using the **get_layers** command.

DESCRIPTION

This command adds shapes and associated details to an existing error object that is associated with the default error type class.

When an error type is created with the **create_drc_error_type** command, it is associated with one of the following error type classes: default, open, openlocator, short, and spacing. An error object is interpreted based on its error type class. Use this command only for adding details to error objects associated with the default error type class. To add details to error objects associated with the open error type class, use the **add_open_drc_error_detail** command. For the other error type classes, you specify the details when you create the error object; you cannot add details to an existing error object.

The command returns 1 if successful; 0 otherwise.

EXAMPLES

The following example creates an error record of the type "Pin made fat" in an error view, and then adds details to it:

```
prompt> set errId [create_drc_error -error_view "mydesign_mydrc.err" \  
-type "Pin made fat"]  
{25635}  
prompt> add_drc_error_detail -drc_error $errId \  
-error_view "mydesign_mydrc.err" \  
-net [get_nets myNetName] -layer [get_layers myLayerName] \  
-rectangles {1 {574.0700 430.3600 578.9700 433.4400}}
```

1

SEE ALSO

create_drc_error(2)
create_drc_error_type(2)
create_open_drc_error(2)
create_open_locator_drc_error(2)
create_short_drc_error(2)
create_spacing_drc_error(2)
add_open_drc_error_detail(2)
list_drc_error_types(2)
report_drc_error_type(2)
get_drc_errors(2)

add_end_cap

Adds end cap cells at both ends of a cell row.

SYNTAX

```
status add_end_cap
[-mode both | bottom_left | upper_right]
[-mirror]
[-respect_padding]
[-respect_blockage]
-lib_cell lib_cell_name
[-vertical_cells lib_cell_names [-fill_corner]]
[-respect_keepout]
[-ignore_soft_blockage]
[-skip_fixed]
```

Data Types

lib_cell_name collection of one cell

ARGUMENTS

- mode both | bottom_left | upper_right**
Specifies how to place end cap cells. The **both** value places the cell at both ends of the row. The **bottom_left** value places the cell at the bottom of a vertical row or to the left of a horizontal row. The **upper_right** value places the cell at the top of a vertical row or to the right of a horizontal row. The default is **both**.
- mirror**
Specifies to flip the orientation of the cell when the command places it. By default, this option is disabled.
- respect_padding**
Specifies to respect macro padding so that the command does not place end cap cells in the padding areas around a macro. By default, this option is disabled.
- respect_blockage**
Specifies to respect placement blockages so that the command does not place the end cap cells inside placement blockages. By default, this option is disabled.
- lib_cell *lib_cell_name***
Specifies the reference library cell to be used as horizontal end caps. Provide only one.
- vertical_cells *lib_cell_names***
Specifies the names of the reference library cells to be used as vertical end caps. By default, the tool does not add vertical end cap cells.

-fill_corner

Specifies whether to fill corners where vertical and horizontal end caps meet with horizontal end caps. You must use this option with the **-vertical_cells** option. By default, the tool does not fill corners.

-respect_keepout

Prevents placing end caps inside keepout margins around macro cells. Keepout margins on other types of cells do not apply.

-ignore_soft_blockage

Specifies only hard blockages are honored when used with the **-respect_blockage** option. You must use this option with the **-respect_blockage** option.

-skip_fixed

Specifies not to place end caps at locations that are occupied by fixed cells. If placing a horizontal end cap at either end of a row overlaps a fixed cell, no horizontal end cap is placed. By default, the tool does not respect fixed cells. You must use this option with the **-vertical_cells** option.

DESCRIPTION

This command allows you to add horizontal end cap cells at both ends of a cell row and, optionally, to add them along horizontal edges. An end cap cell is typically not a logic cell and is added before placement to serve a special purpose to the row, such as a decoupling capacitor. You should specify a proper end cap cell because the tool can accept any standard cell as an end cap cell.

You should specify the right size vertical end caps to avoid unfilled gaps. Vertical end caps are inserted in the order specified provided that the remaining gaps are larger than or equal to the smallest vertical end caps, or else the next in order will be used.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example adds horizontal end cap cells named MY_END_CAP to the current design.

```
prompt> add_end_cap -lib_cell MY_END_CAP
```

SEE ALSO

set_keepout_margin(2)

add_open_drc_error_detail

Adds additional details to an existing error object that is associated with the open error type class.

SYNTAX

```
status add_open_drc_error_detail  
-error_view mw_error_view  
-drc_error drc_error  
-node node_number  
-rectangles rectangles  
[-layer layer]
```

Data Types

<i>mw_error_view</i>	list or collection of one item
<i>drc_error</i>	list or collection of one item
<i>node_number</i>	integer
<i>rectangles</i>	list
<i>layer</i>	list or collection of one item

ARGUMENTS

- error_view *mw_error_view***
Specifies the error view in which to create the error record. You may specify only a single error view; specifying more than one error view causes an error. This is a required option.
- drc_error *drc_error***
Specifies the error object to which to add these details. You can specify a single error object; specifying more than one error object causes an error. You can add error details only to existing error objects. To specify the error object, use the collection returned by the **create_open_drc_error** command or the **get_drc_errors** command or use the object ID of the error object. The specified error object must be associated with the open error type class. For more information about error type classes, see the man page for the **create_drc_error_type** command. This is a required option.
- node *node_number***
Identifies the node to which the specified rectangle shapes belong. An open type error is described by two or more disjoint nodes, where a node is a set of connected net shapes. This option uniquely identifies the node, while the **-rectangles** option specifies the net shapes associated with the node. This is a required option.
- rectangles *rectangles***
Specifies the rectangle shapes to add to the error object. The format for specifying the rectangle shapes is
rectangles: {*num_rectangles rectangle*+}
rectangle: {*llx lly urx ury*}
This is a required option.

add_open_drc_error_detail

-layer layer

Specifies the layer associated with the shapes. You can specify the layer by using the layer name from the technology file or by using the **get_layers** command.

DESCRIPTION

This command adds shapes and associated details to an existing error object that is associated with the open error type class.

The command returns 1 if successful; 0 otherwise.

EXAMPLES

The following example opens an error view named mydesign_mydrc.err, creates an error type named "Open Net" in the open error type class, creates an error record of the "Open Net" type, and then adds details to the error record:

```
prompt> set cellId [open_mw_cel -not_as_current mydesign_mydrc.err]
{mydesign_mydrc}
prompt> set typeId [create_drc_error_type -name "Open Net"
    -class "open" \
    -info "There are opens in the net" -error_view $cellId]
1024
prompt> set openErrId [create_open_drc_error \
    -type $typeId -error_view $cellId]
{1280}
prompt> add_open_drc_error_detail -error_view $cellId \
    -drc_err $openErrId -node 1 -layer RV \
    -rectangles {2 {574.0700 430.3600 578.9700 430.3600} \
        {574.0700 436.3600 574.0700 453.3600}}
```

1

SEE ALSO

create_open_drc_error(2)
create_drc_error_type(2)
list_drc_error_types(2)
report_drc_error_type(2)
get_drc_errors(2)

add_pg_pin_to_db

Converts rail or non-pg_pin based logic library (db) into pg_pin based logic library.

SYNTAX

```
status add_pg_pin_to_db  
input_db_filename  
[-mw_library_name mw_lib_name]  
[-pg_map_file pg.map]  
-output pg_db_filename  
[-dirty_flow]  
[-verbose]  
[-pg_map_template pg_pin_map_template_filename]  
[-force_update]  
[-expanded]
```

ARGUMENTS

input_db_filename
Specifies the name of the input logic library (.db) file, which is in non-pg_pin-based format. It is mandatory.

-mw_library_name mw_lib_name
Specifies one or more Milkyway library name(s) (FRAM view) that correspond to the input db file.

-pgap_file pgap
Specifies the file name of mapping between non-pg_pin-based data and pg_pin-based data.

-output pg_db_filename
Specifies the name of new pg_pin-based db file that is generated after the conversion. It cannot be the same name as the input db file. This option is mandatory.

DESCRIPTION

Command **add_pg_pin_to_db** converts rail or non-pg_pin based logic library (db) into pg_pin based logic library. There are three cases: 1) with complete Milkyway library If all cells have single P/G rail (1P1G), -pg_map_file option is not required but -mw_library_name should be specified to derive the mapping from the Milkyway Fram view. If some cells have multiple P/G rails, -pg_map_file is required for those cells. 2) without Milkyway library (db only) If you only specify a logic library (.db), no matter if it is a single P/G rail library or not, you must specify -pg_map_file to make the conversion. 3) with partial Milkyway library If some cells exists in both logic library and Milkyway library while others only exist in logic library, -pg_map_file should be specified for the cells that are missing in the Milkyway library.

EXAMPLES

The following example converts the logic library *old.db* into pg_pin-based *pg.db*.

```
prompt> add_pg_pin_to_db old.db -mw_library_name {mw_lib}  
        -pg_map_file  pg.map -output pg.db
```

SEE ALSO

`add_pg_pin_to_lib(2)`

add_pg_pin_to_lib

converts rail or non-pg_pin based logic library (.lib) into pg_pin based logic library (.lib).

SYNTAX

```
status add_pg_pin_to_lib  
input_lib_filename  
[-mw_library_name mw_lib_name]  
[-pg_map_file pg.map]  
[-pg_map_template template_filename]  
[-expanded]  
-output pg_lib_filename  
[-dirty_flow]  
[-common_shell_path common_shell_path]  
[-verbose]  
[-force_update]
```

ARGUMENTS

input_lib_filename
Specifies the name of the input logic library (.lib) file, which is in non-pg_pin-based format. It is mandatory.

-mw_library_name *mw_lib_name*
Specifies one or more Milkyway library name(s) (FRAM view) that correspond to the input db file.

-pgap_file *pgap*
Specifies the file name for mapping between non-pg_pin-based data and pg_pin-based data including PM attributes.

-pg_map_template *template_filename*
Specifies the file name for map template.

-expanded
Specifies to generate expanded map template with wildcard expanded. By default it is OFF meaning compressed map template using wildcards will be generated.

-output *pg_lib_filename*
Specifies the name of new pg_pin-based lib file that is generated after the conversion. It cannot be the same name as the input lib file. This option is mandatory.

DESCRIPTION

Command **add_pg_pin_to_lib** converts rail or non-pg_pin based logic library (.lib) into pg_pin based logic library (.lib). There are three cases: 1) with complete Milkyway library If all cells have single P/G rail (1P1G), -pg_map_file option is not required but -mw_library_name should be specified to derive the mapping from the

Milkyway Fram view. If some cells have multiple P/G rails, `-pg_map_file` is required for those cells. 2) without Milkyway library (.lib only) If you only specify a logic library (.lib)and no Milkyway library, for a single P/G rail library or multi-rail library, you must specify `-pg_map_file` to make the conversion. 3) with partial Milkyway library If some cells exists in both logic library and Milkyway library while others only exist in logic library, `-pg_map_file` should be specified for the cells that are missing in the Milkyway library and for PM cells.

EXAMPLES

The following example converts the logic library `"old.lib"` into pg_pin-based `"pg.lib"`.

```
prompt> add_pg_pin_to_lib old.lib -pg_map_file pg.map
        -output pg.lib
```

SEE ALSO

`add_pg_pin_to_db(2)`

add_port_state

Adds state information to a supply port.

SYNTAX

```
string add_port_state  
supply_port_name  
-state {state_name state_value}
```

Data Types

<i>supply_port_name</i>	string
<i>state_name</i>	string
<i>state_value</i>	string

ARGUMENTS

supply_port_name
Specifies the name of the supply port. Hierarchical names are allowed.

-state {*state_name state_value*}

Specifies the name and value of a state of the supply port. You can repeat this option. The state value can be one of the following:

- A single floating point number that represents the nominal voltage for the specified state. For example, to define a state called s88 with a nominal voltage of 0.88, use the following syntax:

-state {s88 0.88}
- Three floating point numbers that represent the minimum, nominal, and maximum voltages of the specified state, respectively. For example, to define a state called active_state with a minimum voltage of 0.88, a nominal voltage of 0.90, and a maximum voltage of 0.92, use the following syntax:

-state {active_state 0.88 0.90 0.92}
- The keyword **off**, which indicates an inactive state. For example, to define an inactive state called off_state, use the following syntax:

-state {off_state off}

DESCRIPTION

This command adds state information to a supply port. The first occurrence of the *supply_port_name* option is the default state of the supply port. You can use this to represent off-chip supply sources that are not driven by the test bench. This option defines the voltage level supplied to the chip; it provides a convenient shortcut to facilitate verification and analysis without requiring the creation of a power

domain and a supply network within the verification environment. An error occurs if *supply_port_name* does not already exist prior to executing this command.

This command returns the fully qualified name from the current scope of the created port or a null string if the port is not created.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows the states and voltages of a supply port named VN1:

```
prompt> add_port_state VN1 \  
-state {active_state 0.88 0.90 0.92} \  
-state {off_state off}
```

SEE ALSO

```
add_pst_state(2)  
create_pst(2)  
report_pst(2)
```

add_pst_state

Defines the states of each of the supply nets for one possible state of the design.

SYNTAX

```
status add_pst_state  
state_name  
-pst table_name  
-state supply_states
```

Data Types

<i>state_name</i>	string
<i>table_name</i>	string
<i>supply_states</i>	list

ARGUMENTS

state_name
Specifies the name of the power state.

-pst table_name
Specifies the power state table (PST) to which this state applies.

-state supply_states
Lists the supply net state names in the corresponding order of the fb-supplies option listing in the **create_pst** command.

DESCRIPTION

The **add_pst_state** command defines the states of each of the supply nets for one possible state of the design. It is an error if the number of supply state names is different from the number of supply nets within the power state table.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example defines the supply net state names for the s1, s2 and s3 power states:

```
create_pst pt -supplies {PN1 PN2 SOC/OTC/PN3}  
add_pst_state s1 -pst pt -state {s08 s08 s08}  
add_pst_state s2 -pst pt -state {s08 s08 off}  
add_pst_state s3 -pst pt -state {s08 s09 off}
```

add_pst_state

SEE ALSO

`add_port_state(2)`
`create_pst(2)`
`report_pst(2)`

add_row

Creates a list of rows in the design.

SYNTAX

```
status add_row
[-in base_array_name]
[-tile_pattern tile_pattern_name]
[-tile_name tile_name]
-area {{ll_x_ll_y}_{ur_x_ur_y}}
[-minimal_channel_height channel_height]
[-direction direction]
[-flip_first_row] | [-no_double_back]
[-no_start_from_first_row]
[-no_snap_to_wire_track]
[-dont_snap_to_existing_row]
[-no_double_back]
```

Data Types

<i>base_array_name</i>	string
<i>tile_pattern_name</i>	string
<i>tile_name</i>	string
<i>channel_height</i>	float
<i>direction</i>	string

ARGUMENTS

-in *base_array_name*
Specifies the name of the base array in which to create rows. The **-in *base_array_name***, **-tile_pattern *tile_pattern_name*** and **-tile_name *tile_name*** options are mutually exclusive. By default, this option is off.

-tile_pattern *tile_pattern_name*
Specifies the name of the tile pattern to use in rows. The **-in *base_array_name***, **-tile_pattern *tile_pattern_name*** and **-tile_name *tile_name*** options are mutually exclusive. By default, this option is off.

-tile_name *tile_name*
Specifies the name of the unit tile to use in rows. The **-in *base_array_name***, **-tile_pattern *tile_pattern_name*** and **-tile_name *tile_name*** options are mutually exclusive. The default is the tile name **unit** for the rows.

-area {{*ll_x_ll_y*}_{*ur_x_ur_y*}}
Specifies the coordinates for the lower-left and upper-right corners of the area in which to create a list of rows.

-minimal_channel_height *channel_height*
Specifies the channel height amount to provide for routing between rows. By default, it is **0.0**.

-direction *direction*
 Specifies the direction of rows to be created. The valid values for *direction* are **horizontal** and **vertical**. By default, the direction of rows is the same as the direction of the base array where the rows are located.

-flip_first_row
 Specifies to flip a row at the bottom of a horizontal core area or at the left side of a vertical core area. The **-flip_first_row** and **-no_double_back** options are mutually exclusive. By default, this option is off.

-no_start_from_first_row
 Specifies that you do not want a pair of rows at the bottom of a horizontal core area or at the left side of a vertical core area. Normally, the area can contain such pairs. By default, this option is off.

-no_snap_to_wire_track
 Specifies that you do not want the bottom of the horizontal row or the left side of the vertical row to snap to the wire track. Normally, they can. By default, this option is off.

[-dont_snap_to_existing_row]
 Snaps to an existing row. Normally, row positions are not snapped. By default, this option is off.

-no_double_back
 Specifies that you do not want the area to contain pairs of rows with one row in each pair flipped. Normally, the area can contain such pairs. By default, this option is off.

DESCRIPTION

This command adds rows into a specified area. The command returns a value of **1** if it succeeds or **0** if it fails.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates rows in the specified area.

```
prompt> add_row -area {{100 100} {300 300}} -tile unit
Info: 6 rows are added.
1
```

SEE ALSO

cut_row(2)

add_tap_cell_array

Adds tap cells to the design, forming a two-dimensional array structure.

SYNTAX

```
integer add_tap_cell_array
[-master_cell_name cell_name]
[-voltage_area {voltage_areas_collection}]
[-distance tap_cell_distance]
[-pattern normal | every_other_row | stagger_every_other_row]
[-offset distance]
[-tap_cell_identifier tap_cell_prefix]
[-tap_cell_separator tap_cell_separator]
[-no_tap_cell_under_layers layer_list]
[-well_port_name port_name]
[-well_net_name net_name]
[-substrate_port_name port_name]
[-substrate_net_name substrate_tie_net_name]
[-connect_power_name power_net_name]
[-connect_ground_name ground_net_name]
[-fill_boundary_row true | false]
[-fill_macro_blockage_row true | false]
[-boundary_row_double_density true | false]
[-macro_blockage_row_double_density true | false]
[-left_macro_blockage_extra_tap by_rule | must_insert | no_insert]
[-right_macro_blockage_extra_tap by_rule | must_insert | no_insert]
[-left_boundary_extra_tap by_rule | must_insert | no_insert]
[-right_boundary_extra_tap by_rule | must_insert | no_insert]
[-ignore_soft_blockage true | false]
[-at_distance_only true | false]
[-skip_fixed_cells true | false | tap]
[-respect_keepout]
```

Data Types

<i>cell_name</i>	string
<i>voltage_areas_collection</i>	collection
<i>tap_cell_distance</i>	float
<i>tap_cell_prefix</i>	string
<i>tap_cell_separator</i>	string
<i>layer_list</i>	list
<i>port_name</i>	string
<i>substrate_tie_net_name</i>	string
<i>power_net_name</i>	string
<i>ground_net_name</i>	string

ARGUMENTS

-master_cell_name *cell_name*
Specifies the name of the library reference cell to be used as a tap cell.

add_tap_cell_array

`-voltage_area {voltage_areas_collection}`
 Specifies the voltage areas in which to insert tap fillers. Tap fillers are not inserted outside of the specified voltage areas. By default, the command inserts tap fillers in all voltage areas.

`-distance tap_cell_distance`
 Specifies the distance (unit is um) between two tap cells in a row.

`-pattern normal | every_other_row | stagger_every_other_row`
 Specifies the pattern of tap cell insertion. When you use the **normal** argument, tap cells are added to every row having the specified tap distance spacing. The tap distance should be approximately twice the design rule distance limit. When you use the **every_other_row** argument, the command places tap cells in every other row (on the odd rows only). This allows you to reduce the number of required tap cells by approximately half compared to the normal pattern. The tap distance specified in the **every_other_row** argument should be approximately twice the design rule distance limit. When you use the **stagger_every_other_row** argument, the command places tap cells in every row. The tap cells on even rows are offset with half the specified distance relative to the odd rows, producing a checkerboard-like pattern. This pattern also allows you to reduce the number of required tap cells by approximately half compared to the normal pattern. The tap distance specified in this pattern should be four times the design rule distance limit. The default is **normal**.

`-offset distance`
 Specifies the offset distance (unit is um) when you use the **-pattern** option with the **stagger_every_other_row** argument.

`-tap_cell_identifier tap_cell_prefix`
 Specifies the name prefix for inserted tap cells.

`-tap_cell_separator tap_cell_separator`
 Specifies a separator character to be used when composing the instance name of the tap cell. The tap cell instance name consists of a prefix, the library reference cell name, the tap layer number, and a rolling number. For example, the instance name "tapfiller!MY_TAP!98!25" has prefix "tapfiller," the library reference cell name "MY_TAP," the tap layer number "98," the rolling number "25," and the tap_cell_separator "!." This allows you to do pattern matching selection of the tap cells, if necessary.

`-no_tap_cell_under_layers layer_list`
 Avoids placing tap cells under specified metal preroutes.

`-well_port_name port_name`
 Specifies the port name of the well tie.

`-well_net_name net_name`
 Specifies the net name of the well tie.

`-substrate_port_name port_name`
 Specifies the port name of the substrate tie.

`-substrate_net_name substrate_tie_net_name`
 Specifies the net name of the substrate tie.

`-connect_power_name power_net_name`
 Specifies the net to connect the tap cell's power pin to, if there is more than one power net in the design,

`-connect_ground_name ground_net_name`
 Specifies the net to connect the tap cell's ground pin to, if there is more than one ground net in the design.

`-fill_boundary_row true | false`
 Specifies the density adjustment of the tap cell array. When the `every_other_row` tap placement pattern is specified in the pattern option, every other row is supposed to be empty to reduce the number of tap cells inserted. This option instructs the tool to fill the section of a row that is adjacent to the chip boundary to avoid tap rule violation. The default setting is true. When set to false, the section of the row adjacent to the chip boundary may need to rely on taps outside the boundary to satisfy the tap distance rule. This option is only effective when the `every_other_row` tap pattern is selected.

`-fill_macro_blockage_row true | false`
 Specifies the density adjustment of the tap cell array. When the `every_other_row` tap placement pattern is specified in the pattern option, every other row is supposed to be empty to reduce the number of tap cells inserted. This option instructs the tool to fill the section of a row that is adjacent to the macro/blockage boundary to avoid tap rule violation. The default setting is true. When set to false, the section of the row adjacent to the macro/blockage boundary may need to rely on taps outside the boundary to satisfy the tap distance rule. This option is only effective when the `every_other_row` tap pattern is selected.

`-boundary_row_double_density true | false`
 Specifies the density adjustment of the tap cell array. When the `stagger_every_other_row` tap placement pattern is specified in the pattern option, every other row is supposed to be staggered to reduce the number of tap cells inserted. This option instructs the tool to double the tap density on the section of a row that is adjacent to the chip boundary to avoid tap rule violation. The default setting is true. When set to false, the section of the row adjacent to the chip boundary needs to rely on taps outside the boundary to satisfy the tap distance rule. This option is only effective when the `stagger_every_other_row` tap pattern is selected.

`-macro_blockage_row_double_density true | false`
 Specifies the density adjustment of the tap cell array. When the `stagger_every_other_row` tap placement pattern is specified in the pattern option, every other row is supposed to be staggered to reduce the number of tap cells inserted. This option instructs the tool to double the tap density on the section of a row that is adjacent to the macro/blockage boundary to avoid tap rule violation. The default setting is true. When set to false, the section of the row adjacent to the macro/blockage boundary needs to rely on taps outside the boundary to satisfy the tap distance rule. This option is only effective when the `stagger_every_other_row` tap pattern is selected.

`-left_macro_blockage_extra_tap by_rule | must_insert | no_insert`
`| no_insert`
 Specifies the density adjustment of the tap cell array. The inter tap distance

is often set to twice the minimum tap rule distance. By default, the tool assumes that the minimum tap distance is half of the inter tap spacing specified in the `-distance` option. The default setting is `by_rule` where the tool may insert an extra tap cell at the location of half inter tap spacing near the right edge of a row adjacent to a macro or blockage. This extra tap is inserted if the right row edge is more than half inter tap distance from the closest tap on the same row. The default setting may or may not insert an extra tap but always ensures that the minimum tap distance rule is not violated. When such tap is not needed, the `must_insert` option forces a tap to be inserted at exactly the right edge of the row. When the default setting needs to insert a tap, the `no_insert` option forces the tool to skip this insertion. When the `no_insert` option is used, standard cells near the right side of the row may need to rely on taps inside the macro/blockage to satisfy the tap distance rule. The `must_insert` option does not always cause a tap to be placed against the right edge of the row. This only happens when the distance to the closest tap is less than half inter tap distance.

`-right_macro_blockage_extra_tap by_rule | must_insert | no_insert`
`| no_insert`

Specifies the density adjustment of the tap cell array. The inter tap distance is often set to twice the minimum tap rule distance. By default, the tool assumes that the minimum tap distance is half of the inter tap spacing specified in the `-distance` option. The default setting is `by_rule` where the tool may insert an extra tap cell at the location of half inter tap spacing near the right edge of a row adjacent to a macro or blockage. This extra tap is inserted if the left row edge is more than half inter tap distance from the closest tap on the same row. The default setting may or may not insert an extra tap but always ensures that the minimum tap distance rule is not violated. When such tap is not needed, the `must_insert` option forces a tap to be inserted at exactly the left edge of the row. When the default setting needs to insert a tap, the `no_insert` option forces the tool to skip this insertion. When the `no_insert` option is used, standard cells near the left side of the row may need to rely on taps inside the macro/blockage to satisfy the tap distance rule. The `must_insert` option does not always cause a tap to be placed against the left edge of the row. This only happens when the distance to the closest tap is less than half inter tap distance.

`-left_boundary_extra_tap by_rule | must_insert | no_insert`

Specifies the density adjustment of the tap cell array. The inter tap distance is often set to twice the minimum tap rule distance. By default, the tool assumes that the minimum tap distance is half of the inter tap spacing specified in the `-distance` option. The default setting is `by_rule` where the tool may insert an extra tap cell at the location of half inter tap spacing near the left edge of a row adjacent to the design's boundary. This extra tap is inserted if the left row edge is more than half inter tap distance from the closest tap on the same row. The default setting may or may not insert an extra tap but always ensures that the minimum tap distance rule is not violated. When such tap is not needed, the `must_insert` option forces a tap to be inserted at exactly the left edge of the row. When the default setting needs to insert a tap, the `no_insert` option forces the tool to skip this insertion. When the `no_insert` option is used, standard cells near the left side of the row may need to rely on taps outside the design's boundary to satisfy the tap distance rule. The `must_insert` option does not always cause a tap to be placed against the left edge of the row. This only happens when the distance to the closest tap is less than half inter tap distance.

`-right_boundary_extra_tap by_rule | must_insert | no_insert`

Specifies the density adjustment of the tap cell array. The inter tap distance is often set to twice the minimum tap rule distance. By default, the tool assumes that the minimum tap distance is half of the inter tap spacing specified in the `-distance` option. The default setting is `by_rule` where the tool may insert an extra tap cell at the location of half inter tap spacing near the right edge of a row adjacent to the design's boundary. This extra tap is inserted if the right row edge is more than half inter tap distance from the closest tap on the same row. The default setting may or may not insert an extra tap but always ensures that the minimum tap distance rule is not violated. When such tap is not needed, the `must_insert` option forces a tap to be inserted at exactly the right edge of the row. When the default setting needs to insert a tap, the `no_insert` option forces the tool to skip this insertion. When the `no_insert` option is used, standard cells near the right side of the row may need to rely on taps outside the design's boundary to satisfy the tap distance rule. The `must_insert` option does not always cause a tap to be placed against the right edge of the row. This only happens when the distance to the closest tap is less than half inter tap distance.

`-ignore_soft_blockage true | false`

Instructs the tool to ignore soft blockages during tap cell insertion. The default is `false`.

`-at_distance_only true | false`

Specifies the density adjustment of the tap cell array. The command uses `false` by default and inserts extra tap cells besides those inserted at distance and distance/2. When this option is `true`, the tool inserts tap cells at distance or distance/2 (or distance/4 in stagger mode) only and nowhere else. This could cause DRC violations. This is only effective while four specific options of `add_tap_cell_array` are set properly as follows:

```
-left_macro_blockage_extra_tap by_rule
-right_macro_blockage_extra_tap by_rule
-left_boundary_extra_tap by_rule
-right_boundary_extra_tap by_rule
By default, they are set to by_rule.
```

DESCRIPTION

This command adds tap cells to the design, forming a two dimensional array structure. A tap cell is a special nonlogic cell with well and/or substrate ties. Tap cells are typically used when most or all standard cell in the library contain no substrate or well taps. The design rules typically specify a distance limit from every transistor of a standard cell to a well or substrate tie.

This command is used to insert tap cells before global placement so that all standard cells subsequently placed can satisfy the distance limit because of the tap cells that have been inserted. You should properly specify the tap distance and offset based on your specific design rule's distance limit. The command has no knowledge of the design rule's distance limit. A visual check is recommended after this command to ensure that all standard cell placeable areas are properly protected by tap cells. You can select several tap cell placement patterns.

The command can connect a substrate or well port of a tap cell to a specified net.

`add_tap_cell_array`

It can also connect a power or ground port of the tap cells to a specified power or ground net.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

```
prompt> add_tap_cell_array -master_cell_name Cell11\  
-distance 30 -pattern normal\  
-voltage_area [get_voltage_areas "V*"] \  
-no_tap_cell_under_layer {M1, M2}
```

SEE ALSO

```
insert_tap_cells_by_rules(2)  
set_keepout_margin(2)
```

add_to_collection

Adds objects to a collection, resulting in a new collection. The base collection remains unchanged.

SYNTAX

```
collection add_to_collection  
base_collection  
object_spec  
[-unique]
```

Data Types

<i>base_collection</i>	collection
<i>object_spec</i>	list

ARGUMENTS

base_collection

Specifies the base collection to which objects are to be added. This collection is copied to the result collection, and objects matching *object_spec* are added to the result collection. The *base_collection* can be the empty collection (empty string), subject to some constraints as explained in the DESCRIPTION section.

object_spec

Specifies a list of named objects or collections to add.
If the base collection is heterogeneous, only collections can be added to it. If the base collection is homogeneous, the object class of each element in this list must be the same as in the base collection. If it is not the same class, it is ignored. From heterogeneous collections in the *object_spec*, only objects of the same class of the base collection are added. If the name matches an existing collection, the collection is used. Otherwise, the objects are searched for in the database using the object class of the base collection.
Some special rules apply to the *object_spec* when the base collection is empty. The rules are explained in the DESCRIPTION section.

-unique

Indicates that duplicate objects are to be removed from the resulting collection. By default, duplicate objects are not removed.

DESCRIPTION

The **add_to_collection** command allows you to add elements to a collection. The result is a new collection representing the objects in the *object_spec* added to the objects in the base collection.

Elements that exist in both the base collection and the *object_spec*, are duplicated in the resulting collection. Duplicates are not removed unless you use the **-unique** option. If the *object_spec* is empty, the result is a copy of the base collection.

add_to_collection

If the base collection is homogeneous, the command searches in the database for any elements of the *object_spec* that are not collections, using the object class of the base collection. If the base collection is heterogeneous, all implicit elements of the *object_spec* are ignored.

When the *base_collection* argument is the empty collection, some special rules apply to the *object_spec*. If the *object_spec* is not empty, there must be at least one homogeneous collection somewhere in the *object_spec* list, but its position in the list does not matter. The first homogeneous collection in the *object_spec* list becomes the base collection and sets the object class for the function. The examples show the different errors and warnings that can be generated.

For background on collections and querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example from PrimeTime (using the **get_ports** command) gets all ports beginning with mode, then adds the CLOCK port:

```
prompt> set xports [get_ports mode*]
{"mode[0]", "mode[1]", "mode[2]"}

prompt> add_to_collection $xports [get_ports CLOCK]
{"mode[0]", "mode[1]", "mode[2]", "CLOCK"}
```

The following example from PrimeTime adds the cell u1 to a collection containing the SCANOUT port:

```
prompt> set sp [get_ports SCANOUT]
{"SCANOUT"}

prompt> set het [get_cells u1]
{"u1"}

prompt> query_objects -verbose [add_to_collection $sp $het]
{"port:SCANOUT", "cell:u1"}
```

The following examples show how **add_to_collection** behaves when the base collection is empty. Adding two empty collections yields the empty collection. Adding an implicit list of only strings or heterogeneous collections to the empty collection generates an error message, because no homogeneous collections are present in the *object_spec* list. Finally, as long as one homogeneous collection is present in the *object_spec* list, the command succeeds, even though a warning message is generated. The example uses the variable settings from the previous example.

```
prompt> sizeof_collection [add_to_collection "" ""]
0
```

```
prompt> set A [add_to_collection "" [list a $het c]]
Error: At least one homogeneous collection required for argument 'object_spec'
to add_to_collection when the 'collection' argument is empty (SEL-014)
```

```
prompt> add_to_collection "" [list a $het $sp]
Warning: Ignored all implicit elements in argument 'object_spec'
to add_to_collection because the class of the base collection
could not be determined (SEL-015)
{"SCANOUT", "u1", "SCANOUT"}
```

In the case where the base collection is empty and it is intended to view the results of add_to_collection in the base collection, the following can be done:

```
prompt> set col {}
prompt> foreach_in_collection i [get_designs *] {
    current_design $i
    set col [add_to_collection $col [get_cells *data_reg*]]
}
```

```
prompt> query_objects $col
{data_reg1 data_reg2}
```

SEE ALSO

```
collections(2)
query_objects(2)
remove_from_collection(2)
sizeof_collection(2)
```


add_to_rp_group

Adds a cell, hierarchical group, or keepout to an existing relative placement group.

SYNTAX for Leaf Cells

status **add_to_rp_group**

```
rp_groups
-leaf cell_name
[-column integer]
[-row integer]
[-pin_align_name pin_name]
[-orientation direction]
[-alignment bottom-left | bottom-right]
[-num_rows integer]
[-num_columns integer]
```

Data Types for Leaf Cells

<i>rp_groups</i>	list or collection
<i>cell_name</i>	cell
<i>pin_name</i>	string
<i>direction</i>	list of strings

SYNTAX for Hierarchical Groups

status **add_to_rp_group**

```
rp_groups
-hierarchy group_name
[-instance instance_name]
[-column integer]
[-row integer]
[-alignment bottom-left | bottom-right]
```

Data Types for Hierarchical Groups

<i>group_name</i>	list or collection
<i>instance_name</i>	cell

SYNTAX for Keepouts

status **add_to_rp_group**

```
rp_groups
-keepout keepout_name
[-column integer]
[-row integer]
[-width integer]
[-height integer]
```

[-type hard | soft | space]

Data Types for Keepouts

keepout_name string

ARGUMENTS

rp_groups

Specifies the relative placement groups in which to add an item. The groups must all be in the same design.

-leaf *cell_name*

Specifies the name of a cell to add to the relative placement groups in *rp_groups*. The specified cell must be in the design that contains the relative placement groups in *rp_groups*.

-column *integer*

Specifies the column position in which to add the item. If you do not specify the column position, it defaults to zero.

-row *integer*

Specifies the row position in which to add the item. If you do not specify the row position, it defaults to zero.

-pin_align_name *pin_name*

Specifies the name of the pin to use for pin alignment of this cell with other cells in a group. This overrides the default pin name specified for the relative placement group into which it is being added. This option can only be used when adding a leaf cell with the **-leaf** option.

-orientation *direction*

Specifies the placement orientation of the cell being added. Specify the orientation with respect to the group into which the cell is being added. You can specify the list of possible orientations from which the tool chooses the first legal orientation for the cell. You can specify the direction using DEF syntax:

DEF syntax:

N, W, S, E, FN, FW, FS, FE

This option can only be used when adding a leaf cell with the **-leaf** option.

-alignment bottom-left | bottom-right

Specifies the alignment to use when placing the cell or group with respect to its parent group.

If you do not specify this option, IC Compiler uses the alignment method defined for the parent group. If a relative placement group has **-compress** set, and you add an element in that group with **-alignment bottom-right**, then the alignment type of that element is ignored.

-num_rows *integer*

Specifies the number of rows to which you add the leaf cell. You can specify multiple rows, but the default is one if you do not set this option.

add_to_rp_group

-num_columns *integer*
 Specifies the number of columns to which you add the leaf cell. You can specify multiple columns, but the default is one if you do not set this option. You can not use the bottom-right and bottom-pin alignment settings for a leaf cell occupying multiple columns.

-hierarchy *group_name*
 Specifies the relative placement group to be added hierarchically to the relative placement groups in *rp_groups*.

-instance *instance_name*
 Specifies the hierarchical cell in which to instantiate the relative placement group specified in **-hierarchy**. The cell must be an instance of the reference design that contains the relative placement group specified in **-hierarchy** and must be in the design containing the relative placement groups specified in *rp_groups*. This option can only be used when adding a relative placement group with the **-hierarchy** option.

-keepout *keepout_name*
 Specifies the name of the keepout to be added to the relative placement groups in *rp_groups*. Although the keepout is not a design object, you provide a name so you can refer to the keepout after it is created.

-width *integer*
 Specifies the width of the keepout being added. If you do not specify the width, the keepout defaults to the width of the widest cell in the column into which the keepout is being added. This option can only be used when adding a keepout with the **-keepout** option.

-height *integer*
 Specifies the height of the keepout being added. If you do not specify the height, the keepout defaults to the height of the tallest cell in the column into which the keepout is being added. This option can only be used when adding a keepout with the **-keepout** option.

-type *hard | soft | space*
 Specifies whether a keepout is hard or soft or space. A hard keepout keeps everything out of a location during legalization stage. A soft keepout allows non-rp cells to come into its location during legalization stage. A space keepout allows non-rp cells to come into its location during placement. By default a keepout is hard.

-soft

DESCRIPTION

This command adds an item to the specified relative placement groups. The relative placement groups must have been previously defined by using the **create_rp_group** command. The item can be either a leaf cell, a relative placement group, or a keepout. The item is placed in a particular lattice position of the group as specified by a row and column.

Relative placement usage is available with Design Compiler Ultra.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses **add_to_rp_group** to add a cell to an existing relative placement group and then adds that group hierarchically to another existing group.

```
prompt> get_rp_groups ripple::grp_ripple
{ripple::grp_ripple}

prompt> add_to_rp_group ripple::grp_ripple -leaf carry_in_1
{ripple::grp_ripple}

prompt> add_to_rp_group example3::top_group -hier grp_ripple -instance U2
{example3::top_group}
```

SEE ALSO

```
create_rp_group(2)
remove_rp_groups(2)
write_rp_groups(2)
```

adjust_fp_floorplan

Adjusts an existing floorplan.

SYNTAX

```
status adjust_fp_floorplan
[-core_aspect_ratio real]
[-core_utilization real]
[-core_width real]
[-core_height real]
[-number_rows int]
[-die_width real]
[-die_height real]
[-use_vertical_row sbool]
[-no_double_back sbool]
[-start_first_row sbool]
[-flip_first_row sbool]
[-row_core_ratio real]
[-left_io2core real]
[-right_io2core real]
[-bottom_io2core real]
[-top_io2core real]
[-min_pad_height sbool]
[-maintain_placement]
[-remove_filler_io]
[-die_origin point]
[-sm_utilization string_list]
[{-fc_periphery string | -fc_in_core string}]
```

ARGUMENTS

- core_aspect_ratio *real***
Specify the required Core Aspect Ratio (Core Height/Core Width).
If specified then the core will be resized to achieve this aspect ratio while maintaining the current placement area.
- core_utilization *real***
Specify the required Core Utilization (Core Physical Area/ Core Placement Area).
If specified then the core will be resized to achieve this new placement area which maintaining the current aspect ratio.
- core_width *real***
Specify the required Core Width in microns.
If specified then the core will be resized to achieve this new width while maintaining the current core height.
This option is usually combined with **-core_height** or **-number_rows** (for horizontal rows).
- core_height *real***
Specify the required Core Height in microns.
If specified then the core will be resized to achieve this new height while

maintaining the current core width.
This option is usually combined with `-core_width` or `-number_rows` (for vertical rows).

`-number_rows` *int*

Specify the required Number of Rows.
If specified then the core will be resized to achieve this new number of rows.
For horizontal rows the current core width will be maintained, for vertical rows the current core height will be maintained.

`-die_width` *real*

Specify the Die Width in microns.
If specified then the die will be resized to achieve this new width while maintaining the current die height.
This option is usually combined with `-die_height`.

`-die_height` *real*

Specify the Die Height in microns.
If specified then the die will be resized to achieve this new height while maintaining the current die width.
This option is usually combined with `-die_width`.

`-use_vertical_row` *sbool*

Specify cell rows are placed vertically

`-no_double_back` *sbool*

Specify cell rows are not double backed.
Double backed rows only have gaps between each row pair (flipped and non-flipped row). Non-double backed rows have gaps between each row.

`-start_first_row` *sbool*

Specify cell rows are started with a complete pair (flipped and non-flipped)

`-flip_first_row` *sbool*

Specify first cell row is flipped.

`-row_core_ratio` *real*

Specify row core ratio, which is the ratio of row height and core height (for horizontal rows) or row width and core width (for vertical rows).

`-left_ioore` *real*

Specify distance in microns from core left edge to right edge of the minimum or maximum left pad (depending on the min pad height setting).

`-right_ioore` *real*

Specify distance in microns from core right edge to left edge of the minimum or maximum right pad (depending on the min pad height setting).

`-bottom_ioore` *real*

Specify distance in microns from core bottom edge to top edge of the minimum or maximum bottom pad (depending on the min pad height setting).

`-top_ioore` *real*

Specify distance in microns from core top edge to bottom edge of the minimum or maximum top pad (depending on the min pad height setting).

`adjust_fp_floorplan`

`-min_pad_height sbool`
 Specify whether core to pad distances are to the smallest pad on the corresponding side instead of the default largest pad.

`-maintain_placement`
 Keep current standard cell and hard macro placement inside the core instead of by default invalidating the placement by moving standard cells and hard macros outside the core.

`-remove_filler_io`
 Remove any filler io pads.

`-die_origin point`
 Set the die origin. This is usually at (0, 0) but can be placed anywhere.

`-sm_utilization string_list`
 Specify individual soft macro utilizations as a list of strings. Each string in the list is of the format:
 `sm_name sm_utilization [fc_area_factor]`
 the optional flip chip pad/driver area factor can only be supplied if the design has flip chip pads/drivers.

`-fc_periphery string`
 Number of flip chip pads or drivers in the periphery (outside core area). If all flip chip pads or drivers are in the periphery the string "all" can be used.
 Any flip chip pads or drivers not in the periphery are assumed to be in the core and will affect utilization calculations.
Note: cannot be combined with `-fc_in_core`.

`-fc_in_core string`
 Number of flip chip pads or drivers inside the core. If all flip chip pads or drivers are inside the core the string "all" can be used.
 All flip chip pads in the core and will affect utilization calculations.
Note: cannot be combined with `-fc_periphery`.

DESCRIPTION

This command allows the user to modify the current die/core size by changing one or more values while maintaining as much of the current settings as possible.

The user can change the die/core size by using one of the following:

- . absolute die size (`-die_width` and/or `-die_height`)
- . absolute core size (`-core_width` and/or `-core_height`)
- . core size based on number of rows (`-number_rows` and/or `-core_width` or `-core_height` depending on row direction)
- . core utilization (`-core_utilization` and/or `-core_aspect_ratio`)

The user can also additionally change one or more of the following:

- . row geometry (direction, double_back, start_first_row, flip_first_row, row_core_ratio)
- . core to pads distances

- . die origin
- . specify core to pad distance is to the minimum pad height
(default is maximum pad height)

Note: For all core resizes the die is resized to maintain the core to pad spacing. For all die resizes the core is resized to maintain the core to pad spacing.

By default changing the dir/core invalidates placement so all standard cells and macros are unplaced (moved out of the core area). If the user wants to retain their current placement then the `-keep_placement` option should be used.

Also after placement blockages and filler pads may be added which will be invalid if the core is resized. Blockages are automatically removed, filler pads can be optionally removed using `-remove_filler_io`.

When sizing by utilization normally just the total physical area of all standard cells and hard macros will be used to calculate the placement area. If the design has soft macros or has flip chip pads/drivers then the utilization calculation can be adjusted to take these into account.

For soft macros the estimated utilization of each soft macro can be specified. This effectively increases the physical size of the soft macro from just the total physical size of all its child standard cells and hard macros to the physical size divided by the utilization e.g. a utilization of 0.5 will double the physical area of the soft macro.

For flip chip drivers and pads there will be a number that are in the core area and a number that are in the periphery (outside the core area). Those flip chip drivers and pads in the core will decrease the placement area so must be taken into account in the utilization calculation.

The user can specify this number by either the in-core number or the peripheral number. As a convenience this number can be the string "all". By default all flip chip drivers/pads are considered to be in the core.

Also with soft macros and flip chip drivers and pads, a number of flip flip chip pads/drivers will overlap each soft macro. This number is automatically estimated for each soft macro based on soft macro size and number of in-core flip chip drivers and pads. In addition to this an extra scale factor can be applied to the resultant area to fine tune this.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

Resize core to 1000 x 1000 microns, maintaining the core to pad spacing

```
prompt> adjust_fp_floorplan -core_width 1000 -core_height 1000
```

Change rows to vertical with row core ratio of 0.9

```
adjust_fp_floorplan
```



```
prompt> adjust_fp_floorplan -use_vertical_row true -row_core_ratio 0.9
```

Change core utilization to 0.9 with soft macro alu1 utilization at 0.5

```
prompt> adjust_fp_floorplan -core_util 0.9 -sm_utilization {"alu1 0.5"}
```

SEE ALSO

adjust_fp_io_placement

Adjusts I/O pad placement.

SYNTAX

```
int adjust_fp_io_placement  
[-side l | r | t | b]  
[-spacing float]  
[-pitch float]  
[-offset float]  
[-undo]  
[list of IO cells]
```

ARGUMENTS

- side l | r | t | b
Indicates on which side of the chip are the IO cells to be adjusted. This is mutually exclusive with giving a list of IO cells.
- spacing *float*
Specifies how much spacing in microns between adjacent IOs. This option is mutually exclusive with -pitch.
- pitch *float*
Specifies the distance from the center of an IO cell to the center of the adjacent IO cell in microns. This option is mutually exclusive with -spacing.
- offset *float*
The distance (in microns) between the lower-most (for left and right IOs) or the left-most (for top and bottom IOs) point of all the adjusted IO cells and the bottom edge or left edge of the chip. If no -offset is specified the IO cells will be packed in the center of the appropriate side.
- undo
Undo the movements made in the previous `adjust_fp_io_placement` call. This option is mutually exclusive with all other options. Only one level of undo is supported.
- list of IO cells*
The IO cells whose placement are to be adjusted. All the selected IOs must be on the same side of the chip. This is mutually exclusive with the -side option.

DESCRIPTION

Allows user to adjust the spacing and location of IO cells. The `adjust_fp_io_placement` command only moves cells with the IO pad type. The IO cells that are moved are marked as FIXED. Any IO-to-IO cell overlaps after the adjustment is done are reported.

adjust_premesh_connection

Connects the load pins in the specified clock tree or subtree either as mesh loads or as part of the premesh tree. If you specify the `-premesh` option, it balances the first-level loads of the premesh tree.

SYNTAX

```
status adjust_premesh_connection  
-root root_name  
[-premesh | -exclude_pins list_of_pins]  
[-operating_condition min | max | min_max]
```

Data Types

<i>root_name</i>	net, port, or pin
<i>list_of_pins</i>	list

ARGUMENTS

`-root root_name`
Specifies the name of the root of the clock tree or subtree. You can specify a clock net, clock port, or pin that drives a clock tree or subtree. If you specify a clock net, the command considers all the load pins of that net. If you specify a pin or a port, the command considers the load pins in the fanout of the specified port or pin. A pin is a load pin if it is either a macro pin or has a nonzero phase delay. This is a required option.

`-premesh`
Balances the first-level load pins of the specified clock tree or subtree with the isolation buffers that drive the load pins below the mesh. Use this option only after running the `analyze_subcircuit` command. This option and the **-exclude_pins** option are mutually exclusive.

`-exclude_pins list_of_pins`
Excludes the specified pins from consideration. The specified pins are connected above the mesh. By default, all load pins in the fanout of the specified clock root are considered. This option and the **-premesh** option are mutually exclusive.

`-operating_condition min | max | min_max`
Specifies the operating condition to be used by the clock tree synthesis engine. If you do not specify this option, the maximum operating condition is used.

DESCRIPTION

This command either adjusts the specified load pins (default behavior) or balances the first-level loads of the specified clock tree or subtree (**-premesh** option).

In default mode, the set of load pins is determined by removing the pins specified

in the **-exclude_pins** option from the set of pins derived from the **-root** option. The command connects these load pins either below the mesh or to the premesh tree. If the load pin is a flip-flop clock pin, drives a flip-flop clock pin, or is a macro pin with a large capacitance, the pin is connected below the mesh. If the load pin has a nonzero phase delay, the phase delay is compared to the maximum delay of all branches associated with flip-flops. If the phase delay is greater than the maximum branch delay, the pin is connected to the premesh tree.

If the clock tree has a macro pin with nonzero phase delay and **split_clock_net - isolate_float_pins** is used. The command decides whether to connect the pin to the premesh tree or below the mesh based on the capacitance value and phase delay of the pin.

If the macro pin has high capacitance value, the pin is connected below the mesh because the mesh has high drive strength. If the pin has nominal capacitance value and nonzero phase delay, the pin is connected to the premesh tree. If you need to set the phase delay of the macro pin, use the **set_clock_tree_exceptions** command.

When connecting load pins after the mesh, the command inserts an isolation buffer after the specified clock root and inserts the load pins after the isolation buffer, so that the clock root drives the isolation buffer and the isolation buffer drives the load pins below the mesh.

When connecting load pins to the premesh tree, the command connects the pins to the specified clock root.

In premesh mode, the command uses the **compile_clock_tree** and **optimize_clock_tree** commands to balance the first-level load pins of the specified clock tree.

EXAMPLE

The following command analyzes the load pins connected to the clknet1 net and either connects them to the premesh tree or below the mesh.

```
prompt> adjust_premesh_connection -root clknet1
```

SEE ALSO

`optimize_clock_tree(2)`
`split_clock_net(2)`
`analyze_subcircuit(2)`

alias

Creates a pseudo-command which expands to one or more words, or lists current alias definitions.

SYNTAX

```
string alias  
[name] [def]
```

Data Types

<i>name</i>	string
<i>def</i>	string

ARGUMENTS

name

Provides a name of the alias to define or display. The name must begin with a letter, and can contain letters, underscores, and numbers.

def

Expansion of the alias. That is, the replacement text for the alias name.

DESCRIPTION

The **alias** command defines or displays command aliases. With no arguments, the **alias** command displays all currently defined aliases and their expansions. With a single argument, the **alias** command displays the expansion for the given alias name. With more than one argument, an alias is created that is named by the first argument and expanding to the remaining arguments.

You cannot create an alias using the name of any existing command or procedure. Thus, you cannot use **alias** to redefine existing commands.

Aliases can refer to other aliases.

Aliases are only expanded when they are the first word in a command.

EXAMPLES

Although commands can be abbreviated, sometimes there is a conflict with another command. The following example shows you can use **alias** to get around the conflict.

```
prompt> alias q quit
```

The following example shows you can also use **alias** to create a shortcut for commonly used command invocations.

```
prompt> alias include {source -echo -verbose}
prompt> alias rt100 {report_timing -max_paths 100}
```

After the previous commands, the command **include script.tcl** is replaced with **source -echo -verbose script.tcl** before the command is interpreted.

The following examples show how to display aliases using **alias**. Note when displaying all aliases, they are in alphabetical order.

```
prompt> alias rt100
rt100      report_timing -max_paths 100
prompt> alias
include    source -echo -verbose
q          quit
rt100      report_timing -max_paths 100
```

SEE ALSO

unalias(2)

align_fp_pins

Physically aligns a set of soft macros (or black box) pins with a set of reference pin objects. These can be terminals or pins on hard macros, IO pad cells, standard cells, internal child hard macros, or another soft macro (or black box).

SYNTAX

```
status align_fp_pins
[-reference object]
[-direction {center | left | right | top | bottom}]
[-fixed]
[-change_layer_width]
[-order_type {low_to_high | high_to_low | net_connection}]
[-align_with_child_hm_pins]
objects
```

ARGUMENTS

-reference *object*

reference object

Object specified here is used to resolve ambiguity in cases where the command is unable to determine the reference pin. For example, a specified net connected to a soft macro pin to be aligned is connected to more than one hard macro. In this case the hard macro whose pin is to be used as a reference can be used as the argument to this option. This option can be omitted if the reference macro can be uniquely determined from the arguments supplied without resorting to the multi-level reference pin selection process (see `net_connection` under `-order_type` for details). For example, if the macro pin collection given contains pins belonging to the same hard macro, then that hard macro is used as the reference macro. This option may need to be used for alignment involving two soft macros. Note that the command uses the object provided by this option only as a source of information of last resort. Information provided by this option is silently ignored if the command can uniquely determine the reference pin (or terminal), whether or not the argument given is relevant or supplied in error. The argument can be any one of soft macro, black box, hard macro, IO pad cell, terminal, or standard cell.

-direction {center | left | right | top | bottom}

direction for alignment

"center" refers to the horizontal or vertical central axis passing through the bounding box of the reference pin object. If the soft macro (or black box) pin to be aligned sits on a horizontal edge, the center of the aligned pin is aligned to the aforementioned vertical central axis. The horizontal central axis is used instead if the aligned pin sits on a vertical edge. "center" is the default value if this option is not used.

If the reference pin object is a terminal, or a pin on an IO cell or a standard

cell, a central axis (horizontal or vertical) of the reference pin object is used regardless of the value the user may have provided.

"left", "right", "top", and "bottom" refer to the reference pin edges to be used as the line of alignment. For example if the reference pin sits on the bottom edge of a hard macro and "-direction left" is specified, the left edge of the soft macro (or black box) pin to be aligned will be made flush with

the left edge of the reference pin.

Visually speaking using "-direction left" or "-direction right" only makes a difference in this case if the reference pin and the aligned pin have different widths. Furthermore, a central axis (horizontal or vertical) of the reference pin is used if the -direction value is incompatible with the location of the aligned pin. Using the same example cited so far, the center of the aligned pin will be aligned with the vertical central axis of the reference pin if either "-direction top" or "-direction bottom" is supplied.

`-fixed`

pins become fixed after alignment

By default pins are marked "movable" after alignment.

`-change_layer_width`

Change pin metal layer and width to match reference pin

The final width of the aligned pin is the default width of the target layer, not the width of the reference pin. By default the aligned pin retains its original layer and its width is not changed.

NOTE: In future releases the soft macro pin layer and width will be automatically changed to match the layer of the reference pins. -

change_layer_width will be made hidden, and a new option will be introduced so that the user can retain current pin layer after the alignment is done.

`-order_type {low_to_high | high_to_low | net_connection}`

methods of aligning with reference pins

There are 3 values that can be specified: net_connection (default), low_to_high, and high_to_low.

net_connection: An aligned pin is moved into alignment with the reference pin that is logically connected to it by a net. When in this mode the command uses net connectivity to automatically determine the reference pins and in many cases this can save the user from having to explicitly specify the reference pins in the argument collection. For example, if the soft macro pin to be aligned is connected to a net whose only other connection is to another soft macro pin, then the latter soft macro pin is assumed to be the reference pin if the user provides the first (but not the second) soft macro pin in the argument collection. In fact the command works harder than that. If reference pin object can not be uniquely determined because the net connected to the pin to be aligned is connected to multiple other objects, a multi-level priority scheme is used to automatically determine the reference pin object: 1. Non-soft macro pins in the argument list 2. Pins on the reference object specified with -reference 3. Fixed soft macro pins 4. Other soft macro or black box pins 5. Hard macro pins 6. IO pad cell pins 7. Terminals 8. Other standard cell pins

Preference is given to the closest pin object when multiple pin objects exist within the same level. Suppose the net is not connected to any other soft macro or black box but is connected to two hard macro pins and an IO pad pin. Then the reference pin is assumed to be the hard macro pin closer to the pin to be aligned. However, the IO pad pin is used as the reference if the user explicitly includes the IO pad pin in the argument collection or specifies the IO pad cell as the argument for -reference.

low_to_high: For example if alignment is done on a horizontal edge and there are 4 reference pins and 3 pins on the other macro to be aligned.

With this option value the pins are aligned in pairs starting from pins having the lowest original horizontal coordinate. The 4th reference pin (i.e. the reference pin with the largest horizontal coordinate) is not used in the

alignment. Pins on vertical edges are operated according to their vertical coordinates.

`high_to_low`: This is the opposite of `low_to_high`. Using the same example above the first reference pin will not be used in the alignment.

There's no difference between `low_to_high` and `high_to_low` if the number of reference and aligned pins are the same. Note the original pin orders of the aligned pins are preserved for both `low_to_high` and `high_to_low` -- the aligned pin with the lowest original coordinate still has the lowest coordinate among the aligned pins.

NOTE: This option is not effective if `-align_with_child_hm_pins` is used, or if the reference pin object is a terminal or a pin of an IO cell or standard cell. In these cases `net_connection` is the sole alignment method.

NOTE: In future releases `net_connection` will no longer be explicitly available. All alignment will be assumed to be aligned by net connection unless `high_to_low` or `low_to_high` is specified. This option will also be given a more appropriate name.

`-align_with_child_hm_pins`

Align with child HM pins

The reference pin are hard macro pins inside a soft macro.

`objects`

collection of pins, nets, or reference terminals (if needed)

This collection can minimally consist of the set of soft macro (or black box) pins to be aligned. Reference pins (or terminals) can be added if they are needed to resolve any ambiguity regarding designation of reference pins. In `-order_type net_connection` mode, which is the default, the command tries it best to automatically determine the reference pins via net connection. See the section for `-order_type` for details. As such the user needs to supply reference pins (or terminals) as arguments only when absolutely necessary, or when `low_to_high` or `high_to_low` is used as value for `-order_type`, since net connection is not used as the basis for alignment and as such reference pins can not be determined from net connectivities.

Net arguments are also acceptable but they can not be mixed with pins and/or reference terminals: the collection should be a homogeneous set of nets. Also note that any terminal supplied is used as a reference. This command does NOT align terminals. They are considered fixed.

DESCRIPTION

This command physically aligns a set of unfixed soft macros pins with a set of reference terminals or pins on soft macros, hard macros, IO pad cells, or standard cells. Fixed soft macro pins will not be aligned.

Aligned pins are automatically snapped to the closest current snap type. Use `get_object_snap_type -type port_shape` to find out current snap type. The default snap type is wiretrack. For this reason pins may not be perfectly aligned if the reference pins and aligned pins are of different sizes, or if they are on different layers and the wiretrack grids of these layers are not aligned. If perfect alignment is desired current snap type can be set to min grid: `set_object_snap_type -type port_shape -snap litho` Perfect alignment is impossible if the reference pins are not snapped to min grid.

NOTE: Alignment will still go ahead if even the aligned pin shorts with another pin

after the alignment, although a warning will be issued. Furthermore, no other pin position legality check is performed.

Please also note that aligning terminals and off-edge soft macro pins are not supported.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

```
align_fp_pins -reference mac1 [get_pins "mac1/A mac1/B mac1/C mac2/A mac2/B mac2/C"]
```

Align specified soft macro pins with "mac1" as the reference macro.

SEE ALSO

```
set_object_snap_type(2)  
get_object_snap_type(2)
```

align_objects

Aligns one or more objects.

SYNTAX

```
status align_objects
[{-anchor object | -parent |
-to value_point_rect}]
[-side alignment_side]
[-offset float]
[-resize]
[-keep_area]
[-ignore_fixed]
objects
```

Data Types

<i>alignment_side</i>	string
<i>objects</i>	collection

ARGUMENTS

- anchor *object***
Specifies the object to which other objects are aligned.
- parent**
Specifies that objects are aligned to the parent edge.
- to *value_point_rect***
Specifies that objects are aligned to the position specified by *value_point_rect*.
This position can be a single value (x for left/right, y for top/bottom), a point, or a rectangle.
- side *alignment_side***
Specifies the side of the object and anchor object to be aligned.
The valid value is one of **left**, **right**, **top**, **bottom**, **hcenter**, and **vcenter**.
The description of the valid values are as follows:
left - indicates to use the left side
right - indicates to use the right side
top - indicates to use the top side
bottom - indicates to use the bottom side
hcenter - indicates to use the horizontal center
vcenter - indicates to use the vertical center
The default depends on the aspect ratio (height/width) of the object's surrounding bbox.
If the aspect ratio is less than or equal to 1, then **left** is the default, otherwise, **bottom** is the default.
- offset *float***
Specifies the offset from the specified anchor object's side.
The default value is **0.0**.

`-resize`
Specifies to resize the object when aligning.

`-keep_area`
Specifies to keep the original object's area when resizing.
By default, the command does not keep the object's area.

`-ignore_fixed`
Specifies to align fixed objects as well as unfixed objects.
By default, the command does not align fixed objects.

`objects`
Specifies a list of objects to align.

DESCRIPTION

This command aligns a list of unfixed objects so that their left, right, top, or bottom edges are coincident with, or at a specified offset from, an anchor object or anchor position.

The user can specify an anchor object or anchor position, or allow the command to automatically determine the anchor object from the supplied list of objects by using the following algorithm:

- If any objects are marked as fixed then the anchor object is selected from the set of fixed objects. The rightmost, leftmost, bottommost, or topmost fixed object for alignment of left, right, top, and bottom, respectively.
- If no objects are marked as fixed, then the anchor object is taken as the leftmost, rightmost, topmost, or bottommost object for alignment of left, right, top, and bottom, respectively.

When aligning an object, the position of the opposite edge can be retained so that the object is resized rather than moved.

The position of the other sides of the object is also retained unless you specify that the area is to be kept, in which case the other sides are moved to achieve the original object area.

When you do not specify the **-resize** option, any movable object can be aligned. If you do specify it, only resizable objects can be aligned.

See the man page for the **get_edit_property** command for details on which objects you can move and resize.

The tool automatically performs snapping of the origin, bounding box, or points using global snap settings unless you specify the **-no_snap** option.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

`align_objects`

EXAMPLES

The following example moves the currently selected objects so that their left edges are 20 units from the left edge of the core.

```
prompt> align_objects [get_selection] -parent -offset 20
```

SEE ALSO

```
distribute_objects(2)  
expand_objects(2)  
flip_objects(2)  
get_object_snap_type(2)  
move_objects(2)  
remove_objects(2)  
resize_objects(2)  
rotate_objects(2)  
set_object_snap_type(2)
```

all_active_scenarios

Lists the active scenarios available in memory.

SYNTAX

string **all_active_scenarios**

ARGUMENTS

None.

DESCRIPTION

This command displays all active scenarios currently in memory. This list excludes scenarios that are inactive.

Multicorner-Multimode Support

This command uses information from active scenarios only.

EXAMPLES

The following example uses **all_active_scenarios** to list the active scenarios.

```
prompt> create_scenario MODE1
prompt> create_scenario MODE2
prompt> create_scenario MODE3
prompt> set_active_scenarios {MODE1 MODE3}
prompt> all_active_scenarios
MODE1 MODE3
```

SEE ALSO

```
all_scenarios(2)
create_scenario(2)
current_scenario(2)
remove_scenario(2)
set_active_scenarios(2)
```

all_ao_cellsfp

Returns a collection of always-on cells available in the design.

SYNTAX

collection **all_ao_cellsfp**

DESCRIPTION

The **all_ao_cells** command returns a collection of existing always-on cells in the design. If the design has dual-powered marked always-on cells, they are returned by this command. However, if the design has always-on power guides that contain regular cells, those cells are also returned by this command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

SEE ALSO

`get_always_on_logic(2)`

all_bounds_of_cell

Returns the collection of all bounds in the specified list of cells from the design.

SYNTAX

```
collection all_bounds_of_cell  
cell_list
```

Data Types

```
cell_list      list
```

ARGUMENTS

```
cell_list
```

Specifies the list of all cells for which to determine the bounds. This can be a simple list or it can be an entire collection manipulation command, enclosed in square brackets [], that returns a collection of cells.

DESCRIPTION

This command returns the collection of all the group or move bounds of a specified cell list. You can use wild cards, such as an asterisk (*) or question mark (?), when specifying the input cell list.

A standard Tcl collection is returned. All of the generic collection manipulating commands, such as **foreach_in_collection**, **sizeof_collection**, and so on, can be performed on this collection.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows the returned collection of all bounds of cells named *cell11*, *cell12*, and *cell13*.

```
prompt> all_bounds_of_cell {cell11 cell12 cell13}  
{cell_bound1 cell_bound2 cell_bound3}
```

The following example shows the returned collection of all bounds of cells in the *u** cell list. In this case, the command returns the collection specified by the **get_cells** command operating on *u**.

```
prompt> all_bounds_of_cell [get_cells u*]  
{my_bound1 my_bound2}
```

Following example shows the returned collection of all the bounds of cells in the

SBLK subdesign. In this case, the command returns the collection specified by the **get_cells** command operating on *SBLK*.

```
prompt> all_bounds_of_cell [get_cells SBLK/*]  
{bound1 bound2}
```

SEE ALSO

```
create_bounds(2)  
foreach_in_collection(2)  
get_cells(2)  
remove_bounds(2)  
report_bounds(2)  
sizeof_collection(2)
```

all_cells_in_bound

Returns the collection of all cells in the specified list of group bounds and move bounds from the design.

SYNTAX

```
collection all_cells_in_bound
{bound_list}
```

Data Types

bound_list list

ARGUMENTS

```
{bound_list}
    Specifies all the lists of bounds for which to determine cells. The tool
    returns the collection of the bounds.
```

DESCRIPTION

This command returns the collection of all the cells from the specified group and move bounds. These bounds are created by using the **create_bound** command. This command inputs a list of multiple bounds and returns a collection of cells from all of them. The tool searches the entire design hierarchy. It returns cells from the entire design, not just from the top level.

A standard Tcl collection is returned. All of the generic collection manipulating commands, such as **foreach_in_collection**, **sizeof_collection**, and so on, can be performed on this collection.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows the returned collection of all cells in the *my_bound1* bound, which was created by using the **create_bound** command.

```
prompt> all_cells_in_bound {my_bound1}
{cell11 top/cell12 top/cell14 cell110}
```

The following example shows the returned collection of all cells from multiple bounds given as a list. The list is composed of **my_bound1** and **my_bound2**.

```
prompt> all_cells_in_bound {my_bound1 my_bound2}
{cell11 cell12 cell13 cell14 cell15}
```

SEE ALSO

`all_bounds_of_cell(2)`
`create_bounds(2)`
`foreach_in_collection(2)`
`remove_bounds(2)`
`report_bounds(2)`
`sizeof_collection(2)`

all_clocks

Returns a collection of all clocks in the current design.

SYNTAX

collection **all_clocks**

ARGUMENTS

This command has no arguments.

DESCRIPTION

Returns a collection containing all clocks in the current design. The clocks must be defined in the design before running this command. To create clocks, use the **create_clock** command. To remove clocks, use **remove_clock**. To list detailed information about all clocks in the design, use **report_clock**.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example applies the **set_dont_touch_network** command to all clocks in the current design.

```
prompt> set_dont_touch_network [all_clocks]
```

SEE ALSO

```
create_clock(2)  
remove_clock(2)  
report_clock(2)  
set_dont_touch_network(2)  
current_design(3)
```

all_connected

Returns the objects connected to a net, port, pin, net instance, or pin instance.

SYNTAX

```
collection all_connected  
[-leaf] object
```

Data Types

object string

ARGUMENTS

-leaf

Specifies that only leaf pins are returned for a hierarchical net. For nonhierarchical nets, there is no difference in output.

object

Specifies the object whose connections are returned. The object must be a net, port, pin, net instance, or pin instance.

DESCRIPTION

Returns a collection of objects connected to the specified net, port, pin, net instance, or pin instance. A net instance is a net in the hierarchy of the design. A pin instance is a pin on a cell in the hierarchy of a design.

If **-leaf** is used, then a list of leaf pins of the net is returned.

To connect nets to ports or pins, use **connect_net**. To break connections, use **disconnect_net**.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses **all_connected** to return the objects connected to MY_NET:

```
prompt> all_connected MY_NET
```

```
prompt> connect_net MY_NET OUT3  
Connecting net 'MY_NET' to port 'OUT3'.
```

```
prompt> connect_net MY_NET U65/Z  
Connecting net 'MY_NET' to pin 'U65/Z'.
```

```
prompt> all_connected MY_NET  
{OUT3 U65/Z}
```

```
prompt> all_connected OUT3  
{MY_NET}
```

```
prompt> all_connected U65/Z  
{MY_NET}
```

This next example uses **all_connected** to associate net load capacitance with the net n47, which is connected to the pin instance C/Z:

```
prompt> set_load 0.147 [all_connected [get_pins U0/U1/C/Z]]  
Set "load" attribute to 0.147 for net "U0/U1/n47"
```

SEE ALSO

```
all_inputs(2)  
all_outputs(2)  
connect_net(2)  
create_net(2)  
current_design(2)  
disconnect_net(2)  
remove_net(2)
```

all_connectivity_fanin

Reports pins, ports, or cells in the fanin of specified sinks.

SYNTAX

```
list all_connectivity_fanin  
-to sink_list  
[-startpoints_only]  
[-only_cells]  
[-flat]  
[-levels count]
```

Data Types

<i>sink_list</i>	list
<i>count</i>	int

ARGUMENTS

-to *sink_list*
Reports a list of sink pins, ports, or nets in the design and connectivity based fanin of each sink in the *sink_list*. If you specify a net, the effect is the same as listing all driver pins on the net.

-startpoints_only
Returns only the connectivity startpoints.

-only_cells
Results in a set of all the cells in the connectivity fanin of the *sink_list*.

-flat
Specifies to function in the flat mode of operation. The two major modes in which **all_connectivity_fanin** functions are hierarchical (default) and flat. When in hierarchical mode, only objects from the same hierarchy level as the current sink are returned. Thus, pins within a level of hierarchy lower than that of the sink are used for traversal but they will not be reported.

-levels *count*
Stops traversal when reaching the perimeter of the search of *count* hops, where counting is performed over the layers of cells that are of equidistant from the sink.

DESCRIPTION

This command reports the connectivity fanin of specified sink pins, ports, or nets in the design. A pin is considered to be in the connectivity fanin of a sink if there is a path through combinational logic from the pin to that sink. The fanin report stops at the clock pins of registers (sequential cells).

NOTE: This command reports same results as `all_fanin` except the cases 1. *false paths* are not considered in this commands 2. manually created clocks are not treated as

paths start points

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following examples show the connectivity fanin of a port in the design. The design comprises three inverters in a chain named *iv1*, *iv2*, and *iv3*. The *iv1* and *iv2* inverters are hierarchically combined in a larger cell named *ii2*.

```
prompt> all_connectivity_fanin -to tout  
{ "ii2/hin", "iv3/in", "iv3/out", "tin", "ii2/hout", "tout" }
```

```
prompt> all_connectivity_fanin -to tout -flat  
  
{ "ii2/iv1/U1/a", "ii2/iv2/U1/z", "tin", "iv3/U1/a", "ii2/iv1/U1/z",  
  "ii2/iv2/U1/a", "iv3/U1/z", "tout" }
```

SEE ALSO

`all_fanin(2)`
`all_fanout(2)`
`report_transitive_fanin(2)`

all_connectivity_fanout

Returns a set of pins, ports, or cells in the fanout of the specified sources.

SYNTAX

```
list all_connectivity_fanout  
-from source_list  
[-endpoints_only]  
[-only_cells]  
[-flat]  
[-levels count]
```

Data Types

<i>source_list</i>	list
<i>count</i>	int

ARGUMENTS

-from *source_list*
Specifies a list of source pins, ports, or nets in the design. The connectivity fanout of each source in *source_list* is reported. If a net is specified, the effect is the same as listing all load pins on the net.

-endpoints_only
Returns only connectivity endpoints as a result.

-only_cells
Results in a set of all cells in the connectivity fanout of the *source_list*, rather than a set of pins or ports.

-flat
Specifies to function in the flat mode of operation. The two major modes in which **all_connectivity_fanout** functions are hierarchical (default) and flat. When in hierarchical mode, only objects from the same hierarchy level as the current source are returned. Thus, pins within a level of hierarchy lower than that of the source are used for traversal but are not reported.

-levels *count*
Stops traversal when reaching the perimeter of the search of *count* hops, where counting is performed over the layers of cells that are of equidistant from the source.

DESCRIPTION

This command reports the connectivity fanout of specified source pins, ports, or nets in the design. A pin is considered to be in the connectivity fanout of a sink if there is a path through combinational logic from that source to the pin. The fanout report stops at the inputs to registers (sequential cells).

NOTE: This command reports same results as `all_fanout` except the cases 1. *false*

paths are not considered in this commands 2. manually created clocks are not treated as paths start points

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows the connectivity fanout of a port in the design. The design comprises three inverters in a chain named *iv1*, *iv2*, and *iv3*. The *iv1* and *iv2* inverters are hierarchically combined in a larger cell named *ii2*.

```
prompt> all_connectivity_fanout -from tin  
{ "iv3/out", "tout", "iv3/in", "ii2/hin", "ii2/hout", "tin" }
```

```
prompt> all_connectivity_fanout -from tin -flat  
{ "tout", "ii2/iv2/U1/z", "ii2/iv1/U1/a", "iv3/U1/z", "iv3/U1/a",  
  "ii2/iv2/U1/a", "ii2/iv1/U1/z", "tin" }
```

```
prompt> all_connectivity_fanout -from tin -levels 1 -only_cells  
{ "iv3", "ii2" }
```

SEE ALSO

`all_fanout(2)`
`all_fanin(2)`
`all_connectivity_fanin(2)`
`report_transitive_fanout(2)`

all_critical_cells

Returns a collection of critical leaf cells in the top hierarchy of the current design.

SYNTAX

```
collection all_critical_cells  
[-slack_range range_value]
```

Data Types

range_value float

ARGUMENTS

`-slack_range range_value`
Specifies a margin of slack for searching top-hierarchy leaf cells in paths whose slacks are in the specified *range_value* relative to the worst slack of the current design. A top-hierarchy leaf cell is a cell in the top hierarchy and with no hierarchy underneath. The *range_value* must be expressed in the same units as the technology library used during optimization. In addition, *range_value* must be positive or 0.0. If no *range_value* is specified, the default is 0.0. A *range_value* of 0.0 means that top-hierarchy leaf cells in the most critical paths (the ones with the worst violations) are returned. If a positive *range_value* is specified, all top-hierarchy leaf cells are returned if they are in near-critical paths with slacks in the *range_value* relative to the worst slack of the current design.

DESCRIPTION

Returns a collection of leaf cells that are in the top hierarchy and in some path with a slack in the *range_value* relative to the worst slack of the current design. This command returns only those cells with no hierarchy underneath. If all timing paths passing through a cell are unconstrained, the cell is assumed to be non-critical, and thus is not returned. You can use this command, along with the **group** command, to group together into a new subhierarchy those cells in the most critical paths. Then, you can try various optimization techniques on the new subhierarchy.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example lists all top-hierarchy leaf cells in the worst critical paths of the current design.

```
prompt> all_critical_cells
```

The following example lists all top-hierarchy leaf cells in those paths within range 5.0 relative to the worst slack of the current design.

```
prompt> all_critical_cells -slack_range 5.0
```

The following example groups all top-hierarchy leaf cells in the worst critical paths into a new hierarchy, called CRIT, under the top hierarchy of the current design.

```
prompt> group [all_critical_cells] -design_name CRIT
```

The following example reports cell connections of all top-hierarchy leaf cells in the worst critical paths of the current design.

```
prompt> report_cell -connections [all_critical_cells]
```

SEE ALSO

```
all_critical_pins(2)  
current_design(2)  
group(2)  
report_cell(2)
```

all_critical_pins

Returns a collection of critical endpoints or startpoints in the current design.

SYNTAX

```
collection all_critical_pins  
[-type endpoint | startpoint]  
[-slack_range range_value]
```

Data Types

range_value float

ARGUMENTS

-type endpoint | startpoint
Specifies that the pins or ports to be searched are either timing endpoints or timing startpoints. The default is 'endpoint'.

-slack_range range_value
Specifies a margin of slack for searching timing endpoints (or startpoints) in paths whose slacks are in the specified *range_value* relative to the worst slack of the current design. The *range_value* must be expressed in the same units as the technology library used during optimization. In addition, *range_value* must be positive or 0.0. If no *range_value* is specified, the default is 0.0. A *range_value* of 0.0 means that endpoints (or startpoints) in the most critical paths (the ones with the worst violations) will be returned. If a positive *range_value* is specified, endpoints (or startpoints) in near-critical paths with slacks in the *range_value* relative to the worst slack of the current design will be returned.

DESCRIPTION

Returns a collection of endpoints (or startpoints) which are in some timing path with a slack in the *range_value* relative to the worst slack of the current design. For a pin, if all timing paths passing through it are unconstrained, it is assumed to be non-critical, and thus it will not be returned. Usually, this command may be used together with **all_fanin** or **all_fanout** commands to report some useful information (e.g., cells in the transitive timing fanin cone of the most critical endpoints).

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example lists all endpoints in the worst critical paths of the current design.

```
prompt> all_critical_pins
```

The following example lists all startpoints in those paths within range 5.0 relative to the worst slack of the current design.

```
prompt> all_critical_pins -type startpoint -slack_range 5.0
```

The following example reports cells in the 3-level transitive fanin timing cone of endpoints in the worst critical paths.

```
prompt> report_cell [all_fanin -to [all_critical_pins] \  
        -only_cells -levels 3 -flat]
```

The following example reports the logic in the transitive fanin of endpoints in the worst critical paths of the current design.

```
prompt> report_transitive_fanin -to [all_critical_pins]
```

SEE ALSO

```
current_design(2)  
all_fanin(2)  
report_cell(2)  
report_transitive_fanin(2)  
report_transitive_fanout(2)  
all_critical_cells(2)  
group(2)
```

all_designs

Returns a collection containing all designs in the current design.

SYNTAX

collection **all_designs**

ARGUMENTS

There are no arguments to this command.

DESCRIPTION

The **all_designs** command returns a collection containing the designs in the current design hierarchy in bottom-up order. You must set the current design using the **current_design** command before using **all_designs**.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

SEE ALSO

current_design(2)

all_dont_touch

Returns a collection of **dont_touch** cells or nets from the current design or from the specified input collection.

SYNTAX

```
collection all_dont_touch  
-cells | -nets  
[input_coll]
```

Data Types

input_coll string

ARGUMENTS

-cells

Specifies that all cells with the **dont_touch** attribute are to be returned. The **-cells** and **-nets** options are mutually exclusive: use only one.

-nets

Specifies that all nets with the **dont_touch** attribute are to be returned. The **-cells** and **-nets** options are mutually exclusive: use only one.

input_coll

Searches the specified input collection and returns the collection of cells or nets having the **dont_touch** attribute. Objects are to be searched only from the the content of *input_coll* rather than from the entire design. By default, this option is off.

DESCRIPTION

This command returns the collection of cells or nets that have the **dont_touch** attribute from the design or from a list that you can specify. You can use wild cards, such as an asterisk (*) or question mark (?), when specifying the input cell list.

The command returns a standard Tcl collection. You can perform all of the generic collection manipulating commands, such as **foreach_in_collection**, **sizeof_collection**, and so on, on this collection handle.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example shows the collection of all cells with the **dont_touch** attribute in the design.

```
all_dont_touch
```



```
prompt> all_dont_touch -cells  
{U20 U21 U23 SBLK/U1 SBLK/U2}
```

The following example shows the collection of all nets with the **dont_touch** attribute from an existing collection where *COLL* is its handle.

```
prompt> all_dont_touch -nets $COLL  
{n2 n5 SBLK/n2}
```

SEE ALSO

```
foreach_in_collection(2)  
get_attribute(2)  
get_cells(2)  
get_nets(2)  
report_attribute(2)  
set_attribute(2)  
set_dont_touch(2)  
set_dont_touch_network(2)  
set_dont_touch_placement(2)  
sizeof_collection(2)
```

all_drc_violated_nets

Returns a collection of DRC-violated nets from the current design or from the specified input collection.

SYNTAX

```
collection all_drc_violated_nets  
-max_capacitance | -max_transition | -max_fanout  
[input_coll]  
[-bound upper]  
[-threshold threshold]
```

Data Types

<i>input_coll</i>	collection
<i>upper</i>	float
<i>threshold</i>	float

ARGUMENTS

-max_capacitance
Returns the collection of maximum capacity nets that violate design rule checking (DRC), as designated by **drc_violated_nets**.

-max_transition
Returns the collection of maximum transition nets that violate DRC, as designated by **drc_violated_nets**.

-max_fanout
Returns the collection of maximum fanout nets that violate DRC, as designated by **drc_violated_nets**.

input_coll
Searches the specified input collection and returns the requested collection of nets that violate DRC constraints. Objects are searched only from the contents of the specified input collection, rather than from the design.

-bound *upper*
Captures all DRC-violated nets that have values less than or equal to the bound specified by *upper*. By default, this option is off.

-threshold *threshold*
Captures all DRC-violated nets that have values greater than or equal to the threshold specified by *threshold*. By default, this option is off.

DESCRIPTION

The command returns a collection of DRC-violated nets from the current design or from the specified input collection. The tool can search the entire design hierarchy. It returns DRC-violated nets from the entire design, not just from the top level.

all_drc_violated_nets

This command returns the three most common types of DRC violations: `max_capacitance`, `max_transition`, and `max_fanout`. You can specify each of the violations separately in this command. If you do not specify an option, a collection of all three types of DRC-violated nets is returned.

The command returns a standard Tcl collection. You can perform all of the generic collection manipulating commands, such as **`foreach_in_collection`**, **`sizeof_collection`**, and so on, on this collection handle.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example shows the collection of all maximum capacity DRC-violated nets from the design.

```
prompt> all_drc_violated_nets -max_cap  
{n1 n2 n3 abc/n14}
```

The following example shows the collection of all **`-max_fanout`** DRC-violated nets from an existing collection stored in `$COLL`.

```
prompt> all_drc_violated_nets -max_fanout $COLL  
{n1 n3 abc/n14}
```

SEE ALSO

`foreach_in_collection(2)`
`sizeof_collection(2)`

all_fanin

Reports pins, ports, or cells in the fanin of specified sinks.

SYNTAX

```
collection all_fanin  
-to sink_list  
[-startpoints_only]  
[-exclude_bboxes]  
[-break_on_bboxes]  
[-only_cells]  
[-flat]  
[-levels count]
```

Data Types

```
sink_list      list  
count          int
```

ARGUMENTS

-to *sink_list*
Reports a list of sink pins, ports, or nets in the design and a timing fanin of each sink in the *sink_list*. If you specify a net, the effect is the same as listing all driver pins on the net.

-startpoints_only
Returns only the timing startpoints.

-exclude_bboxes
Excludes blackboxes from the final result.

-break_on_bboxes
Stops timing fanin traversal on blackboxes.

-only_cells
Results in a set of all the cells in the timing fanin of the *sink_list*.

-flat
Specifies to function in the flat mode of operation. The two major modes in which **all_fanin** functions are hierarchical (default) and flat. When in hierarchical mode, only objects from the same hierarchy level as the current sink are returned. Thus, pins within a level of hierarchy lower than that of the sink are used for traversal but are not reported.

-levels *count*
Stops traversal when reaching the perimeter of the search of *count* hops, where counting is performed over the layers of cells that are of equidistant from the sink.

DESCRIPTION

This command reports the timing fanin of specified sink pins, ports, or nets in the design. A pin is considered to be in the timing fanin of a sink if there is a timing path through combinational logic from the pin to that sink. The fanin report stops at the clock pins of registers (sequential cells).

Multicorner-Multimode Support

Depending on the options used, this command either uses the current scenario or has no dependency on scenario-specific information.

EXAMPLES

The following examples show the timing fanin of a port in the design. The design comprises three inverters in a chain named *iv1*, *iv2*, and *iv3*. The *iv1* and *iv2* inverters are hierarchically combined in a larger cell named *ii2*.

```
prompt> all_fanin -to tout
{ii2/hin iv3/in iv3/out tin ii2/hout tout}
```

```
prompt> all_fanin -to tout -flat
{ii2/iv1/U1/a ii2/iv2/U1/z tin iv3/U1/a ii2/iv1/U1/z
ii2/iv2/U1/a iv3/U1/z tout}
```

SEE ALSO

```
all_fanout(2)
report_transitive_fanin(2)
```

all_fanout

Returns a set of pins, ports, or cells in the fanout of the specified sources.

SYNTAX

```
collection all_fanout  
-clock_tree  
-from source_list  
[-endpoints_only]  
[-exclude_bboxes]  
[-break_on_bboxes]  
[-only_cells]  
[-flat]  
[-levels count]
```

Data Types

<i>source_list</i>	list
<i>count</i>	int

ARGUMENTS

- clock_tree**
Uses all clock source pins and/or ports in the design as the list of sources. Clock sources are specified by using the **create_clock** command. If there are no clocks, or if the clocks have no sources, the report is empty. Use the **report_clock** command to list the sources for all clocks in the design. The **-clock_tree** option generates a report that displays the clock trees or networks in the design. The **-clock_tree** and **-from** options are mutually exclusive.
- from *source_list***
Specifies a list of source pins, ports, or nets in the design. The timing fanout of each source in *source_list* is reported. If a net is specified, the effect is the same as listing all load pins on the net. The **-clock_tree** and **-from** options are mutually exclusive.
- endpoints_only**
Returns only timing endpoints as a result.
- exclude_bboxes**
Excludes blackboxes from the final result.
- break_on_bboxes**
Stops timing fanout traversal on blackboxes.
- only_cells**
Results in a set of all cells in the timing fanout of the *source_list*, rather than a set of pins or ports.
- flat**
Specifies to function in the flat mode of operation. The two major modes in

which **all_fanout** functions are hierarchical (default) and flat. When in hierarchical mode, only objects from the same hierarchy level as the current source are returned. Thus, pins within a level of hierarchy lower than that of the source are used for traversal but are not reported.

-levels *count*

Stops traversal when reaching the perimeter of the search of *count* hops, where counting is performed over the layers of cells that are of equidistant from the source.

DESCRIPTION

This command reports the timing fanout of specified source pins, ports, or nets in the design. A pin is considered to be in the timing fanout of a sink if there is a timing path through combinational logic from that source to the pin. The fanout report stops at the inputs to registers (sequential cells). The source pins or ports are specified by using the **-clock_tree** or **-from source_list** option.

Multicorner-Multimode Support

Depending on the options used, this command either uses the current scenario or has no dependency on scenario-specific information.

EXAMPLES

The following example shows the timing fanout of a port in the design. The design comprises the following three inverters in a chain, iv1, iv2, and iv3. The iv1 and iv2 inverters are hierarchically combined in a larger cell named ii2.

```
prompt> all_fanout -from tin
{iv3/out tout iv3/in ii2/hin ii2/hout tin}

prompt> all_fanout -from tin -flat
{tout ii2/iv2/U1/z ii2/iv1/U1/a iv3/U1/z iv3/U1/a
ii2/iv2/U1/a ii2/iv1/U1/z tin}

prompt> all_fanout -from tin -levels 1 -only_cells
{iv3 ii2}
```

SEE ALSO

```
all_fanin(2)
create_clock(2)
report_clock(2)
report_transitive_fanout(2)
```

all_fixed_placement

Returns the collection of all fixed-placement cells or ports in the design or from a list of objects in the input collection.

SYNTAX

```
collection all_fixed_placement  
-cells | -ports  
[input_coll]
```

Data Types

input_coll collection

ARGUMENTS

-cells

Specifies that the collection of fixed-placement cells is to be returned. If you specify this option, you must not specify the **-ports** option: They are mutually exclusive.

-ports

Specifies that the collection of fixed-placement ports is to be returned. If you specify this option, you must not specify the **-cells** option: They are mutually exclusive. All the ports which are either having implicit `is_fixed` attribute or valid location are treated as fixed-placement ports

input_coll

Specifies the input collection to search and return the collection of fixed-placement objects. Objects are to be searched only from the the content of the specified input collection rather than from the design.

DESCRIPTION

This command can return the collection of all fixed-placement objects in the design. The tool searches the entire design hierarchy. It returns fixed-placement objects from the entire design, not just from the top level.

If you specify a value for *input_coll*, the command searches for fixed-placement objects only from the specified collection, rather than from the entire design.

If you do not specify either the **-cells** or **-ports** option, the **all_fixed_placement** command returns an error.

A standard Tcl collection is returned. All of the generic collection manipulating commands, such as **foreach_in_collection**, **sizeof_collection**, and so on, can be performed on this collection.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows the collection of all fixed-placement cells from the design.

```
prompt> all_fixed_placement -cells  
{ii2/hin, iv3/in, iv3/out, tin, ii2/hout, tout}
```

The following example shows the collection of all fixed-placement ports from the design.

```
prompt> all_fixed_placement -ports  
{"hin", "in", "out", "tin", "hout", "tout"}
```

The following example shows the collection of all fixed-placement cells from an existing collection stored in \$COLL.

```
prompt> all_fixed_placement -cells $COLL  
{ii2/hin iv3/in iv3/out tin ii2/hout tout}
```

SEE ALSO

```
foreach_in_collection(2)  
remove_dont_touch_placement(2)  
set_dont_touch_placement(2)  
sizeof_collection(2)
```

all_high_fanout

Returns a collection of high-fanout nets from the current design or from the specified input collection.

SYNTAX

```
collection all_high_fanout  
-nets  
[-threshold value]  
[input_coll]  
[-through_buf_inv]
```

Data Types

<i>value</i>	float
<i>input_coll</i>	collection

ARGUMENTS

-nets
Specifies that the collection of high-fanout nets is to be returned.

-threshold *value*
Specifies a threshold value used to determine if a net is high-fanout net. The *value* is a user-specified value. By default, the "high_fanout_net_threshold" value is used to determine if a net is a high-fanout net.

input_coll
Search the specified input collection for high_fanout nets. Objects are to be searched only from the contents of the specified input collection rather than from the design.

-through_buf_inv
Indicates to treat a buffer tree as transparent. The leaf loads of the buffer tree are treated as the fanouts of the net.

DESCRIPTION

This command can return the collection of all high-fanout nets in the design. The tool searches the entire design hierarchy. It returns nets from the entire design, not just from the top level.

To determine if a net is a high-fanout net, use the **high_fanout_net_threshold** variable value as the threshold value. The command returns all nets with a fanout count higher than this threshold as high-fanout nets. You can also specify the threshold by using the optional **-threshold** option.

If you specify a value for *input_coll*, the command searches for high-fanout objects only from the specified collection, rather than from the entire design.

all_high_fanout

A standard Tcl collection is returned. All of the generic collection manipulating commands, such as **foreach_in_collection**, **sizeof_collection**, and so on, can be performed on this collection handle.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows the collection of all the high-fanout nets from the design.

```
prompt> all_high_fanout -nets  
{ii2/hin iv3/in iv3/out tin ii2/hout tout}
```

The following example shows the collection of all the high-fanout nets from an existing collection stored in \$COLL.

```
prompt> all_high_fanout -nets $COLL  
{ii2/hin iv3/in iv3/out tin ii2/hout tout}
```

The following example shows the collection of all the high-fanout nets from the design having a fanout count of more than 100.

```
prompt> all_high_fanout -nets -threshold 100  
{ii2/hin iv3/in iv3/out tin ii2/hout tout}
```

SEE ALSO

all_fanin(2)
all_fanout(2)
foreach_in_collection(2)
report_net_fanout(2)
sizeof_collection(2)

all_ideal_nets

Returns a collection of ideal nets from the current design or from the specified input collection.

SYNTAX

```
collection all_ideal_nets  
[input_coll]
```

Data Types

input_coll collection

ARGUMENTS

input_coll
Specifies the input collection to search for ideal nets.
If you do not specify this argument, the command searches for ideal nets in the current design.

DESCRIPTION

This command returns a collection of ideal nets.

If you specify a value for *input_coll*, the command searches for ideal nets only in the specified collection.

If you do not specify a collection, the command searches for ideal nets in the current design. The tool searches the entire design hierarchy. It returns ideal nets from the entire design, not just from the top level.

A standard Tcl collection is returned. All of the generic collection manipulating commands, such as **foreach_in_collection**, **sizeof_collection**, and so on, can be performed on this collection handle.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example returns a collection of all ideal nets from the design.

```
prompt> all_ideal_nets  
{ii2/hin iv3/in iv3/out tin ii2/hout tout}
```

The following example returns a collection of all ideal nets from an existing collection stored in \$COLL.

```
prompt> all_ideal_nets $COLL  
{ii2/hin iv3/in iv3/out tin ii2/hout tout}
```

SEE ALSO

```
all_fanout(2)  
foreach_in_collection(2)  
sizeof_collection(2)
```

all_inputs

Returns a collection of input or inout ports in the current design.

SYNTAX

```
collection all_inputs  
[-clock clock_name]  
[-edge_triggered | -level_sensitive]
```

Data Types

clock_name string

ARGUMENTS

```
-clock clock_name  
    Limits the search to ports that have input delay relative to clock_name.  
  
-edge_triggered  
    Limits the search to ports that have edge-triggered input delay as specified  
    by set_input_delay -clock.  
  
-level_sensitive  
    Limits the search to ports that have level-sensitive input delay as specified  
    by set_input_delay -level_sensitive.
```

DESCRIPTION

Returns a collection of all input or inout ports in the current design, unless one of the options limits the search. The **all_inputs** command is usually used with a command that places attributes on input ports. To get detailed information on ports in the current design, use the **report_port** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example lists all input ports in the current design.

```
prompt> all_inputs  
{A1 A2 BIDIR1}
```

The following example sets the drive value of all the input ports on the current design to 10:

```
prompt> set_drive 10.0 [all_inputs]
```

The following example marks with a multicycle value of 0 all paths from inputs having level-sensitive input delay relative to PHI1 to level-sensitive registers clocked by PHI1.

```
prompt> set_multicycle_path 0 \  
        -from [all_inputs -clock PHI1 -level_sensitive] \  
        -to [all_registers -data_pins -clock PHI1]
```

SEE ALSO

```
all_outputs(2)  
current_design(2)  
report_port(2)  
set_drive(2)  
set_input_delay(2)  
set_multicycle_path(2)
```

all_isolation_cellsfp

Returns a collection of isolation cells available in the design.

SYNTAX

collection **all_isolation_cellsfp**

DESCRIPTION

The **all_isolation_cells** command returns a collection of the existing isolation cells in the design. The enabled level shifters also work as isolation cells, but that type of cell is not returned by this command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

SEE ALSO

check_mv_design(2)
all_level_shifters(2)

all_level_shiftersfp

Returns a collection of level-shifter cells available in the design.

SYNTAX

```
collection all_level_shiftersfp  
[-type els | simple]
```

ARGUMENTS

`-type els | simple`
Specifies the type of level-shifter cells to include in the collection. If you specify `-type els`, the command returns only the enabled level-shifter cells. If you specify `-type simple`, the command returns only regular level-shifter cells. If you do not specify this option, the command returns all level-shifter cells.

DESCRIPTION

The **all_level_shifters** command returns a collection of existing level-shifter cells in the design.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

SEE ALSO

`check_level_shifters(2)`

all_macro_cells

Returns a collection of all macro cells in the design or from a list of objects in the input collection.

SYNTAX

```
collection all_macro_cells
[input_coll]
```

Data Types

input_coll collection

ARGUMENTS

input_coll
Specifies the input collection to search and return the collection of macro cells. Objects are to be searched only from the the content of the specified input collection rather than from the design.

DESCRIPTION

This command can return the collection of all macro cells from the design. The tool searches the entire design hierarchy. It returns cells from the entire design, not just from the top level. This command uses the same definition as used in the legalizing process to determine if a cell is a macro cell.

If you specify a value for *input_coll*, the command searches for macro cells only from the specified collection, rather than from the entire design.

A standard Tcl collection is returned. All of the generic collection manipulating commands, such as **foreach_in_collection**, **sizeof_collection**, and so on, can be performed on this collection.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows the collection of all macro cells from the design.

```
prompt> all_macro_cells
{"macro1", "top/macro2", "top/macro4", "macro10"}
```

The following example shows the collection of all macro_cells from an existing collection stored in \$COLL.

```
prompt> all_macro_cells $COLL
```

all_macro_cells

```
{macro1 macro2}
```

SEE ALSO

```
foreach_in_collection(2)  
sizeof_collection(2)  
set_cell_type(2)
```

all_mtcmos_cellsfp

Returns a collection of MTCMOS cells available in the design.

SYNTAX

```
collection all_mtcmos_cellsfp  
[-type coarse | fine]
```

ARGUMENTS

```
-type coarse | fine
```

Specifies the type of the MTCMOS cell to include in the collection.
If you specify `-type coarse`, the command returns only the coarse-grained MTCMOS cells. If you specify `-type fine`, the command returns only the fine-grained MTCMOS cells. If you do not specify this option, the command returns all MTCMOS cells.

DESCRIPTION

The **all_mtcmos_cells** command returns a collection of existing MTCMOS cells in the design.

all_objects_in_bounding_box

Returns the collection of all cells and/or nets in the specified bounding box in the design or from a list of objects in the input collection.

SYNTAX

```
collection all_objects_in_bounding_box  
-cells | -nets  
[-phys_cells]  
-coordinates {llx lly urx ury}  
[-flat]  
[input_coll]
```

Data Types

<i>llx</i>	integer
<i>lly</i>	integer
<i>urx</i>	integer
<i>ury</i>	integer
<i>input_coll</i>	collection

ARGUMENTS

-cells
Specifies that all the cells from the bounding box are to be returned. The **-cells** and **-nets** options are mutually exclusive; you can use only one.

-nets
Specifies that all the nets from the bounding box are to be returned. The **-cells** and **-nets** options are mutually exclusive; you can use only one.

-phys_cells
Returns only physical-only cells.

-coordinates {llx lly urx ury}
Specifies the coordinates of the area from which the objects are to be returned.

-flat
Gets cells or nets inside ilm.

input_coll
Specifies the input collection to search and from which to return the collection of cells or nets from the specified bounding box. Objects are to be searched only from the content of the specified input collection, rather than from the design.

DESCRIPTION

This command can return the collection of all objects from the rectangular area (bounding box) specified by the coordinates described by the **-coordinates** option.

The tool searches the entire design hierarchy. It returns objects from the entire design, not just from the top level.

If you do not specify the **-cell** or **-nets** option, the command returns a collection of both objects together from the specified bounding box.

If you specify a value for *input_coll*, the command searches for objects only from the specified collection, rather than from the entire design. You must specify bounding box coordinates; this command does not provide a default.

A standard Tcl collection is returned. All of the generic collection-manipulating commands, such as **foreach_in_collection**, **sizeof_collection**, and so on, can be performed on this collection.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows the collection of all cells in the bounding box described by the 10 10 90 90 coordinates from the design.

```
prompt> all_objects_in_bounding_box -cells \  
-coordinate {10 10 90 90}  
{u1 u2 u3 abc/u14}
```

The following example shows the collection of all nets in the bounding box described by the 10 10 90 90 coordinates from the design.

```
prompt> all_objects_in_bounding_box -nets \  
-coordinate {10 10 90 90}  
{n1 n2 n3 abc/n14}
```

SEE ALSO

`foreach_in_collection(2)`
`sizeof_collection(2)`

all_outputs

Returns a collection of output or inout ports in the current design.

SYNTAX

```
collection all_outputs  
[-clock clock_name]  
[-edge_triggered | -level_sensitive]
```

Data Types

clock_name string

ARGUMENTS

-clock *clock_name*
Limits the search to ports that have output delay relative to *clock_name*.

-edge_triggered
Limits the search to ports that have edge-triggered output delay as specified by **set_output_delay -clock**.

-level_sensitive
Limits the search to ports that have level-sensitive output delay as specified by **set_output_delay -level_sensitive**.

DESCRIPTION

Returns a collection of all output or inout ports in the current design, unless one of the options limits the search. The **all_outputs** command is usually used with a command that places attributes on output ports. To get detailed information on ports in the current design, use the **report_port** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example lists all output ports.

```
prompt> all_outputs  
{OUT1 OUT2 BIDIR1}
```

The following example sets a capacitive load of 3.5 on each output port.

```
prompt> set_load 3.5 [all_outputs]
```

The following example sets paths leading to output ports clocked by TSTCLK to be

false.

```
prompt> set_false_path -to [all_outputs -clock TSTCLK]
```

SEE ALSO

`all_inputs(2)`
`current_design(2)`
`report_port(2)`
`set_false_path(2)`
`set_load(2)`
`set_output_delay(2)`

all_physical_only_cells

Returns a collection of all physical-only (filler) cells in the design or from a list of objects in the input collection.

SYNTAX

```
collection all_physical_only_cells  
[-lib_cells lib_list | -cell_name list]  
[-coordinates {llx lly urx ury}]  
[input_coll_handle]
```

Data Types

<i>lib_list</i>	list
<i>list</i>	list
<i>llx</i>	float
<i>lly</i>	float
<i>urx</i>	float
<i>ury</i>	float
<i>input_coll_handle</i>	collection

ARGUMENTS

-lib_cells *lib_list*
Specifies a list of library cells. The tool searches physical-only cells of these library cell types. The default is to return all physical-only cells from the entire design area or from whatever option you specify.

-cell_name *list*
Specifies the base instance name filler cells to be searched from the design. The default is to return all physical-only cells from the entire design area or from whatever option you specify.

-coordinates {*llx lly urx ury*}
Specifies the coordinates of the area from which to return the physical-only cells. The default is to return all physical-only cells from the entire design area or from whatever option you specify.

input_coll_handle
Specifies the input collection to search and from which to return the collection of physical-only cells. Objects are to be searched only from the content of the specified input collection rather than from the design. The default is to return all physical-only cells from the entire design area or from whatever option you specify.

DESCRIPTION

This command returns the collection of all physical-only cells from the design. The tool searches the entire design hierarchy. It returns physical-only cells from the entire design, not just from the top level.

You can use wild cards, such as an asterisk (*) or question mark (?), when specifying cell instances names.

A standard Tcl collection is returned. All of the generic collection-manipulating commands, such as the **foreach_in_collection** and **sizeof_collection** commands, for example, can be performed on this collection.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example collects all the physical-only cells from the design.

```
prompt> all_physical_only_cells  
{fill1 fill2 fill3 abc/fill4}
```

The following example collects all physical-only cells from the design that have a lib cell of type *OKOAFILLER*.

```
prompt> all_physical_only_cells -lib_cell OKOAFILLER  
{hin in out tin hout tout}
```

The following example collects all the physical-only cells from an existing collection stored in *\$COLL*.

```
prompt> all_physical_only_cells $COLL  
{ii2/hin iv3/in iv3/out tin ii2/hout tout}
```

The following example collects all physical-only cells from the design that have an instance name starting with *fill*.

```
prompt> all_physical_only_cells -cell_name fill*  
{fill1 fill2 fill3}
```

SEE ALSO

```
all_physical_only_nets(2)  
all_physical_only_ports(2)  
foreach_in_collection(2)  
sizeof_collection(2)
```

all_physical_only_nets

Returns a collection of all physical-only nets in the design or from a list of objects in the input collection.

SYNTAX

```
collection all_physical_only_nets  
[input_coll]
```

Data Types

input_coll collection

ARGUMENTS

input_coll
Specifies the input collection to search and return the collection of physical-only nets. Objects are to be searched only from the the content of the specified input collection rather than from the design.

DESCRIPTION

This command can return the collection of physical-only nets from the design. The tool searches the entire design hierarchy. It returns physical-only nets from the entire design, not just from the top level. It returns power and ground nets together: They are not returned separately.

If you specify a value for *input_coll*, the command searches for physical-only nets only from the specified collection, rather than from the entire design.

A standard Tcl collection is returned. All of the generic collection manipulating commands, such as **foreach_in_collection**, **sizeof_collection**, and so on, can be performed on this collection.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example returns a collection of all physical-only nets from the design.

```
prompt> all_physical_only_nets  
{pnet1 pnet2 gnd vcc}
```

The following example returns a collection of all the physical-only nets from an existing collection stored in \$COLL.

```
prompt> all_physical_only_nets $COLL  
{vcc vdd gnd vcc5}
```

SEE ALSO

```
all_physical_only_cells(2)  
all_physical_only_nets(2)  
foreach_in_collection(2)  
sizeof_collection(2)
```

all_physical_only_ports

Returns a collection of all physical-only ports in the design or from a list of objects in the input collection.

SYNTAX

```
collection all_physical_only_ports  
[input_coll]
```

Data Types

input_coll collection

ARGUMENTS

input_coll
Specifies the input collection to search and return the collection of physical-only ports. Objects are to be searched only from the the content of the specified input collection rather than from the design.

DESCRIPTION

This command can return the collection of all physical-only ports from the design. The tool searches the entire design hierarchy. It returns physical-only ports from the entire design, not just from the top level.

If you specify a value for *input_coll*, the command searches for physical-only ports only from the specified collection, rather than from the entire design.

A standard Tcl collection is returned. All of the generic collection manipulating commands, such as **foreach_in_collection**, **sizeof_collection**, and so on, can be performed on this collection.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example returns a collection of all physical-only ports from the design.

```
prompt> all_physical_only_ports  
{port1 port2 port3}
```

The following example returns a collection of all the physical-only ports from an existing collection stored in \$COLL.

```
prompt> all_physical_only_ports $COLL
```

```
{pport portp1 pport2}
```

SEE ALSO

```
all_physical_only_cells(2)  
all_physical_only_nets(2)  
foreach_in_collection(2)  
sizeof_collection(2)
```

all_registers

Returns a collection of sequential cells or pins in the current design.

SYNTAX

```
collection all_registers
[-no_hierarchy]
[-clock clock_name]
[-rise_clock rise_clock_name]
[-fall_clock fall_clock_name]
[-cells]
[-data_pins]
[-clock_pins]
[-slave_clock_pins]
[-output_pins]
[-inverted_output]
[-level_sensitive | -edge_triggered]
[-master_slave]
```

Data Types

<i>clock_name</i>	string
<i>rise_clock_name</i>	string
<i>fall_clock_name</i>	string

ARGUMENTS

- no_hierarchy**
Limits the search to only the current level of hierarchy. Subdesigns are not searched. By default, this option is off.
- clock *clock_name***
Considers only sequential cells clocked by *clock_name* in the search. By default, this option is off.
- rise_clock *rise_clock_name***
Considers only sequential cells clocked by *rise_clock_name* and having the open edge effectively the rising clock edge. By default, this option is off.
- fall_clock *fall_clock_name***
Considers only sequential cells clocked by *fall_clock_name* and having the open edge effectively the falling clock edge. By default, this option is off.
- cells**
Returns a collection sequential cells that meet the search criteria. This is the default. If no object type is specified in the command (**-cells**, **-data_pins**, **-clock_pins**, etc.), then the command returns a collection of sequential cells.
- data_pins**
Returns a collection of data pins of the sequential cells that meet the search criteria.

-clock_pins
Returns a collection of clock pins of the sequential cells that meet the search criteria.

-slave_clock_pins
Returns a collection of slave clock pins of master-slave registers that meet the search criteria. Slave clock pins are specified as "clocked_on_also" in the library.

-output_pins
Returns a collection of output pins of the sequential cells that meet the search criteria.

-inverted_output
Limits the search to flip-flops that have their output phase inverted with respect to the original HDL description. In Design Compiler, the **compile_seqmap_enable_output_inversion** variable determines whether the compile command allows sequential elements to have their output phase inverted.

-level_sensitive
Limits the search to level-sensitive cells.

-edge_triggered
Limits the search to edge-triggered cells.

-master_slave
Limits search to master_slave cells.

DESCRIPTION

This command returns a collection of sequential cells or pins in the current design, filtered as specified by the options. By default, the command returns a collection of all sequential cells in the design. If you specify *clock_name*, it considers only the sequential cells in the transitive fanout of the sources of the clock. You can use one or more of the options **-cells**, **-data_pins**, **-clock_pins**, **-slave_clock_pins**, or **-output_pins** to return a collection containing the respective types of objects. For example, if you use **-data_pins**, the command returns a collection containing the only the data pins of the sequential cells that meet the search criteria. If you use both **-cells** and **-data_pins**, the command returns a collection containing both the sequential cells and their data pins.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example sets max_delay targets for timing paths leading to data pins of all registers clocked by *PHI2*.

```
prompt> set_max_delay 10.0 -to [all_registers -clock PHI2 -data_pins]
```


The following example returns a list of data pins for all master-slave registers clocked by *clockB*.

```
prompt> all_registers -master_slave -data_pins -clock clockB
```

The following example returns a list of all level-sensitive cells and their clock (enable) pins.

```
prompt> all_registers -level_sensitive -cells -clock_pins
```

The following example shows how to push into an instance named *U1* and find level-sensitive cells without searching subdesigns of that instance.

```
prompt> current_instance U1  
prompt> all_registers -no_hierarchy
```

SEE ALSO

```
create_clock(2)  
current_design(2)  
current_instance(2)  
remove_clock(2)  
set_max_delay(2)
```

all_rp_groups

Returns a collection of specified relative placement groups and all groups in their hierarchy.

SYNTAX

```
collection all_rp_groups  
[rp_groups]
```

Data Types

rp_groups list or collection

ARGUMENTS

rp_groups
Specifies the relative placement groups for which to search.
If you do not specify this argument, the command returns a collection containing all relative placement groups currently loaded in memory.

DESCRIPTION

The **all_rp_groups** command returns a collection of specified groups and all subgroups in their hierarchy. If you do not specify *rp_groups*, the command returns all relative placement groups currently loaded in memory. This command is supported only for designs that do not contain multiply-instantiated designs.

If no relative placement groups are found, the command returns an empty string.

Relative placement usage is available with Design Compiler Ultra.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses the **all_rp_groups** command:

```
prompt> get_rp_groups  
{mul::grp_mul ripple::grp_ripple example3::top_group}  
  
prompt> all_rp_groups  
{mul::grp_mul ripple::grp_ripple example3::top_group}  
  
prompt> all_rp_groups ripple::grp_ripple  
{ripple::grp_ripple mul::grp_mul}
```

```
prompt> remove_rp_groups -all -quiet  
1
```

```
prompt> all_rp_groups
```

SEE ALSO

```
add_to_rp_group(2)  
all_rp_hierarchicals(2)  
all_rp_inclusions(2)  
all_rp_instantiations(2)  
all_rp_references(2)  
create_rp_group(2)  
remove_rp_groups(2)
```

all_rp_hierarchicals

Returns a collection of hierarchical relative placement groups that contain specified groups in their hierarchy. The specified groups can be either included or instantiated in their parent group.

SYNTAX

```
collection all_rp_hierarchicals  
[rp_groups]
```

Data Types

rp_groups list or collection

ARGUMENTS

rp_groups
Specifies the relative placement groups whose ancestor groups are to be in the collection returned by this command. The specified groups can be either included or instantiated in their parent group.
If you do not specify this argument, this command returns a collection containing all hierarchical relative placement groups currently loaded in memory.

DESCRIPTION

The **all_rp_hierarchicals** command returns a collection of hierarchical groups that are ancestors of the relative placement groups specified in *rp_groups*. This command is supported only for designs that do not contain multiply-instantiated designs.

If you do not specify *rp_groups*, the command returns a collection containing all hierarchical relative placement groups currently loaded in memory.

If no relative placement groups are found, the command returns an empty string.

Relative placement usage is available with Design Compiler Ultra.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses the **all_rp_hierarchicals** command:

```
prompt> get_rp_groups  
{b::top a0::a0_group b::u_top/a1_group  
  b::u_top/a1_group_include b::u_nxt/a1_group
```

all_rp_hierarchicals

```

b::a1_group_include a1::a1_group}

prompt> all_rp_hierarchicals
{b::a1_group_include b::u_nxt/a1_group
  b::u_top/a1_group_include b::u_top/a1_group b::top}

prompt> all_rp_hierarchicals b::u_top/a1_group_include
{b::u_top/a1_group}

prompt> remove_rp_groups -all -quiet
1

prompt> all_rp_hierarchicals

```

SEE ALSO

```

add_to_rp_group(2)
all_rp_groups(2)
all_rp_inclusions(2)
all_rp_instantiations(2)
all_rp_references(2)
create_rp_group(2)
remove_rp_groups(2)

```

all_rp_inclusions

Returns a collection containing of hierarchical relative placement groups that include the specified groups.

SYNTAX

```
collection all_rp_inclusions  
[rp_groups]
```

Data Types

rp_groups list or collection

ARGUMENTS

rp_groups
Specifies the included relative placement groups whose parent groups are to be included in the collection returned by this command.
If you do not specify this argument, this command returns a collection containing all hierarchical relative placement groups in the design that contain included groups.

DESCRIPTION

The **all_rp_inclusions** command returns a collection containing the hierarchical groups that include the relative placement groups specified in *rp_groups*. If you do not specify *rp_groups*, this command returns a collection containing all hierarchical relative placement groups currently in memory that include other relative placement groups. This command is supported only for designs that do not contain multiply-instantiated designs.

If no relative placement groups are found, the command returns an empty string. Relative Placement usage is available with Design Compiler Ultra.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses the **all_rp_inclusions** command:

```
prompt> get_rp_groups  
{mul::grp_mul mul::big_grp ripple::grp_ripple  
  example3::g2 example3::top_group}  
  
prompt> all_rp_inclusions  
{mul::big_grp example3::g2}
```

```
prompt> all_rp_inclusions mul::grp_mul  
{mul::big_grp}
```

```
prompt> remove_rp_groups -all -quiet  
1
```

```
prompt> all_rp_inclusions
```

SEE ALSO

```
add_to_rp_group(2)  
all_rp_groups(2)  
all_rp_hierarchicals(2)  
all_rp_instantiations(2)  
all_rp_references(2)  
create_rp_group(2)  
remove_rp_groups(2)
```

all_rp_instantiations

Returns a collection of hierarchical relative placement groups that instantiate specified groups.

SYNTAX

```
collection all_rp_instantiations  
[rp_groups]
```

Data Types

rp_groups list or collection

ARGUMENTS

rp_groups
Specifies the instantiated relative placement groups whose parent groups are to be included in the collection returned by this command.
If you do not specify this argument, this command returns a collection containing all hierarchical relative placement groups in the design that contain instantiated groups.

DESCRIPTION

The **all_rp_instantiations** command returns a collection of hierarchical relative placement groups that instantiate the groups specified in *rp_groups*. If you do not specify *rp_groups*, this command returns a collection containing all hierarchical relative placement groups currently in memory that instantiate other relative placement groups. This command is supported only for designs that do not contain multiply-instantiated designs.

If no relative placement groups are found, the command returns an empty string.

Relative placement usage is available with Design Compiler Ultra.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses the **all_rp_instantiations** command:

```
prompt> get_rp_groups  
{mul::grp_mul ripple::grp_ripple example3::top_group}  
  
prompt> all_rp_instantiations  
{ripple::grp_ripple example3::top_group}
```



```
prompt> all_rp_instantiations ripple::grp_ripple  
{example3::top_group}
```

```
prompt> remove_rp_groups -all -quiet  
1
```

```
prompt> all_rp_instantiations
```

SEE ALSO

```
add_to_rp_group(2)  
all_rp_groups(2)  
all_rp_hierarchicals(2)  
all_rp_instantiations(2)  
all_rp_references(2)  
create_rp_group(2)  
remove_rp_groups(2)
```

all_rp_references

Returns a collection of relative placement groups that directly contain the specified cells, which are either leaf cells or hierarchical cells that contain instantiated relative placement groups.

SYNTAX

```
collection all_rp_references  
[cell_list]  
[-design design_name]
```

Data Types

<i>cell_list</i>	list
<i>design_name</i>	design

ARGUMENTS

cell_list
Specifies the cells (either leaf cells or hierarchical cells that contain instantiated relative placement groups) whose parent groups are to be included in the collection returned by this command.
If you do not specify this argument, this command returns a collection containing all relative placement groups that directly contain any leaf cells in the specified design.

-design *design_name*
Specifies the design in which to locate the specified cells.
If you do not specify this option, the tool looks for the cells in the current design.

DESCRIPTION

The **all_rp_references** command returns a collection of relative placement groups that directly contain the cells (either leaf cells or hierarchical cells that contain instantiated relative placement groups) specified in *cell_list*. This command is supported only for designs that do not contain multiply-instantiated designs.

If you do not specify *cell_list*, the command returns a collection containing all relative placement groups that directly contain any leaf cells in the specified design.

A group directly contains a leaf cell if the cell was added to the group using the **-leaf** option of the **add_to_rp_group** command.

A group directly contains a hierarchical cell if the cell was used to hierarchically instantiate another group using the **-instance** option the **add_to_rp_group** command.

Relative placement usage is available with Design Compiler Ultra.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses the **all_rp_references** command:

```
prompt> get_rp_groups
{mul::grp_mul ripple::grp_ripple example3::ripple
 example3::top_group}

prompt> all_rp_references
{mul::grp_mul ripple::grp_ripple example3::ripple}

prompt> all_rp_references U1 -design mul
{mul::grp_mul}

prompt> remove_rp_groups -all -quiet
1

prompt> all_rp_references
```

SEE ALSO

```
add_to_rp_group(2)
all_rp_groups(2)
all_rp_inclusions(2)
all_rp_instantiations(2)
all_rp_references(2)
create_rp_group(2)
remove_rp_groups(2)
```

all_scenarios

Lists all defined scenarios available in memory.

SYNTAX

string **all_scenarios**

ARGUMENTS

The **all_scenarios** command has no arguments.

DESCRIPTION

Displays the scenarios currently defined in memory. This list includes both active and inactive scenarios.

Multicorner-Multimode Support

This command uses information from both active and inactive scenarios.

EXAMPLES

The following example uses **all_scenario** to list the available scenarios.

```
prompt> create_scenario MODE1
prompt> create_scenario MODE2
prompt> all_scenarios
MODE1 MODE2
```

SEE ALSO

```
all_active_scenarios(2)
set_active_scenarios(2)
current_scenario(2)
remove_scenario(2)
create_scenario(2)
```

all_size_only_cells

Returns the collection of cells that have been specified size_only attribute on them.

SYNTAX

```
collection all_size_only_cells  
cell_list
```

Data Types

```
cell_list      list
```

ARGUMENTS

```
cell_list
```

Specifies the list of cells, from which size_only cells are returned. This can be a simple list or it can be an entire collection manipulation command, enclosed in square brackets [], that returns a collection of cells. By default, all cells in the design are considered.

DESCRIPTION

This command returns the collection of size only cells from the specified cell list. You can use wild cards, such as an asterisk (*) or question mark (?), when specifying the input cell list.

A standard Tcl collection is returned. All of the generic collection manipulating commands, such as **foreach_in_collection**, **sizeof_collection**, and so on, can be performed on this collection handle.

Multicorner-Multimode Support

This command uses information from active scenarios only.

EXAMPLES

```
prompt> all_size_only_cells  
{"U20", "U21", "U23", "SBLK/U1", "SBLK/U2"}
```

Following example shows the returned collection of size only cells in the *SBLK* subdesign. In this case, the command returns the collection specified by the **get_cells** command operating on *SBLK*.

```
prompt> all_size_only_cells [get_cells SBLK/*]  
{"SBLK/U1", "SBLK/U2"}
```

SEE ALSO

`set_size_only(2)`
`get_attribute(2)`
`foreach_in_collection(2)`
`get_cells(2)`
`set_attribute(2)`
`report_attribute(2)`
`sizeof_collection(2)`

all_spare_cells

Returns a collection of all spare cells in the design or from a list of objects in the input collection.

SYNTAX

```
collection all_spare_cells  
[input_coll]
```

Data Types

input_coll collection

ARGUMENTS

input_coll
Specifies the input collection to search and return the collection of spare cells. Objects are to be searched only from the content of the specified input collection, rather than from the design.

DESCRIPTION

The **all_spare_cells** command returns the collection of all spare cells in the entire design hierarchy (not just those in the top level of the design).

If you specify a value for the *input_coll* option, the tool searches for spare cells only in the collection this option identifies, rather than in the entire design.

The **all_spare_cells** command looks for an internal attribute placed earlier on existing cells during physical optimization or on additional cells at the end of physical optimization. (For more information, see the man page for the **insert_spare_cell** command.)

The tool returns a standard Tcl-language collection. You can perform all of the generic-collection manipulating commands, such as **foreach_in_collection**, **sizeof_collection**, and so on, on this collection.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example returns a collection of all spare cells in the design.

```
prompt> all_spare_cells  
{spare1 top/spare2 top/spare4 spare10}
```

The following example returns a collection of all spare cells from an existing

collection stored in \$COLL.

```
prompt> all_spare_cells $COLL  
{spare1 spare2}
```

SEE ALSO

```
foreach_in_collection(2)  
sizeof_collection(2)
```


all_threestate

Returns a collection of threestate cells or nets.

SYNTAX

```
collection all_threestate  
-nets  
[input_coll]
```

Data Types

input_coll collection

ARGUMENTS

-nets

Specifies that the collection of threestate nets is to be returned.

input_coll

Search the specified input collection and return a collection of threestate nets. Objects are to be searched only from the the content of the specified input collection rather than from the design.

DESCRIPTION

This command can return a collection of all threestate nets from the design. The tool searches the entire design hierarchy. It returns threestate nets from the entire design, not just from the top level.

If you specify a value for *input_coll*, the command searches for threestate nets only from the specified collection, rather than from the entire design.

Only those nets are returned as three state, which are driven by a pin which is tri-state or bi-directional.

If you do not specify the **-nets** option, the **all_threestate** command returns an error.

A standard Tcl collection is returned. All of the generic collection manipulating commands, such as **foreach_in_collection**, **sizeof_collection**, and so on, can be performed on this collection handle.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example returns a collection of all threestate nets from the design.

```
prompt> all_threestate -nets  
{ii2/hin iv3/in iv3/out tin ii2/hout tout}
```

SEE ALSO

```
all_fanout(2)  
foreach_in_collection(2)  
sizeof_collection(2)
```

all_tieoff_cells

Returns a collection of all tie-off cells in the current design or in the input collection.

SYNTAX

collection **all_tieoff_cells**

ARGUMENTS

The **all_tieoff_cells** command has no arguments.

DESCRIPTION

This command returns a collection of all tie-off cells from the current design. The tool searches the entire design hierarchy. It returns tie-off cells from the entire design, not just from the top level.

Tie-off cells are the constant cells that the tool introduced in the design. To find the value of a tie-off cell, the tool examines the corresponding lib_cell of each cell to determine if it is logic-0 or logic-1.

If you specify a value for *input_coll*, the command searches for tie-off cells or nets only from the specified collection, rather than from the entire design.

This command returns a standard Tcl collection. All of the generic collection manipulating commands, such as **foreach_in_collection**, **sizeof_collection**, and so on, can be performed on this collection.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example returns a collection of all tie-off cells from the design.

```
prompt> all_tieoff_cells
{logic1 logic0}
```

The following example returns a collection of all tie-off cells from an existing collection stored in \$COLL.

```
prompt> all_tieoff_cells $COLL
{logic1}
```

SEE ALSO

`foreach_in_collection(2)`

```
sizeof_collection(2)
```

allocate_fp_budgets

Performs proportional timing budgeting.

SYNTAX

allocate_fp_budgets

```
[-black_box_cells bb_cells_list]  
[-fixed_delay_objects objects]  
[-file_format_spec file_format_string]  
[-no_interblock_logic]  
[-cells budget_cell_names]  
[-incremental]  
[-advanced]  
[-no_split]  
[-create_qtm_models [-qtm_model_path output_directory]]
```

Data Types

<i>bb_cells_list</i>	list
<i>objects</i>	collection
<i>file_format_string</i>	string
<i>budget_cell_names</i>	list
<i>output_directory</i>	string

ARGUMENTS

- black_box_cells *bb_cells_list***
Specifies a list of black box cells to budget. The cells should have a timing model. By default, none of the black box cells are rebudgeted.
- fixed_delay_objects *objects***
Specifies a collection of objects that are implemented and their related delays do not change. The objects can include one or more soft macro cells, hard macro cells, black box cells, hierarchical cells for plan groups, pins of soft macros with ILMs, and pins of hierarchical cells for plan groups. The budgets allocated to them are equal to the current delay. By default, there are no fixed delay objects.
- file_format_spec *file_format_string***
Specifies the directory and naming style of the resulting Synopsys Design Constraints (SDC) files.
- no_interblock_logic**
Specifies that the top-level delays are considered fixed. There is no proportional budget allocation at the top level.
- cells *budget_cell_names***
Specifies a list of budget cell names. By default, all plan groups and soft macros in the current design are budgeted.
- incremental**
Performs budgeting in incremental mode. In incremental mode, budgeting

retrieves certain block implementation information from the ILMs of soft macros and budgets the delays using this information.

-advanced

Specifies that budgeting is to be based on virtual in-place optimization (IPO).

-no_split

Do not to split long lines. By default, SDC files created by the budgeter split long lines by using a backslash () symbol at the end of each line.

-create_qtm_models

Creates quick timing models (QTM) for the plan groups or soft macros that are budgeted. The delay arcs and the constraint arcs in the generated QTM represent the budgets on the plan groups or soft macros. You can load the QTMs for the soft macros or plan groups after they are committed to soft macros, to check timing at the top level and to verify the budgets. By default, quick timing models are not created.

-qtm_model_path *output_directory*

Specifies the path to write the quick timing model (QTM) files that are generated by the tool.

By default, the files are written to the current work directory. This option takes effect only when the **-create_qtm_models** option is specified.

DESCRIPTION

This command performs proportional timing budgeting for all plan groups or soft macros in the design. It prints out Synopsys Design Constraints (SDC) files, which contain timing constraints describing the budgets.

By default, the **allocate_fp_budgets** command disables recovery and removal arcs in a way that is similar to that of the **report_timing** and timing optimization commands. To enable recovery and removal arcs, set the **enable_recovery_removal_arcs** variable to true. This variable setting is honored by the **allocate_fp_budgets** command, the **report_timing** command, and the timing optimization commands, such as **place_opt**.

Multicorner-Multimode Support

This command uses information from active scenarios only.

EXAMPLES

The following example performs timing budgeting for a design with three blocks named BlockA, BlockB, and BlockC. It creates three SDC files in the current directory, named BlockA.sdc, BlockB.sdc, and BlockC.sdc, which contain the timing constraints for the corresponding block.

```
prompt> allocate_fp_budgets
```

The following example creates three files named BlockA.sdc, BlockB.sdc, and

allocate_fp_budgets

BlockC.sdc. In this case, the delay internal to the file named BlockA.sdc is considered to be fixed. This potentially affects all three generated files.

```
prompt> allocate_fp_budgets -fixed_delay_objects [get_cells BlockA]
```

SEE ALSO

```
check_fp_budget_result(2)  
check_fp_timing_environment(2)  
place_opt(2)  
report_timing(2)  
enable_recovery_removal_arcs(3)
```

analyze_fp_rail

Analyzes a complete or partial power network for voltage (IR) drop and electromigration (EM) on the specified power and ground nets.

SYNTAX

```
status analyze_fp_rail
-nets nets
[-power_budget power]
[-analyze_power]
[-voltage_supply voltage]
[-pad_masters pad_masters]
[-read_pad_instance_file file_name]
[-read_pad_master_file file_name]
[-use_pins_as_pads]
[-top_level_only]
[-create_virtual_rails layer]
[-ignore_blockages]
[-ignore_conn_view_layers layer]
[-read_power_compiler_file file_name]
[-read_prime_power_file file_name]
[-read_default_power_file file_name]
[-output_directory directory_name]
```

Data Types

<i>nets</i>	collection or list
<i>power</i>	float
<i>voltage</i>	float
<i>pad_masters</i>	string
<i>file_name</i>	string
<i>layer</i>	collection or list
<i>directory_name</i>	string

ARGUMENTS

-nets *nets*
Specifies the names of the power or ground nets on which to perform power network analysis. The default is to perform it on all power and ground nets.

-power_budget *power*
Specifies a total power budget for the analyzed design on which to perform constraint-driven power network analysis. The power budget is divided among the instances according to their sizes. For hard macros or standard cells, the power budget is computed based on the percentage of the total area. For a hierarchical block, the it is computed based on the sum of all cells and blocks inside it. The power unit is in milliwatts. The default is **1000**. For example, if the power budget for the entire design is 500 milliwatts, the total area of the top-level blocks is 500,000 square microns, and one of the blocks has an area of 25,000 square microns, power network analysis allocates the power for that 25,000-square-micron block as follows:

$500\text{mW} * (25,000/50,000) = 25\text{mW}$

-analyze_power

Specifies the power network analysis|synthesis (PNA|PNS) engine to analyze power for each instance from ICC. Before using this tcl option for power analysis, the default toggle rate (default as 0.1) and default static probability (default 0.5) for primary inputs and black box outputs can be set using **set power_default_toggle_rate** and **set power_default_static_probability** commands. The toggle rate and/or static probability for a specified object, e.g. a net, pin or port can be set using **set_switching_activity**. The switching activity interchange format (SAIF) file may also be read using **read_saif** command. **set_cell_internal_power** can be used to set the power value of a pin per toggle. Please refer to **report_power** command for more detailed information. **report_power -cell -flat -nosplit** is recommended to be run first prior to PNA with **-analyze_power**.

-analyze_power can be used concurrently with option **-power_budget**. If a power budget is specified using **-power_budget** together with **-analyze_power** and the power budget is larger than the sum of instance power calculated by **-analyze_power**, the difference between power budget and the total power by **-analyze_power** will be assigned to those instances that are not assigned in **-analyze_power** if there is any such instances. Otherwise, the power budget is ignored. Power budget is set to 0 by default with **"-analyze_power"**.

-analyze_power can also be used concurrently with **-read_default_power_file**. The power specified in the default power file has a higher priority than that calculated by **-analyze_power** option, i.e. if the power value is calculated and read from the default power file for one instance, the instance will be assigned with the power from the default power file.

"-analyze_power" is mutually exclusive with **"-read_power_compiler_file"** and **"-read_prime_power_file"**.

By default, power is NOT calculated from ICC unless this option is explicitly used.

-voltage_supply voltage

Specifies at the power pad the voltage of the net to be analyzed. The voltage unit is volts. The default is **1.5**.

-pad_masters pad_masters

Specifies the pad masters and the associated net as follows

net_name:pad_master_name

If net name is not specified, the pad master is used for both power and ground nets. By default, PNA will assume all the pads logically connected to signal **1** or **0** as power pads.

-read_pad_instance_file file_name

Specifies the pad instance file name and (optionally) the net name. If you do not specify the net name, the command assumes that the specified pad instance is used for both power and ground nets.

The *instance_and_net* format is as follows:

instance_name [net_name]

For example, a valid *instance_and_net* value might be **VDD1 VDD**. By default, this option is off.

-read_pad_master_file file_name

Specifies the pad master file name and (optionally) the net name. If you do

not specify the net name, the command assumes that the specified pad master is used for both power and ground nets. The *master_name_and_net* format is as follows:

master_name [*net_name*]

For example, a valid *master_name_and_net* value might be **VDD.FRAM VDD**. By default, this option is off.

-use_pins_as_pads

Specifies to regard pins as pads in block level simulation.

-top_level_only

Specifies for the power network analysis (PNA) engine to ignore cells inside soft macros, but to consider all the power and ground net wires and vias in the design. By default, the PNA engine analyzes the design as if it were flat.

-create_virtual_rails layer

Allows Power network analysis to create virtual pseudo straps that represent the metal 1 straps for the standard cell pin connections because, during floorplanning stages, metal 1 straps for the standard cell pin connections usually are not available. During power planning, Power network analysis then analyzes the IR (voltage) drop and electromigration to predict what the effects might be when the metal 1 straps are available. This option creates pseudo horizontal straps with specified layers for standard cell pin connections based on the row information in the database. By default, this option is off.

-ignore_blockages

Ignores blockages for virtual pads.
By default, this option is off.

-ignore_conn_view_layers layer

Ignores all metal layers (select the All option) or the specified metal layer(s) (select the Specified option) in connectivity (.CONN) views. By default, this option is off and none of metal layers are ignored.

-read_power_compiler_file file_name

Obtains the power calculation source from a Power Compiler report file. By default, this option is off.

-read_prime_power_file file_name

Obtains the power calculation source from a Prime Power/Prime Time PX report file. By default, this option is off.

-read_default_power_file file_name

Uses the PNA default format to obtain a power calculation. The *power_name_and_value* format is as follows, where *power_value* is given in mW:
instance_or_master_name [*power_value*]

For example, a valid *power_name_and_value* value might be as follows:

instance X1 0.2

master abc.FRAM 0.1

By default, the command also reads a power information file generated from AstroRail by using the **poDumpPowerInfo** command.

`-output_directory` *directory_name*

The specified directory stores the IR/EM results with the file name *design_name.net_name.pw_hl.pna* and the power allocation results with the file name *design_name.net_name.power*. The default is *pna_output*.

DESCRIPTION

This command allows user to perform a power network analysis (PNA) in a complete or partially built power/ground network at design planning stage. If the power ports in standard cells and/or hard macros are yet to be connected to the P/G network, the PNA will make virtual connection to those unconnected power ports and analyze the IR drop and EM in the power network based on user provided power budget and voltage supply.

Multicorner-Multimode Support

This command is not supported in a multi-scenario design flow.

EXAMPLES

The following example shows how to analyze the voltage drop on net *VDD* and *VSS* in the power network. The total power budget in the design is *1000mW*, the voltage supply is *1.2V*, and the pad master is *pvdi.FRAME* for net *VDD* and *pvdo.FRAME* for net *VSS*.

```
prompt> analyze_fp_rail\  
-nets {VDD VSS}\  
-power_budget 1000\  
-voltage_supply 1.2\  
-pad_masters "VDD:pvdi.FRAME VSS:pvdo.FRAME"
```

The following example shows how to analyze the voltage drop on net *VDD* in the power network. The power is analyzed by ICC and the voltage supply is *1.2V*.

```
prompt> analyze_fp_rail\  
-nets VDD\  
-analyze_power\  
-voltage_supply 1.2
```

SEE ALSO

`create_fp_virtual_pad(2)`
`load_fp_rail_map(2)`

analyze_fp_routing

Analyzes the existing global routing to determine where nets cross plan groups or voltage areas boundaries. Creates feedthroughs on plan groups or on hierarchical modules that belong to voltage areas. In the case of plan groups, it records the location, level and direction of the pins, including feedthrough pins, that are implied by the routing.

SYNTAX

```
int analyze_fp_routing  
[-preview_pin_feedthroughs {plan_groups | voltage_areas}]  
[-finalize_pins_feedthroughs {plan_groups | voltage_areas}]  
[-output_feedthroughs {nets | ports}]  
[-include_flip_chip_style_connections]  
[-minimum_feedthrough_length route_length]
```

ARGUMENTS

-preview_pin_feedthroughs {plan_groups | voltage_areas}
Specify the types of blocks on which feedthrough ports are expected to be created. Use this option to preview the list of future feedthrough nets. Feedthrough ports will be created where the global routes of these nets cross blocks of the specified types when **-finalize_pins_feedthroughs** is used in a future call to **analyze_fp_routing**. A file named "feedthrough_nets" containing the names of future feedthrough nets is generated. Refer to the **-output_feedthroughs** option for details on the content format of the "feedthrough_nets" file.
If a file named "feedthrough_nets" is already present in the current working directory a warning will be issued and the existing file overwritten. An error will be issued if it cannot be overwritten. This option can be used together with **-output_feedthroughs** but there is little point in doing that because "ports" given to **-output_feedthroughs** is ignored and the file "feedthrough_nets" is generated whether or not **-output_feedthroughs nets** is specified. This option is mutually exclusive with the **-finalize_pins_feedthroughs** option.

-finalize_pins_feedthroughs {plan_groups | voltage_areas}
Create feedthrough ports on logical module crossed by global routes This option causes **analyze_fp_routing** to insert feedthroughs into the logical hierarchy based on the existing global routing. If **plan_groups** is selected, pin locations for each crossing with a plan group are also calculated and saved. If **voltage_areas** is selected, feedthrough nets and ports are created for nets that have routing crossing a voltage area. For the **voltage_areas**, if the routing crosses a plan group which also belongs to a voltage area, the feedthrough is created on that plan group; otherwise, the command selects a hierarchical module within this voltage area and creates a feedthrough on it. A voltage area need not correspond to a single hierarchy module. If a voltage area corresponds to more than one hierarchical module, this command will create feedthroughs on the hierarchical modules which have connections to the processing net. Note that for **voltage_areas**, only logical feedthrough nets and ports are created. The physical location of the boundary crossing is not recorded. Those logical feedthrough ports are listed in a text file

"VA_Feedthroughs.port" under the current directory.

These two options can be combined if you want to create feedthroughs on both plan groups and voltage areas in one run.

-output_feedthroughs {nets | ports}

Specify the feedthrough reports to be generated. The acceptable values for this option are **nets** and **ports**. If **nets** is specified, a file named "feedthrough_nets" which lists all nets with feedthroughs are created. Note that the names are of flat nets that are still present on the top block. The feedthrough creation and net splitting has been done only in the logical netlist. Also listed with the nets are the corresponding blocks which the nets pass through. Therefore, an entry in **feedthrough_nets** has the following format:

net block1 block2

where this is a list of plan groups, voltage area names or a mixture of both, depending on how the **-finalize_pins_feedthroughs** or **-preview_pin_feedthroughs** options are specified.

If **ports** is specified, a number of feedthrough port files are created. A feedthrough port file is created for each block on which logical feedthrough ports have been created. It is named in the format *block.feedthrough_ports*, where *block* is the name of a plan group or voltage area. The feedthrough port file contains a list of created feedthrough ports on the named block. Related feedthrough ports are grouped together. The feedthrough ports are printed out in the format *block/port_name*. Note that **nets** and **ports** can be specified at the same time. In all cases, if an identically named file is already present in the current working directory, a warning is issued and the existing file is overwritten. An error is issued and the file is skipped if it cannot be overwritten; the command will neither terminate nor error out.

When used with the **-finalize_pins_feedthroughs** option, this option causes the **-analyze_fp_routing** command to output the names of created logical feedthrough nets across blocks or logical feedthrough ports on blocks into a text file or a set of text files. The types of blocks reported (plan groups, voltage areas, or both) depends on the block type specified using the **-finalize_pins_feedthroughs** option.

When used with **-preview_pin_feedthroughs**, the output is the same as if **-finalize_pins_feedthroughs** were used except for one important difference: the logical feedthrough ports and nets named in the output files have not been created. As **-preview_pin_feedthroughs** implies, the output simply provides a preview of the feedthrough nets and ports that will be created in the Milkyway database if the **-finalize_pins_feedthroughs** option is used. Because no new feedthrough ports are created in **-preview_pin_feedthroughs** mode, "ports" is not an applicable argument to **-output_feedthroughs** and will be silently ignored. However, the use of this option is redundant in this case because the file "feedthrough_nets" is generated even if this option is not used. Refer to the **-preview_pin_feedthroughs** option.

If this option is used alone, that is, not in conjunction with the **-preview_pin_feedthroughs** or **-finalize_pins_feedthroughs** options and the design has been previously finalized, "feedthrough_nets" and/or feedthrough port files are generated. This is useful if you neglected to generate these file when finalize was performed or they were unintentionally deleted. If the design has not been finalized, using this option alone is equivalent to using **analyze_fp_routing -preview_pin_feedthroughs plan_groups**.

-include_flip_chip_style_connections

This option specifies that the command also processes the nets that are

connected to top level flip-chip bump pins or I/O pins that overlap with voltage areas or plan groups. It will create feedthrough nets or ports as needed for those nets. The default behavior of **analyze_fp_routing** does not process the nets that are connected to flip-chip bump pins or the top level terminals that overlap with voltage areas or plan groups.

-minimum_feedthrough_length route_length

This option allows you to specify the minimum route length within a voltage areas boundary for feedthrough creation. It applies to voltage areas only and will be ignored on plan groups. By default, this value will be set to zero. When the route length within a voltage area boundary is less than the minimum feedthrough length, the **analyze_fp_routing** command will skip such routing and no logic feedthroughs will be created on the net. The **analyze_fp_routing** command will create feedthroughs only when the total feedthrough route length within a voltage area boundary is more than the minimum feedthrough length. This option applies to "pure" feedthroughs only, that is, it applies only to nets that have no logical connection to the voltage area. The dimensions of this value is microns.

DESCRIPTION

Use this command to analyze the global routing with respect to crossings of plan groups or voltage areas. This information can be used to output a list of feedthrough nets on plan groups and/or voltage areas. It can update the database by inserting feedthroughs into the logical hierarchy where plan group crossings do not have corresponding connections in hierarchy preservation and by saving the position of each boundary crossing as a guide to pin placement later. It can also update the database by inserting feedthroughs into hierarchical modules that belong to voltage areas when the router runs across a voltage area.

The pin locations on plan groups are used later during commit to generate a pin placement. Inserted feedthroughs on voltage areas can be used during buffer insertion to optimize timing.

The behavior of this command depends on pin constraints set previously with the **set_fp_pin_constraints** command and on voltage area constraints set previously with the **set_fp_voltage_area_constraints** command. It also depends on the last global route and the pin/voltage area constraints which controlled that global route. The nets that this command analyzes can be limited by a prior run of the **set_fp_pin_constraints** and **set_fp_voltage_area_constraints** commands. If **set_fp_pin_constraints** or **set_fp_voltage_area_constraints** was called prior to running this command and it specified a limited set of nets, then this command will analyze those nets only based upon the state of the last global route.

Use **-preview_pin_feedthroughs** to preview the nets that would have feedthroughs (across the block types specified by **-preview_pin_feedthroughs**) created within them. Use the **set_fp_pin_constraints** and **set_fp_voltage_area_constraints** commands to include or exclude those "previewed" nets and insert feedthroughs on the included nets with the **-finalize_pins_feedthroughs** option.

This command errors out if the **-preview_pin_feedthroughs**, **-finalize_pins_feedthroughs**, and **-output_feedthroughsoptions** are not used, because in this case the command performs no useful work.

EXAMPLES

The following example creates feedthroughs on plan groups if necessary and creates pin locations for plan groups for later pin assignment during commit. It also creates feedthroughs on hierarchical modules that belong to voltage areas, if needed, so that later buffer insertion can be done on those feedthroughs within the voltage areas. Feedthrough nets are also written to the text file "feedthrough_nets."

```
prompt> analyze_fp_routing -finalize_pins_feedthroughs {plan_groups voltage_areas} -  
output_feedthroughs nets
```

The following example allows the you to preview feedthrough nets and ports to be created on voltage areas. Previewed feedthrough nets are written to the text file "feedthrough_nets."

```
prompt> analyze_fp_routing -preview_pin_feedthroughs voltage_areas
```

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

SEE ALSO

```
set_fp_pin_constraints(2)  
set_fp_voltage_area_constraints(2)
```

analyze_rail

Performs the targeted rail analyses on the specified nets. This command generates the data needed to run PrimeRail and constructs a script that runs PrimeRail within the IC Compiler session. This command enables several target analyses, such as power and ground network integrity analysis, voltage drop analysis, and electromigration analysis. A previously generated PrimeRail command script can also be used to drive the rail analysis.

SYNTAX

```
status analyze_rail  
nets  
[-integrity]  
[-voltage_drop]  
[-decap]  
[-electromigration]  
[-inrush]  
[-primerail_script_file pr_script | -script_only]
```

Data Types

<i>nets</i>	collection or list
<i>pr_script</i>	string

ARGUMENTS

nets

Specifies the names of the power and ground nets to be analyzed. If the specified net is an internal power (ground) net, both its name and its supply voltage must be specified in brackets. For example: { VDD GND {inst/VDDV 1.08}}. Note that the voltage unit is volt. This is a required option.

-integrity

Performs connectivity checking on the specified power and ground nets. When enabled, the power and ground network integrity is reported in a Milkyway error view, and a resistivity map and parasitics map data are generated.

-voltage_drop

Performs voltage drop analysis on the specified power and ground nets. When enabled, a voltage drop map and power map data are generated, and voltage drop violations are reported in a Milkyway error view.

By default, the command performs voltage drop analysis if no specific analysis type is specified.

-decap

Performs decoupling capacitance analysis for the top design. This mode is available only during dynamic rail analysis. Therefore, you need to specify the following rail options with the **set_rail_options** command: -*filler_lib_cells*, -*decap_lib_cells*, and -*vd_threshold*. The first two options are required and -*vd_threshold* specifies the targeting voltage threshold. If the targeting voltage threshold is not set, the default target is achieve 10

analyze_rail

percent improvement of voltage drop.

These options can be set by using the **set_rail_options** command.

For example, If you want to perform decoupling capacitor insertion on the VDD net of a cell with filler cells, FillerCell1 and FillerCell2 and decoupling capacitor cells, DecapCell1 and DecapCell2. The voltage threshold target is above 1.5V (1.8V power supply), set the following rail options together with the dynamic rail analysis options:

```
set_rail_options -filler_lib_cells FillerCell1 FillerCell2 -decap_lib_cells
decapCell1 DecapCell2 -vd_threshold 1.5
```

Then run the **analyze_rail** command with the *-decap* option on the VDD net:

```
analyze_rail VDD -decap
```

The *-vd_threshold* option is optional if you want to use the default target setting, which reduces the maximum voltage drop by 10 percent.

When decoupling capacitor analysis is complete, a decap replacement report called *.decap_ude_N* is output in the PrimeRail run directory, where *N* is the number of decoupling capacitor analysis iterations. You can use `prompt> source report file name` to perform filler-decap replacement.

-electromigration

Performs electromigration analysis on the specified power and ground nets. When enabled, electromigration map data is generated and electromigration violations are reported in a Milkyway error view.

-inrush

Performs in-rush current analysis on the specified power and ground nets. When enabled, an in-rush current waveform file in the FSDB format will be generated and the leakage current, peak in-rush current and wake-up time will be reported the log file.

-primerail_script_file *pr_script*

Specifies and uses an existing PrimeRail script to perform the PrimeRail run. This script is usually generated from an earlier **analyze_rail** run.

When a user-specified script file is used, all other currently enabled options are ignored. For example, if you run the **analyze_rail -voltage_drop -primerail_script_file *myscript.cmd*** command, the *-voltage_drop* option is ignored.

This option and the *-script_only* option are mutually exclusive; you can specify only one.

-script_only

Generates the PrimeRail script for the target analyses, such as integrity, voltage drop or electromigration, without actually executing the analysis run. This allows you to modify and reuse the run script as desired.

This option and the *-primerail_script_file* option are mutually exclusive; you can specify only one.

DESCRIPTION

This command prepares the data needed for a PrimeRail rail analysis, and then calls PrimeRail to perform the rail analysis run. By default, the command performs voltage drop analysis on the specified power and ground nets. PrimeRail generates map data files that can be viewed to analyze the results. These map files are saved to the following directory at the end of the run: */output_dir/rail_map_data/current_mw_cell_net_name*. directory at the end of the run. PrimeRail also generates

an error view that documents issues found during the run. You can display the errors in the error browser, depending on the analysis options you chose.

Power and ground network integrity analysis checks for missing vias, insufficient vias, pin connections, and other related power and ground integrity, and saves the identified issues into a Milkyway error view. This analysis also calculates the minimum path resistivity on the target power and ground nets and saves the results into the map data files that can be viewed via the resistivity map.

Voltage drop analysis analyzes the voltage drop of the specified power and ground nets and saves the results into map data files that can be viewed in the voltage drop map. If a voltage drop threshold has been set with the **set_rail_options** command, threshold violations are reported to the error view.

Electromigration analysis calculates current density or pin currents and saves the results into the map data files that can be viewed in the electromigration map. Electromigration analysis also calculates the electromigration violations based on the default rules that are currently loaded in the Milkyway design library. The violations are saved to a Milkyway error view.

In-rush analysis calculates the in-rush current of the power management cells, then performs dynamic voltage drop analysis. In-rush analysis will ignore the -**analysis_mode** option and remains in dynamic mode. In-rush analysis outputs a waveform file in the FSDB format, an in-rush current summary in the log file, and a map file of the voltage drop analysis.

You can perform more than one target analysis, such as resistivity and voltage drop, in a single run.

When the **-analysis_mode** option is set to static in the **set_rail_options** command, the **analyze_rail** command will proceed with static voltage drop and electromigration analysis, and will generate:

- o Average voltage drop value in map file
- o Average electromigration value in map file
- o Average voltage drop error cell
- o Average electromigration error cell

When **-analysis_mode** option is set to dynamic in the **set_rail_options** command, the **analyze_rail** will proceed with dynamic voltage drop and electromigration analysis, and will generate:

- o Average and peak voltage drop values in map file
- o Average, peak and root mean square (RMS) electromigration values in map file
- o Peak voltage drop error cell
- o Average, peak and RMS electromigration error cells

EXAMPLE

The following example performs voltage drop analysis:

```
prompt> set_rail_options -default
prompt> analyze_rail {VDD VDDV}
```

analyze_rail

The following example performs power and ground network integrity analysis:

```
prompt> set_rail_options -user_defined_taps VDD_tap.txt  
prompt> analyze_rail VDD -integrity
```

The following example performs power and ground network integrity analysis and voltage drop analysis:

```
prompt> analyze_rail {VDD VSS} -integrity -voltage_drop
```

SEE ALSO

```
create_rail_setup(2)  
report_rail_options(2)  
set_rail_options(2)
```

analyze_subcircuit

Invokes circuit simulation on a subcircuit of the current design. This command is suitable for analyzing clock meshes.

SYNTAX

```
status analyze_subcircuit
[-name string]
[-to net_or_pins]
[-analysis_mode max | min | max_then_min]
[-effort low | medium | high]
[-simulator name_of_simulator]
[-spice_header_files list_of_files]
[-driver_subckt_files list_of_files]
[-starrcxt_map_file file_name]
[-configuration list_of_scenario_configurations]
[-output_directory directory]
[-input_rise_transition transition_time]
[-input_fall_transition transition_time]
[-period pulse_duration]
[-clock clock_name]
[-no_extraction]
[-from pins]
[-tie_high pins_or_nets]
[-tie_low pins_or_nets]
[-zero_resistance nets]
[-reset_annotation]
[-purge_all_annotations]
[-projected_loads load_configuration]
[-projected_drivers driver_configuration]
[-projected_input_skew input_skew_configuration]
[-probe_wires net]
[-verbose]
[-starrcxt_nxtgrd_file ]
```

Data Types

<i>string</i>	<i>name_of_simulator</i>
<i>net_or_pins</i>	collection of one net or multiple pins
<i>name_of_simulator</i>	string
<i>list_of_files</i>	list
<i>list_of_scenario_configurations</i>	list
<i>directory</i>	path
<i>transition_time</i>	float > 0.0
<i>clock_name</i>	string
<i>pins</i>	collection
<i>pins_or_nets</i>	collection
<i>nets</i>	collection
<i>load_configuration</i>	list
<i>driver_configuration</i>	list
<i>input_skew_configuration</i>	list

ARGUMENTS

`-name string`

Specifies the name of this subcircuit, which is used to distinguish it from other subcircuits in the same design. The string is used to construct both circuit elements and file names, so it must only contain alphanumeric characters and underscores. With that restriction, you can use any name, but if a clock is being analyzed you should use the name of the clock (if this is unique). Many files are generated based on this name.

`-to net_or_pins`

Specifies a collection of input pins that are driven by the circuit. If you specify a single net, the input pins associated with the net are included as load pins. (In earlier releases, the option was known as **-load.**)

`-analysis_mode max | min | max_then_min`

Specifies how buffer delays are calculated. If set to max, the delays for the buffers are calculated using their maximum delay value. If set to min, their minimum delay value is used. If set to max_then_min, maximum analysis is performed, followed by minimum analysis. You can use this option only if the TLUPlus specifies two corners.

`-effort low | medium | high`

Specifies the precision of the analysis done by the simulator. The effort setting is reformatted to suit the input syntax of the circuit simulator and sent as a parameter to the circuit simulation engine to control the precision of its analysis. The default is low.

`-simulator name_of_simulator`

Specifies which simulator to use. Only HSIM and NanoSim simulators are supported. The default is NanoSim.

`-spice_header_files list_of_files`

Specifies the locations of the spice device models of the transistors. Spice device models are needed for all the transistors. You specify the device models with the file path, and the file paths are copied into the simulator input. If you specify multiple files, they are included in the order specified. Note that file locations are relative to the current working directory (not relative to the temporary working directory, as was done in some earlier versions of mesh analysis).

`-driver_subckt_files list_of_files`

Specifies the locations of the subcircuit models of the buffers in the clock mesh tree. The circuit simulator requires a subcircuit model for all the buffers in the clock mesh tree. You specify the subcircuit models with the file path, and the file paths are copied into the simulator input. If you specify multiple files, they are included in the order specified. Note that file locations are relative to the current working directory.

`-starrcxt_map_file file_name`

Specifies the file name that contains the physical layer mapping information between the input database and the nxtgrd file. This file maps every TCAD process layer to a corresponding layer of layout database. This is a mandatory option if you want to use StarRC for extraction.

-configuration *list_of_scenario_configurations*

Specifies the SPICE settings for each scenario to be analyzed. The list can be of any length.

The format of the *list_of_scenario_configurations* argument is

```
-configuration {  
  {-scenario_name scenario  
   [-max_driver_subckt_files max_files]  
   [-max_spice_header_files header_max]  
   [-min_driver_subckt_files min_files]  
   [-min_spice_header_files header_min]  
  } ...  
}
```

The **-scenario_name** option is a required option and the specified name must be the same as the name used in the **create_scenario** command that created the scenario.

If you do not specify the **-max_driver_subckt_files** or **-**

min_driver_subckt_files option for a scenario, that scenario uses the files specified in the **-driver_subckt_files** command option. If you do not specify the **-max_spice_header_files** or **-min_spice_header_files** option for a scenario, that scenario uses the files specified in the **-spice_header_files** command option.

-output_directory *directory*

Specifies the directory used to contain the simulation driving files and the results of the simulation. If you do not specify this option, the **-name** option is used for the name of the output directory. If no directory by this name exists, one is created. Because many files are created in this directory, you must specify a directory with write permissions and sufficient disk space.

-input_rise_transition *transition_time*

Specifies the rise transition time for the root of the subcircuit. This value must be greater than zero. The parameter overrides any rise transition time otherwise observable in the circuit.

If you do not specify this option, the tool considers all the clocks found at the root of the subcircuit, and uses the slowest rise transition time.

-input_fall_transition *transition_time*

Specifies the fall transition time for the root of the subcircuit. This value must be greater than zero. The parameter overrides any fall transition time otherwise observable in the circuit.

If you do not specify this option, the tool considers all the clocks found at the root of the subcircuit, and uses the slowest fall transition time.

-period *pulse_duration*

Controls the waveform (width of the pulse) applied at the root node. This value must be greater than zero.

If you do not specify this option, the tool uses a pulse with half the width of the relevant clock period. In general, you should not have to change this option.

-clock *clock_name*

Specifies for which clock domain the analysis should be done. This especially helps to select the appropriate clock path when multiple clocks arrive at pins of a cell in the clock mesh circuitry.

-no_extraction

Controls whether the tool runs the **extract_rc** command to extract parasitics before writing the simulation driving files.

By default, the tool runs the **extract_rc** command. If your design already has current parasitic information written in the output directory from a previous run, you can suppress extraction to save time.

-from_pins

Specifies the startpoint to analyze a circuit.

If you do not specify this option, the command automatically determines a single driving point of a clock tree based on the load of the clock tree. If you design the tree from the loads up, specify the pins to be driven for timing purposes. You must specify this option if you are analyzing circuits with complex cells such as AND gates, multiplexers, and integrated clock-gating (ICG) cells. (In earlier releases, This option was known as **-inputs**.)

-tie_high_pins_or_nets

Holds the specified nets (or the nets attached to the specified pins) at logic one during simulation. If you specify a net, it is equivalent to specifying all the pins on that net.

This option is for advanced users only and might be useful for enable pins.

-tie_low_pins_or_nets

Holds the specified nets (or the nets attached to the specified pins) at logic zero during simulation. If you specify a net, it is equivalent to specifying all the pins on that net.

This option is for advanced users only and might be useful for disable pins.

-zero_resistance_nets

Sets the wire resistance for the specified nets to zero to perform "what if" analysis.

This option is for advanced users and can be used to estimate the amount of skew that has been introduced by wire resistance.

-reset_annotation

Removes the mesh timing annotation data from the previous **analyze_subcircuit** session from a single net.

You must use this option together with the **-to** option.

-purge_all_annotations

Removes the mesh timing annotation data from the previous **analyze_subcircuit** session from the entire design.

-projected_loads_load_configuration

Specifies additional loads that are to be included in the simulation in accordance with the configuration list. The configuration list defines projected load groups that are included in the simulation. You can define any number of projected load groups in the configuration list.

The format of the projected loads configuration list is

```
-projected_loads {  
    {-within rectangle  
        [-capacitance capacitance_value]  
        [-load_pin library_pin -number_of_loads positive_integer]  
        [-load_wire_capacitance capacitance_value]  
    }...
```

```
}
```

The **-within** option is required for each projected load group. It defines the mesh elements to which additional capacitance is added. Only mesh elements within the specified rectangle are in the load group.

You can specify the additional capacitance in the following ways:

- * Use the **-capacitance** option to specify a capacitance value.

This value is added to all mesh elements in the load group.

- * Use the **-load_pin** option to specify a load pin.

This is typically the name of a buffer or an integrated clock-gating cell (ICG). The tool multiplies the capacitance attribute of the specified pin by the value of the **-number_of_loads** option and adds this value to all mesh elements in the load group. The **-number_of_loads** option estimates the number of integrated clock-gating cells (ICGs) and buffer loads found within the specified region.

- * Use the **-load_wire_capacitance** option to specify a small capacitance value that is added to the value associated with each load pin to model the comb routing to the pin.

The specified capacitance is added to the load pin capacitance, which is calculated by adding the load pin capacitance to the wire capacitance times the number of loads. This capacitance is modeled as being distributed uniformly over the mesh straps in the specified region.

If you specify multiple projected load region statements, their effect is cumulative. That is, the capacitance at a point is impacted by the sum of all the projected load statements that overlap it.

When constructing the projected groups configuration for a hierarchical design, you can get the bounding box of a cell or soft macro from its **bbox** attribute. You can get the capacitance of a library pin, for your own reference, from its **capacitance** attribute.

For example, you can define two projected load regions with this syntax:

```
analyze_subcircuit -name testc ... -projected_loads {  
  {-within {{100.0 300.5} {800.0 440.6}}  
    -capacitance 0.3e-12  
    -load_pin tcbn90ghphvtwc/BUFFHVTD24  
    -load_wire_cap 0.001e-12  
    -number_of_loads 1000}  
  {-within {{1900.0 2300.5} {3800.0 2440.6}}  
    -load_pin tcbn90ghphvtwc/BUFFHVTD24  
    -number_of_loads 1000}  
}
```

-projected_drivers *driver_configuration*

Specifies additional drivers that are to be included in the simulation in accordance with the configuration list. The configuration list defines projected drivers that are included in the simulation. You can define any number of projected drivers in the configuration list.

To run **analyze_subcircuit**, your design must first have a mesh driver at each location where a driver might be needed. With this option, you can change the number and type of drivers seen by the simulation. This allows you to experiment with various drivers to get the set of drivers that is sufficient to drive your design reliably without overdriving the mesh and wasting power and area.

The configuration list defines the projected drivers that are included in the

simulation. You can define any number of projected drivers in the configuration list.

The format of the projected drivers configuration list is

```
-projected_drivers {  
    {-within rectangle | -nearest point  
    [-replicate integer]  
    [-projected_lib_cell lib_cell]  
    }...  
}
```

To identify the affected mesh drivers, you must specify either the **-within** option or **-nearest** option for each driver group. When you use the **-within** option, the specification affects the mesh drivers within the specified rectangle. When you use the **-nearest** option, the specification affects the mesh driver that is nearest to the specified point. The **-nearest** option might be effective for tuning the design based on the results of a previous analysis.

You can use the following options to specify additional drivers:

* **-replicate**

When you use this option, the simulation model includes the specified number of drivers in parallel for each affected mesh driver.

For example, if you specify **-replicate 4**, the SPICE simulation sees your original mesh driver plus three new ones in parallel. Note that these driver cells are not actually added to the design and they do not have locations or routing associated with them.

The only tool that sees them is the SPICE simulation.

You should not specify a value of 0. If you do, the SPICE simulation does not see the original cell; that is, part of the mesh that is not driven.

However, any routing parasitics associated with the comb route from its output pin are retained regardless of the value of this parameter. Because mesh drivers are typically close to the mesh and are therefore short and of low capacitance, this should not be a major inaccuracy.

* **-projected_lib_cell *lib_cell***

When you use this option, the simulation uses the specified library cell instead of the actual mesh driver.

If you specify multiple projected driver groups and their geometric regions overlap on a driver, they are processed in the order they appear, and only the final one impacts the driver. This can be used to specify a projected driver for the entire design in one large region and to specify additional driver sizing in a series of small regions.

For example, you can define two projected driver groups with this syntax:

```
analyze_subcircuit -name testc ... -projected_drivers {  
    {-within {{100.0 300.5} {800.0 440.6}} -replicate 4}  
    {-within {{1900.0 2300.5} {3800.0 2440.6}}  
    -projected_lib_cell tcbn90ghphvtwc/BUFFHVT24}  
}
```

-projected_input_skew *input_skew_configuration*

Specifies that simulation is to model skew on the input pins of the mesh drivers.

One of the major motivations for using a clock mesh is to reduce skew in the presence of on-chip variation (OCV) or other imperfections in a standard

clock tree. Because the premesh tree design flow generally produces a low-skew tree, to determine how effective the mesh would be at skew reduction is difficult. Because skew reduction is determined mainly by the width and pitch of the mesh, you are likely to create excess mesh straps unless you perform OCV analysis. You can use the **-projected_input_skew** option to calibrate the effectiveness of the mesh by simulating the effect of delay on the mesh driver inputs in one or more regions.

The worst case OCV probably occurs when a clock tree driver is close to the root of the premesh tree that has a very large additional delay, which might make half of what the chip sees as a large mesh driver input delay. If you consider this case too pessimistic, you might want to take only one quarter of the chip; that is, probably the quadrant with the largest load. Add delays to the mesh driver inputs in that quadrant by one typical gate delay.

The skew configuration list defines the delay to simulate on the mesh driver inputs in a region. You can define any number of input skew groups in the configuration list.

The format of the skew configuration list is

```
-projected_input_skew {
    {-within rectangle
     -rise_delay time
     -fall_delay time
     [-rise_transition time]
     [-fall_transition time]
    }...
}
```

The **-within** option is required for each input skew group. It defines the mesh drivers for which to model skew. Only mesh drivers within the specified rectangle are in the input skew group. If you specify multiple input skew groups, the specified geometric regions must not overlap.

You can specify the following delay values for each input skew group:

* **-rise_delay**

Delays the input waveform for the rising edge by the specified time. You must specify this value for each input skew group.

* **-fall_delay**

Delays the input waveform for the falling edge by the specified time. You must specify this value for each input skew group.

* **-rise_transition**

Specifies the rising transition time for the input waveform. If you do not specify this option, the simulation uses the rising transition time of the root of the tree.

* **-fall_transition**

Specifies the falling transition time for the input waveform. If you do not specify this option, the simulation uses the falling transition time of the root of the tree.

For example, if the mesh has an active area of 3000 by 4000 microns with the largest loads in the lower right quadrant and a gate delay is 0.025 ns, define the input skew group as follows:

```
analyze_subcircuit -name testc ... -projected_input_skew {
    {-within {{1500.0 0.0} {3000.0 2000}}
     -rise_delay 0.025 -fall_delay 0.020
     -rise_transition 0.1 -fall_transition 0.08}
}
```

-probe_wires net

Specifies that extraction is to insert probe points at regular intervals along the wires of the specified net, which is typically a clock mesh net. You can specify only a single net. These probe points allow analysis to separate the skew on the mesh from the skew on the loads, which is typically larger and more random, so each can be addressed and minimized appropriately. Circuit performance in a clock mesh is determined by the efficacy of the drivers, mesh straps, and the routing from the mesh straps to the loads. Sometimes it is useful to distinguish between the delays and skew on the mesh straps and the skew induced by the comb routing to the load pins. The **-probe_wires** option causes the extraction to insert probe points along the long wires associated with the specified net. Short wires will not have probe points added. The spacing between probe points is determined by the tool. After simulation, the tool prints out a summary of the maximum, minimum, and average latency and transition times observed on these probe points. This summary might be useful for tuning the distribution of mesh drivers and adjusting the widths of the mesh straps. Only one net may be so instrumented. You can use this option only when you use StarRC for extraction.

-verbose

Displays the extraction and circuit simulation messages to the console. By default, these messages are redirected into log files in the output directory.

-starrcxt_nxtgrd_file

Specifies the tcad grd file that contains conductor sheet resistance and models for 3-D parasitic capacitance calculation for StarRC. This is a mandatory option if you want to use StarRC for extraction. You need to specify one file for every scenario corner in the proper order in the list.

DESCRIPTION

Within IC Compiler, timing prediction is generally done with static timing analysis. But for some purposes, such as clock meshes, it is necessary or desirable to use circuit simulation. This command generates the necessary files to run a circuit simulator, invokes the simulator, and reads the output of simulation back into your design as annotations for delay and transition times.

By default, the **analyze_subcircuit** command extracts and simulates the circuit as it stands. You can also use this command to perform "what if" analysis to see how the mesh would behave with additional loads or different drivers. This is especially important in large hierarchical designs where the turnaround time of physical synthesis can be long and the need for upfront analysis is greater. For details about specifying additional loads, see the description for the **-projected_loads** option. For details about specifying additional drivers, see the description for the **-projected_drivers** option. For details about modeling input skew, see the description for the **-projected_input_skew** option.

FILES

Inputs required:

You must have a circuit-level model for each of the buffer gates in your clock tree,

and a transistor model for each of the transistors in the circuit models.

Files generated:

This command creates the *output directory* if it is not already present, and populates it with files for simulation and annotation. These include the circuit files, command files, and measurement files. One of these files is a shell script that is used to invoke the circuit simulator. Simulation produces many output and log files, which will also populate the directory. This command then processes those output files and generates an annotation file. This file can be sourced into your design, to update the delays and transition times of your subcircuit to match the circuit simulation results.

LIMITATIONS

The subcircuit to be analyzed is modeled as being driven by a single signal. That signal is applied at the unique point, determined automatically by traversing the circuit from the load pins, through any buffers, and stopping at an input port or a multiple input gate. The circuit to be simulated must not have any "side inputs." It does timing from one point to many.

If you want to analyze a set of mesh drivers whose inputs are not yet connected, you can do this by using the **-from** option and listing all the mesh driver input pins. For the purpose of simulation, these will all be driven in parallel, with no skew among them. This type of simulation is only for early exploration, before the clock tree "above" the mesh drivers has been designed.

Buffers that are "shorted together" in a mesh or other structure are supported. The resistance and capacitance of the mesh is determined by extraction, and is expressed in a parasitic input file with values obtained from the IC Compiler extractor or StarRC.

The entire subcircuit must be in the top level of the current Milkyway design. It must not extend into an ILM or subdesign.

The underlying circuit simulator is not case sensitive, so elements "buff_27" and "BUFF_27" are indistinguishable, and the design must not contain both names. For similar reasons, circuit elements should not contain the dot character ".".

NanoSim simulator will generally model coupling among the elements of the clock tree itself. The simulation of the tree accepts a transition time for the root gate. If the tree is driven by multiple clocks and the input signal has widely divergent transition times when operating in the various modes, this effect will not be modeled. The loads of the flip-flops are modeled by lump capacitance, not by active transistor models. If this operation is done before the signal nets have been routed, which would normally be the case, the final gate placement, routing geometry and detailed placement may not match the geometry in the final chip. There is no modeling of IR drop in the power or ground wires. The actual VSS and VDD used for each buffer gate is assumed to be static and is determined before simulation starts.

For more information on simulation refer to :

NanoSim User Guide
HSIM User Guide

Multicorner-Multimode Support

This command uses information from active scenarios only.

EXAMPLES

In the following example, a clock tree called `clk1` has been implemented with a set of specified library references to be used as buffers, driving a clock mesh net called `mesh_net`.

After this command has been run, the **report_timing** and **report_clock_tree** commands will reflect the delays and transitions observed in the circuit simulation.

```
prompt> analyze_subcircuit -name clk1 -to mesh_net \  
-output_directory clk_working \  
-spice_header_file original/models.sp \  
-driver_subckt_file {original/drivers.sp extra_cells.sp}
```

The following command uses HSIM simulator for analysis.

```
prompt> analyze_subcircuit -name clk1 -to mesh_net \  
-output_directory clk_working \  
-spice_header_file original/models.sp \  
-simulator hsim \  
-driver_subckt_file {original/drivers.sp extra_cells.sp}
```

In the following example, a clock tree called `clk2` has been implemented with a set of specified library references used as buffers driving a set of pins, but not using a mesh net.

```
prompt> set load_pins \  
[get_pins {addr_reg[27]/CP S45_reg[6]/CP S47_reg[13]/CP \  
S37_reg[10]/CP}]  
prompt> analyze_subcircuit -name addr -to $load_pins \  
-spice_header_file original/models.sp \  
-driver_subckt_file original/drivers.sp -effort high
```

If each scenario has a different SPICE file/header file, you should use **-configuration** as shown in the following example.

```
prompt> analyze_subcircuit -to clk \  
-driver_subckt_files max_spice_model \  
-spice_header_files header_file \  
-config {      {-scenario_name scenario1  
-max_driver_subckt_files max_spice_model_1  
-max_spice_header_file max_header_file_1}  
{-scenario_name scenario2 -max_spice_header_file max_header_file_2}  
}
```

SEE ALSO

```
create_clock_mesh(2)  
add_clock_drivers(2)  
compile_premesh_tree(2)
```

```
route_mesh_net(2)
create_scenario(2)
compile_clock_tree(2)
set_extraction_options(2)
report_clock_tree(2)
set_annotated_delay(2)
set_annotated_transition(2)
```

append_to_collection

Adds objects to the collection in the specified variable. The variable is updated.

SYNTAX

```
collection append_to_collection  
var_name  
object_spec  
[-unique]
```

Data Types

<i>var_name</i>	collection
<i>object_spec</i>	list

ARGUMENTS

var_name
Specifies a variable name. The objects matching *object_spec* are added into the collection referenced by this variable.

object_spec
Specifies a list of named objects or collections to add.

-unique
Indicates that duplicate objects are to be removed from the resulting collection. By default, duplicate objects are not removed.

DESCRIPTION

The **append_to_collection** command allows you to add elements to a collection. This command treats the variable name given by *var_name* as a collection, and appends all of the elements in *object_spec* to that collection. If the variable does not exist, it is created as a collection with elements from *object_spec* as its value. If the variable does exist, and it does not contain a collection, it is an error.

The result of the command is the collection which was initially referenced by *var_name*, or the collection created if the variable did not exist.

The **append_to_collection** command is provides the same semantics as a common use of **add_to_collection** but with significant improvement in performance. An example of replacing **add_to_collection** with **append_to_collection** is given below.

```
set var_name [add_to_collection $var_name $objs]
```

Using **add_to_collection** this becomes:

```
append_to_collection var_name $objs
```

The **append_to_collection** command can be much more efficient than **add_to_collection** if you are building up a collection in a loop. The arguments of the command have the same restrictions as the **add_to_collection** command. Please see the **add_to_collection** man page for more information about those restrictions.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example from IC Compiler shows how a collection can be built up with **append_to_collection**:

```
prompt> set xports
Error: can't read "xports": no such variable
      Use error_info for more info. (CMD-013)
prompt> append_to_collection xports [get_ports in*]
{"in0", "in1", "in2"}
prompt> append_to_collection xports CLOCK
{"in0", "in1", "in2", "CLOCK"}
```

SEE ALSO

```
add_to_collection(2)
foreach_in_collection(2)
index_collection(2)
remove_from_collection(2)
sizeof_collection(2)
```


apropos

Searches the command database for a pattern.

SYNTAX

```
string apropos  
[-symbols_only]  
pattern
```

Data Types

pattern string

ARGUMENTS

-symbols_only
 Searches only command and option names.

pattern
 Searches for the specified *pattern*

DESCRIPTION

The **apropos** command searches the command and option database for all commands that contain *pattern*. The *pattern* argument can include the wildcard characters "*" and "?". The search is case-sensitive. For each command that matches the search criteria, the command help is printed as though **help -verbose** was used with the command.

Whereas **help** looks only at command names, **apropos** looks at command names, the command one-line description, option names, and option value-help strings. The search can be restricted to only command and option names with the *-symbols_only* option.

When searching for dash options, do not include the leading dash. Search only for the name.

EXAMPLES

In the following example, assume that the **get_cells** and **get_designs** commands have the **-exact** option. Note that without the *-symbols_only* option, the first search picks up commands which have the string "exact" in the one-line description.

```
prompt> apropos exact  
get_cells                    # Create a collection of cells  
  [-exact]                    (Wildcards are considered as plain characters)  
  patterns                   (Match cell names against patterns)  
  
get_designs                  # Create a collection of designs  
  [-exact]                   (Wildcards are considered as plain characters)  
  patterns                   (Match design names against patterns)
```

```
real_time          # Return the exact time of day

prompt> apropos exact -symbols_only
get_cells          # Create a collection of cells
  [-exact]          (Wildcards are considered as plain characters)
  patterns          (Match cell names against patterns)

get_designs        # Create a collection of designs
  [-exact]          (Wildcards are considered as plain characters)
  patterns          (Match design names against patterns)
```

SEE ALSO

```
help(2)
```

archive_design

Archives the given design(s) to a new location.

SYNTAX

```
status archive_design
-source library_path
-design design_names
-archive archive_directory
[-complete]
[-overwrite]
```

Data Types

<i>library_path</i>	string
<i>design_names</i>	string
<i>archive_directory</i>	string

ARGUMENTS

-source *library_path*
The path to the Milkyway library that contain the design(s) of interest.

-design *design_names*
The Milkyway designs to archive. To archive multiple designs, enclose the design names in quotation marks delimited by the space character, ' '. To archive all designs in the main library, pass "*" as the argument. By default, the latest version of the design is archived, but a specific version may be specified e.g., "top_design;3"

-archive *archive_directory*
The location of the new archive directory.

-complete
By default, archive_design takes a minimalistic approach and will only archive designs in reference libraries if they are needed by the design(s) being archived. If this option is used, even designs in referenced libraries that are not used will be copied over to the archive location. This allows the archived design to be further optimized later if needed.

DESCRIPTION

This command will copy the given Milkyway design, logic libraries, TLU+ files, and create a design_setup.tcl file in the given archive directory. The purpose is to extract one or more designs of interest from a library containing many designs, leaving you with a smaller, self-contained set of files for that design.

For the Milkyway design, archive_design will copy the latest version (unless otherwise specified) of all views of the design(s) being archived, and any subdesigns that are referenced directly or indirectly by these design(s). Note that the LM view is not archived. If -complete is specified, then all designs contained

in archived reference libraries will be copied. The main library and all needed reference libraries will each be placed in separate directories under the design/ subdirectory. The library references maintained in the archived libraries are updated to reflect the new locations of the reference libraries in the archive location.

The logic .db libraries that are copied come from the variables **search_path**, **target_library**, and **link_library**. **target_library** and **link_library** contain a list of logic libraries. Minimum libraries set using the command **set_min_library** will also be copied. If the variables are not set or the files not found, then it is not copied to the archive location. Logic .db libraries will be stored under the logic_db/ subdirectory.

TLU+ files that are archived are defined with the **set_tlu_plus_files** command. If no values have been specified or the files not found, then the TLU plus file will not be copied. TLU+ files will be stored under the tlu_plus/ subdirectory.

The design_setup.tcl file contains variable initializations that can be sourced by the user. It contains the values of application variables from the current session similar to the output of **write_app_var**. Variable settings in the design_setup.tcl file may be modified from the current session values to reflect the new archive location. This file will be stored in the root directory of the archive location.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example is a typical usage of **archive_design**:

```
prompt> source ~/my_designs/ORCA/orig_data/design_setup.tcl
prompt> archive_design -source ~/my_designs/ORCA/design -
design "icc_top_route icc_top_place" -archive my_archive
```

SEE ALSO

```
set_tlu_plus_files(2)
search_path(3)
target_library(3)
link_library(3)
set_min_library(2)
write_app_var(2)
```

assign_flip_chip_nets

Creates or reconnects nets between flip-chip drivers to bumps.

SYNTAX

```
status assign_flip_chip_nets
[-personality_type personality_type_list]
[-prefix prefix]
[-uniquify num_to_uniquify]
[-terminal_layers layer_name_or_collection]
[-terminal_names terminal_name_list]
[-eco]
[-multiple_terminal_pin pattern]
```

Data Types

<i>personality_type_list</i>	list
<i>prefix</i>	string
<i>num_to_uniquify</i>	integer
<i>layer_name_or_collection</i>	collection
<i>terminal_name_list</i>	list

ARGUMENTS

-personality_type *personality_type_list*
Specifies a list of personality types for matching the flip-chip drivers and bumps. Only drivers and bumps with identical personalities are matched. Each personality type is a string name.
By default, the tool processes all personality types.

-prefix *prefix*
Specifies a prefix string name. Use this option when a new net must be generated to connect a flip-chip driver and a bump. The default prefix is "fcNet".

-uniquify *num_to_uniquify*
Specifies whether to and how to uniquify the driver nets inside a driver group. A driver group is multiple drivers whose flip-chip ports are connected by the same net. The *num_to_uniquify* argument accepts integers from -n to 0 to n.
n means n-to-1 uniquify and to assign one bump for every n flip-chip driver port.
1 means 1-to-1 uniquify and to assign one bump for each flip-chip driver port. New nets will be created with the original net name followed by "_#". For example, VDD net will be uniquified to VDD, VDD_1, VDD_2, ...
0 means "do not uniquify" and to assign only one bump for the multiple flip-chip driver ports. The original driver net is reused.
-n means 1-to-n connection and to assign n bumps for each flip-chip driver port.
The default behavior is 1-to-1 uniquify.
To form a 1-to-n connection, first assign_flip_chip_nets with "-uniquify 1", then assign_flip_chip_nets again with "-uniquify -n."

-terminal_layers *layer_name_or_collection*
 Specifies the layers to use in the multiple terminal pin assignment. Only the terminals on the given layers will be assigned a bump, the rest will not be taken into assignment. This option is only effective when used together with the **-multiple_terminal_pins** option. By default, all terminals of the multiple terminal pins will be assigned a bump. Specify the layer either by using the **get_layers** command or by specifying the name of the layer.

-terminal_names *terminal_name_list*
 Specifies specific terminals of flip-chip driver pins to be assigned. This option cannot be used together with the **-multiple_terminal_pins** or **-terminal_layers** options. Each terminal name must be specified in a "cell_instance_name/pin_name:terminal_name" format. Pattern matching with wildcards is not supported. If this option is used, each specified terminal will be counted as an individual driver pin. The flip-chip driver pins that do not occur in the list will be assigned as usual.

-eco
 Assigns flip-chip nets only for unconnected drivers or bumps or both, and keeps the existing flip-chip connections.

DESCRIPTION

Based on the current placement results, you can use this command to locate the nearest unassigned bumps, match them to flip-chip drivers using the shortest wire length method, and then automatically assign new logical nets or reconnect the nets between the flip-chip bumps and drivers. According to the **-uniquify** style you specify, the matching can be 1-to-1 or n-to-1 (n drivers to 1 bump) or 1-to-n (1 driver to n bumps).

A flip chip driver pin can have multiple terminals. If each of the terminals needs one bump to be assigned, use the **-multiple_terminal_pins** option to specify the pin list. If only some of the terminals on specific layers need to be assigned, use the **-terminal_layers** option to specify which layers to use. If only some specific terminals need to be assigned, use the **-terminal_names** option. By default, one bump will be assigned to each pin.

For bumps with an existing logical connection, the previous connection to the bump is removed and a new bump is assigned. The newly matched bump is reconnected to the matching flip-chip driver. The new bump net inherits the net name from the flip-chip driver pad to which the bump is now connected. A new net is created only if the flip-chip driver pad has no existing net connection.

The driver-to-bump matching is processed by personality types previously defined by the **set_flip_chip_type** command. Only drivers and bumps with identical personality types are matched. The tool does not process unspecified personality types if you use the **-personality_type** option.

Prerequisites

Use the **set_attribute** command to specify a port of the flip-chip driver cell as the designated flip-chip port. This identifies the port that is used to connect to a bump.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example performs driver-to-bump matching of the personality types "red" and "yellow".

```
prompt> assign_flip_chip_nets -personality_type {"red" "yellow"}
```

The following example performs 3-to-1 driver-to-bump matching

[illegible]

```
prompt> report_flip_chip_driver_bump
BUMP_CELL NET DRIVER_CELL(S)
"BUMP1" "vdd_1" "VDDPST_1" "VDDPST_2" "VDDPST_3"
"BUMP2" "vdd_2" "VDDPST_4" "VDDPST_5" "VDDPST_6"
"BUMP3" "vdd" "VDDPST_7" "VDDPST_8"
```

The following example performs multiple-terminal-aware assignment

```
prompt> assign_flip_chip_nets -personality_type "A" -multiple_terminal_pin DI/*  
Matching 4 A flip chip driver ports to 16 bumps using Munkres algorithm...  
Initializing Munkres Matrix...  
Solving Munkres Matrix...  
>>>>>>>>>>>>>>>>>>>>>  
Storing the matching result...  
Clearing up Munkres...
```

```
Reporting the multiple terminal pin matching property on bump cell
AREAPAD_S_120
[0]      pin name: DI/VSS_TEST      terminal name: VSS_TEST 3
```

```
Reporting the multiple terminal pin matching property on bump cell
AREAPAD_S_121
[0]      pin name: DI/VSS TEST      terminal name: VSS_TEST 2
```

Reporting the multiple terminal pin matching property on bump cell
AREAPAD \$ 63

Reporting the multiple terminal pin matching property on bump cell
AREAPAD_S_64

The following example performs multiple-terminal-aware assignment by specifying terminal names

SEE ALSO

```
assign_flip_chip_nets
166
```


balance_inter_clock_delay

Performs interclock delay balancing.

SYNTAX

```
status balance_inter_clock_delay  
[-clock_trees list_of_clocks]  
[-max_target_delay float_value]  
[-operating_condition min | max | min_max]
```

Data Types

<i>list_of_clocks</i>	list
<i>float_value</i>	float

ARGUMENTS

- clock_trees *list_of_clocks***
Performs interclock delay balance on the clocks specified in the *list_of_clocks*. When this option is omitted and no balance group defined, the balancing is performed on all clocks in the design. When this option is omitted and any balance group defined through `set_inter_clock_delay_options`, the balancing is performed on clocks defined in balance groups.
- max_target_delay *float_value***
Specifies the maximum target delay using a positive float value. The tool balances all of the clocks' longest paths up to the **-max_target_delay** value if no other higher priority options are specified.
- operating_condition min | max | min_max**
Sets the operating condition to reflect the worst case (such as long delays, low voltage, or high temperatures). This option can be combined with another operating condition. If no operating condition option is specified (min, max or min_max) is set, the default is **-max**.

DESCRIPTION

The **balance_inter_clock_delay** command performs interclock delay balance by inserting delay cells at the root of the clock to meet the clock insertion delay requirements. If timing violations occur due to an unbalanced clock insertion delay in a different clock domain, set a clock delay requirement with a small longest path delay value and run the **balance_inter_clock_delay** command. This minimizes the differences between the longest paths among the clocks.

Multicorner-Multimode Support

This command uses information from the clock tree synthesis scenario.

EXAMPLES

The following command performs interclock delay balance using the maximum target delay options:

```
prompt> balance_inter_clock_delay\  
-max_target_delay 0.5
```

The following command performs interclock delay balance for the *CLK1* clock tree to 0.5ns and *CLK2* to 0.8ns, by using the **set_inter_clock_delay_options** command and the **balance_inter_clock_delay** command:

```
prompt> set_inter_clock_delay_options\  
-target_delay_clock CLK1\  
-target_delay_value 0.5  
prompt> set_inter_clock_delay_options\  
-target_delay_clock CLK2\  
-target_delay_value 0.8
```

```
prompt> balance_inter_clock_delay
```

The following command performs interclock delay balance for the *CLK1* clock tree lags *CLK2* by 0.5ns, using the **set_inter_clock_delay_options** command and the **balance_inter_clock_delay** command:

```
prompt> set_inter_clock_delay_options\  
-delay_offset 0.5\  
-offset_to CLK1 -offset_from CLK2
```

```
prompt> balance_inter_clock_delay
```

The following command performs interclock delay balance only on *CLK4* and *CLK5* under the worst and the best operating conditions, even though the design may have *CLK1*, *CLK2*, and *CLK3* clocks:

```
prompt> balance_inter_clock_delay\  
-clock_trees [get_clocks {CLK4 CLK5}]\  
-operating_condition min_max
```

SEE ALSO

set_inter_clock_delay_options(2)

calculate_caa_based_yield2db

Calculate critical area analysis based yield and add it into db library.

SYNTAX

```
status calculate_caa_based_yield2db  
library_name/db_file_name  
-particle_distr_func_file file_name]  
[-data_kit_type tsmc | tsmc_encr]  
[-layer_alias_dsd_format {x_y_z_r_t}]  
[-output db_file_name]
```

ARGUMENTS

-particle_distr_func_file file_name
Specifies the particle distribution function file. User can define ShrinkFactor in this file for half-node support. The shrinkage factor impacts the total critical area in Poisson equation, thus impact to the cell yield. If ShrinkFactor is not specified, default to 1. It is mandatory.

-data_kit_type tsmc | tsmc_encr
Specifies the data kit type. Currently, it only support tsmc or tsmc_encr. If the particle distribution function file from tsmc is plain text format, use the tsmc data kit type, if it's in encrypted format, please use the tsmc_encr type. Default is tsmc_encr.

-layer_alias_dsd_format {x_y_z_r_t}
The -layer_alias_dsd_format option is needed when there are multiple routing layers defined in the library and CAA tables. It specifies the metal scheme that will be used for the tapeout. The order is {Mx My Mz Mt Mr}. For example, {5 2 1 0 0} means that M2, M3, M4, M5, and M6 are Mx layers; M7 and M8 are My; and M9 is Mz. For more details about the metal scheme, consult the TSMC manual.

-output db_file_name
Specifies the db file name to be written out, it will contains the CAA based yield data directly if the calculation finished successfully. It's optional when the input is a library_name, as users can choose using write_lib to write out the CAA yield db. It becomes required when the input is a .db file

DESCRIPTION

Command **calculate_caa_based_yield2db** calculate CAA based yield, using critical_area_table defined in input library, particle distribution function, routing layer alias dsd format, then add cell yield model to library db file. This data can be used to improve yield in many stages of the whole design flow by various methods, such as yield based cell swapping, optimization, etc.

The cell yield data is stored as cell attribute, by the format of "failure rate", from the Poisson equation, we can know their relationship: $\text{Cell\_CA\_Yield} = \exp(-\text{Cell\_FR})$; Cell failure rate can be reported by "report_lib yield <library_name>", if

the library is generated from .lib.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example calculates CAA based yield for library read from "*test.lib*" , and write out "*yield.db*" file which contains yield data.

```
prompt> read_lib test.lib
prompt> calculate_caa_based_yield2db test -
particle_distr_func_file ddk_1.txt \
-data_kit_type tsmc caa_yield -layer_alias_dsd_format {1 2 1 2 1}
prompt> write_lib test -output yield.db
prompt> report_lib -yield test
```

SEE ALSO

report_critical_area(2)

calculate_hier_antenna_property

Calculates the hierarchical antenna properties of a hard macro.

SYNTAX

```
integer calculate_hier_antenna_property  
-diffusion_layers {list_of_layers}  
[-top_cell_pin_only true | false]  
[-treat_source_drain_as_diodes true | false]  
[-report_diodes true | false]  
[-poly_layers {list_of_layers}]  
[-contact_layers {list_of_layers}]  
[-poly_text_layers {list_of_layers}]  
[-m1_text_layers {list_of_layers}]  
[-m2_text_layers {list_of_layers}]  
[-m3_text_layers {list_of_layers}]  
[-m4_text_layers {list_of_layers}]  
[-m5_text_layers {list_of_layers}]  
[-m6_text_layers {list_of_layers}]  
[-m7_text_layers {list_of_layers}]  
[-m8_text_layers {list_of_layers}]  
[-m9_text_layers {list_of_layers}]  
[-m10_text_layers {list_of_layers}]  
[-m11_text_layers {list_of_layers}]  
[-m12_text_layers {list_of_layers}]  
[-m13_text_layers {list_of_layers}]  
[-m14_text_layers {list_of_layers}]  
[-m15_text_layers {list_of_layers}]  
[-gate_class1_marking_layers {list_of_layers}]  
[-gate_class2_marking_layers {list_of_layers}]  
[-gate_class3_marking_layers {list_of_layers}]
```

Data Types

list_of_layers list

ARGUMENTS

- diffusion_layers {*list_of_layers*}
Specifies a list of the layer[:datatype] or layer names for all diffusion layers. This option is required.
- top_cell_pin_only true | false
Extracts hierarchical antenna properties for only nets that connects to pins in the top cell of the macro (the default is true). When you disable this option, the command extracts hierarchical antenna properties for nets that connect pins in the top cell of the macro and pins in the child cells contained within the macro.
- treat_source_drain_as_diodes true | false
Treats MOS source and drain as protection diodes, and considers any MOS source and drain regions that are connected to a net to be a part of the diode

protection area available to that net. The default is true.

-report_diodes true | false

Reports the location of generated protection diodes. The default is true.

-poly_layers {*list_of_layers*}

Specifies a list of the layer[:datatype] or layer names for all poly layers.

-contact_layers {*list_of_layers*}

Specifies a list of the layer[:datatype] or layer names for all contact layers (connecting poly to metal1).

-poly_text_layers {*list_of_layers*}

Specifies a list of the layer[:datatype] or layer names for the poly text layer.

-m1_text_layers {*list_of_layers*}

Specifies a list of the layer[:datatype] or layer names for the metal1 text layer.

-m2_text_layers {*list_of_layers*}

Specifies a list of the layer[:datatype] or layer names for the metal2 text layer.

-m3_text_layers {*list_of_layers*}

Specifies a list of the layer[:datatype] or layer names for the metal3 text layer.

-m4_text_layers {*list_of_layers*}

Specifies a list of the layer[:datatype] or layer names for the metal4 text layer.

-m5_text_layers {*list_of_layers*}

Specifies a list of the layer[:datatype] or layer names for the metal5 text layer.

-m6_text_layers {*list_of_layers*}

Specifies a list of the layer[:datatype] or layer names for the metal6 text layer.

-m7_text_layers {*list_of_layers*}

Specifies a list of the layer[:datatype] or layer names for the metal7 text layer.

-m8_text_layers {*list_of_layers*}

Specifies a list of the layer[:datatype] or layer names for the metal8 text layer.

-m9_text_layers {*list_of_layers*}

Specifies a list of the layer[:datatype] or layer names for the metal9 text layer.

-m10_text_layers {*list_of_layers*}

Specifies a list of the layer[:datatype] or layer names for the metal10 text layer.

calculate_hier_antenna_property

```

-metal1_text_layers {list_of_layers}
    Specifies a list of the layer[:datatype] or layer names for the metal1 text
    layer.

-metal2_text_layers {list_of_layers}
    Specifies a list of the layer[:datatype] or layer names for the metal2 text
    layer.

-metal3_text_layers {list_of_layers}
    Specifies a list of the layer[:datatype] or layer names for the metal3 text
    layer.

-metal4_text_layers {list_of_layers}
    Specifies a list of the layer[:datatype] or layer names for the metal4 text
    layer.

-metal5_text_layers {list_of_layers}
    Specifies a list of the layer[:datatype] or layer names for the metal5 text
    layer.

-gate_class1_marking_layers {list_of_layers}
    Specifies a list of the marking layer[:datatype] or layer names for the gate
    thickness class1.

-gate_class2_marking_layers {list_of_layers}
    Specifies a list of the marking layer[:datatype] or layer names for the gate
    thickness class2.

-gate_class3_marking_layers {list_of_layers}
    Specifies a list of the marking layer[:datatype] or layer names for the gate
    thickness class3.

```

DESCRIPTION

This command invokes Hercules on an auto-generated runset to calculate the hierarchical antenna properties of the current design (hard macro). The hierarchical antenna properties include gate size, routing area, and diode protection for all the ports of the macros. These properties can be used by Zroute and the classic router to check antenna rules on nets connected to the macro pins.

Before this command is used, the library and CEL view must be opened. The hierarchical antenna properties are stored in the FRAM view of the cell. Both the CEL and FRAM views of the cell must exist before you execute this command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example runs the command to extract the hierarchical antenna properties of a hard macro with gate class1 and class2 marking layers.

```
prompt> calculate_hier_antenna_property \  
-diffusion_layers {2 5} -poly_layers {7} \  
-contact_layers {14} -poly_text_layers {7:12} \  
-m1_text_layers {15:12} -m2_text_layers {17:12} \  
-m3_text_layers {19:12} -m4_text_layers {21:12} \  
-m5_text_layers {96:12} -gate_class1_marking_layers {12:12} \  
-gate_class2_marking_layers {24:12}
```

SEE ALSO

```
extract_hier_antenna_property(2)  
extract_zrt_hier_antenna_property(2)
```


cd

Changes the current directory.

SYNTAX

```
int cd  
[directory]
```

Data Types

directory string

ARGUMENTS

directory
Any existing directory name.

DESCRIPTION

Changes the current directory to *directory*. If you specify no argument, your login or home directory becomes the new current directory. By default, the current directory is the directory where `dc_shell` or the Design Analyzer is invoked.

A file specification of `dot` (`.`) is shorthand for the current directory. Dot is commonly included in the **search_path** variable. Changing the current directory changes the interpretation of `"."` in the **search_path** variable.

Changing the current directory in the Design Analyzer does not affect the current directory of the your UNIX environment.

Note: The **sh** command cannot be used to change directories.

EXAMPLES

```
prompt> cd /usr/designer
```

```
prompt> pwd  
/usr/designer
```

```
prompt> cd joe
```

```
prompt> pwd  
/usr/designer/joe
```

```
prompt> cd ../bob
```

```
prompt> pwd  
/usr/designer/bob
```

```
prompt> cd ~
```

```

prompt> pwd
/usr/designer/joe

prompt> cd ~doug

prompt> pwd
/usr/designer/doug

prompt> cd

prompt> pwd
/usr/designer/joe

prompt> cd /usr/designer

prompt> ls
bob          designs          doug
joe          one.db          two.db

prompt> cd designs

prompt> ls
one.db          other.db

prompt> search_path = ". ~"

prompt> read {one.db, two.db}
Loading db file '/usr/designer/designs/one.db'
Loading db file '/usr/designer/joe/two.db'

prompt> cd /tmp

prompt> read one.db
Loading db file '/usr/designer/joe/one.db'

```

SEE ALSO

`pwd(2)`

change_fp_soft_macro_to_black_box

Converts the specified soft macros to black boxes.

SYNTAX

```
status change_fp_soft_macro_to_black_box  
black_boxes
```

ARGUMENTS

```
black_boxes  
    Specifies the soft macros to convert.
```

DESCRIPTION

Converts the specified soft macros to black boxes (the original soft macro is removed and replaced with an empty one).

All instantiated subdesigns and leaf cells in the specified soft macro are removed from the logical (hierarchy preservation) information in the database.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example converts the soft macro alu1 to a black box.

```
prompt> change_fp_soft_macro_to_black_box [get_cell alu1]
```

SEE ALSO

```
estimate_fp_black_boxes(2)  
get_cells(2)
```

change_link

Changes the design to which a cell is linked.

SYNTAX

```
status change_link  
object_list  
design_name  
[-view view_name]  
[-freeze_silicon]
```

Data Types

<i>object_list</i>	list
<i>design_name</i>	string
<i>view_name</i>	string

ARGUMENTS

object_list
Specifies the cells or references in the current design for which the link is to be changed.

design_name
Specifies the name of the design to which the cells or references in *object_list* are to be linked.

-view *view_name*
Specifies the Milkyway view of the reference design to which the cells or references in *object_list* are to be linked.

-freeze_silicon
Supports the ECO freeze silicon flow. The cell instances will be in the transitional state after the command is executed. The **place_freeze_silicon** or **map_freeze_silicon** command will swap the cell instances with the available spare cells of the specified reference library cell.

DESCRIPTION

The **change_link** command specifies a design for which the link for a cell or reference is to be changed. If a cell is specified, it is changed to be one occurrence of the specified design. If a reference is specified, all cells of the given reference type are changed to be occurrences of the design.

The cell or reference link can only be changed to a compatible design. For example, the design must have the same number of ports with the same name and direction as the cell or reference.

This command can also be used to change the reference of a physical-only cell to another physical-only library cell.

Physical-only cells and standard cells are not compatible, so you can only change a physical-only cell's reference to another physical-only library cell. Similarly, references of standard cells can only be changed to logical library cells.

The **-view** option allows the reference to be changed to a Milkyway view other than the FRAM view. This option must be used carefully, as minimal checking is done when it is used. This option also triggers a complete link of the design, even if it is already linked.

Although the *design_name* argument accepts names in the format *library/library_cell*, it does not imply that the actual library cell used for the new cell will be from the specified library. The actual library cell used is determined by the current link library settings.

During **change_link** activity, all Synopsys database format link information is copied from the old design to the new design. If the old design is a synthetic module, all attributes of the old synthetic module are moved to the new link. After running the **change_link** command, link the design with the **link -force** command.

EXAMPLES

The following example shows the use of **change_link** to change the reference of the U1 cell to BUFX3 from INVX3:

```
prompt> get_attribute [get_cells U1] ref_name
INVX3
1

prompt> change_link [get_cells U1] BUFX3
1

prompt> get_attribute [get_cells U1] ref_name
BUFX3
1
```

SEE ALSO

```
link(2)
map_freeze_silicon(2)
place_freeze_silicon(2)
```

change_macro_view

Changes the view of the macro that is used.

SYNTAX

```
status change_macro_view  
-reference cell_reference_name  
-view view_name  
[-quiet]
```

Data Types

<i>cell_reference_name</i>	string
<i>view_name</i>	string

ARGUMENTS

```
-reference cell_reference_name  
    Specifies the cell reference name for which we want to change the reference  
    view.  
    All instances of the cell reference will use the view specified by -view  
    view_name.  
  
-view view_name  
    Specifies the view name that will be used.  
    Valid values: FRAM and CEL.  
  
-quiet  
    Suppresses the messages.
```

DESCRIPTION

This command changes the macro's view. All instances of the cell reference are affected by this command.

The command returns a status indicating success or failure.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example replaces the existing CEL view with the FRAM view for the mem_ctrl macro.

```
prompt> change_macro_view -reference mem_ctrl -view FRAM
```

The following example defines that the CEL view is to be used for the mem256x16

reference cell.

```
prompt> change_macro_view -reference mem256x16 -view CEL
```

SEE ALSO

`create_macro_fram(2)`

change_names

Changes the names of ports, cells, and nets in a design.

SYNTAX

```
integer change_names
[-rules name_rules]
[-hierarchy]
[-verbose]
[-names_file names_file]
[-log_changes log_file]
[-restore]
[-dont_touch object_list]
[-instance instance]
[-new_name new_name]
```

Data Types

<i>name_rules</i>	string
<i>names_file</i>	string
<i>log_file</i>	string
<i>object_list</i>	list
<i>instance</i>	string or instance object
<i>new_name</i>	string

ARGUMENTS

-rules *name_rules*
Specifies a name rule set that details the rules for modifying names to which the object names conform. The *name_rules* file is defined by using the **define_name_rules** command. By default, this value is the *name_rules* file specified by the **default_name_rules** variable. The tool ignores the **-rules** option if you specify the **-names_file** option.

-hierarchy
Specifies that all names in the design hierarchy are to be modified. By default, the tool changes only objects in the *current design*.

-verbose
Specifies that every name change is to be reported. By default, the tool provides only a summary detailing the number of name changes in the design.

-names_file *names_file*
Specifies a file of manual name changes for designs in dc_shell. By default, the command automatically makes name changes. If you specify this option, the tool ignores the **-rules** option.

-log_changes *log_file*
Specifies the log file in which to cumulatively record all name changes to the specified file. If the tool executes multiple **change_names** commands, the *log_file* must be empty before the first **change_names** command is executed.

-restore

Reverses the changes recorded in the *names_file* during the current session and restores the contents of the file to the state it was in when opened.

-dont_touch *object_list*

Specifies the designs on which the **change_names** command is not to be applied. By default, there is no design in the list. Any design under the hierarchy of the *current design* can be added to the list to be labeled *dont_touch*, ensuring that **change_names** does not apply any changes to them. Name "." is considered to be the *current design*.

-instance *instance*

Specifies the instance on which the **change_names** command is to be applied. This option must be used with **-new_name** to specify a new instance name. This option is mutually exclusive with all other options except **-new_name** and **-verbose**.

-new_name *new_name*

Specifies the new instance name when **-instance** option is used. The new name must consist of only alphanumeric characters and the underscore (_), and must start with an alphabetical character. The new name cannot be a reserved word using by Verilog, System Verilog, or VHDL. The new name should not conflict with existing instance, port, and net name within the same design. The leading backslash will be removed from the new name.

DESCRIPTION

Changes the names of ports, cells, and nets in a design to conform to specified name rules. This command cannot be used to change the names of library cells. Names are changed so that they are unique within the design, but only if they do not conform to the specified rules. Names are also changed to enforce the bus member to use the same base name as its owning bus. Use the **report_names** command to show the effects of executing this command without actually making the changes.

Note that if you examine the **report_names** report and do not want the names of bus members to be changed to use the same base name as the owning bus, set the **change_names_dont_change_bus_members** environment variable to **true** and run the command again.

There are two primary reasons for using the **change_names** command:

- It enables you to modify design object names in the tool so that the names match those that are ultimately created for a design by the **write** command. The names the tool displays in reports and in other information match those used in your target system.
- It enables you to define naming rules specific to your target system that might not be included in a **write** command format. For example, you might be using VHDL as a design transfer mechanism, but the naming rules of your system might be more restrictive than those supported by the true VHDL format.

To obtain a list of available name rules, use the **report_name_rules** command. For information on naming rules that can be affected during the execution of the **change_names** file, see the **define_name_rules** man page.

When you execute **change_names** with no options specified, it operates on ports, cells, and nets in the *current design*. When you specify the **-hierarchy** option, changes are expanded to include all design objects within the *current design* object hierarchy. Specifying the **-names_file** option overrides all other options. The *names_file* option contains a mapping of user-defined name changes. These name changes are not subject to any name rules, but an error is reported if any name changes are nonunique.

The names file specified by *names_file* in the **-names_file** option has one format without a delimiter and another format with a semicolon (;) as a delimiter. These formats are described below.

Names File Without a Delimiter

This format matches the format of the **report_names** command output. Thus, the output of **report_names** can be redirected to a file, edited, and then reused as input to the **change_names** command.

If you are creating this format of the names file, all information above the report bar formed by the dashed line (-----) is ignored. If no report bar is present in the file, the file is parsed. After the bar, each line of the file is parsed into four fields. The fields must be separated by whitespace characters, such as blanks, tabs, or new lines. The fields are as follows:

```
design_name    object_type    object_name    new_name
```

Names File With a Delimiter

Create a names file with a delimiter (;) using the methodology described in the **Names File Without Delimiter** section above, and adding the following line above the report bar formed by the dashed line (-----):

```
use delimiter:true
```

The fields must be separated by whitespace characters, such as blanks, tabs, or new lines. The fields are as follows:

```
design_name    object_type    object_name    new_name ;
```

The *design_name* is a design currently read into the tool. The *object_type* is a port, cell, or net. The *object_name* is the name of an object within the specified *design_name*. The *new_name* is the name designated to replace *object_name*.

Following these rules and assuming that *TOP* is the current design, the example names file can be reformatted and still be valid.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows a names file without a delimiter:

```
*****
Report : names
        -rules MY_RULES
Design : TOP
Version: v3.0
Date   : Tue Aug 13 14:24:23 2007
*****
```

Design	Type	Object	New Name
TOP	cell	U\$1	U_1
TOP	net	NET_NAME_IS_WAY_TOO_LONG	NET_NAME_IS_WAY_TOO
TOP	net	12345	N12345

The following example shows a names file with a delimiter:

use delimiter:true

Design	Type	Object	New Name
TOP	cell	U\$1	U_1 ;
TOP	net	NET_NAME_IS_WAY_TOO_LONG	NET_NAME_IS_WAY_TOO ;
TOP	net	12345	N12345 ;

```
TOP cell U$1 U_1
. net NET_NAME_IS_WAY_TOO_LONG NET_NAME_IS_WAY_TOO
TOP net 12345 N12345
```

The following example shows how to change the names in the *current design* to conform to the rules defined by the **default_name_rules** variable:

```
prompt> list default_name_rules
default_name_rules = "EXAMPLE"
```

```
prompt> change_names
Information: Using name rules 'EXAMPLE'.
Information: 4 names changed in design 'TOP'.
```

The following example shows how to use the **-verbose** option to show each name changed by the **change_names** command.

```
prompt> change_names -verbose
Information: Using name rules 'EXAMPLE'.
```

Design	Type	Object	New Name

TOP	port	A1	a_
TOP	port	A2	a_a
TOP	port	A3	a_b
TOP	port	A4	a_c

The following example shows how to use the **-hierarchy** option to change names throughout the design hierarchy. Typically, you want to change the names of all designs in the hierarchy to conform to your name rules.

```
prompt> change_names -hierarchy
Information: Using name rules 'EXAMPLE'.
Information: 12 names changed in design 'HALF_ADDER'.
Information: 35 names changed in design 'SUBTRACTOR'.
Information: No names changed in design 'TOP'.
```

The following example shows how to use the **define_name_rules** command to create a set of simple name rules. In this example, the name rules require that all names use uppercase characters or numerals.

```
prompt> define_name_rules CAPS_ONLY -allow "A-Z 0-9"
```

```
prompt> change_names -rules CAPS_ONLY -verbose
Information: Using name rules 'CAPS_ONLY'.
```

Design	Type	Object	New Name
MY_BUFFER	port	a1	A1
MY_BUFFER	port	a2	A2
MY_BUFFER	port	b1	B1
MY_BUFFER	port	b2	B2
MY_BUFFER	cell	u1	U1
MY_BUFFER	cell	u2	U2

The following example shows how to make manual name changes to design objects by using the **report_names** and **change_names** commands. First, a report of all of the original design objects is redirected to a file. This file is then edited to make manual changes. Finally, the edited names file is used to direct name changes made by **change_names**. In this case, name rules are ignored and your manual changes are given precedence.

```
prompt> report_names -original > TOP.names
prompt> /* Edit the names file TOP.names here */
prompt> change_names -names_file TOP.names
Information: 15 names changed using names file 'TOP.names'
```

The following example shows how to use the **-instance** option to change one instance name.

```
prompt> change_names -instance a1 -new_name a2
```

SEE ALSO

`define_name_rules(2)`
`report_name_rules(2)`
`report_names(2)`
`default_name_rules(3)`

change_selection

Changes the selection in the GUI, taking a collection of objects and changing the selection according to the type of change specified.

SYNTAX

```
int change_selection
[-name slct_bus]
[-replace]
[-add]
[-remove]
[-toggle]
[-type object_type]
[-clock_trees clock_tree_list]
collection
```

Data Types

<i>slct_bus</i>	string
<i>object_type</i>	list
<i>clock_tree_list</i>	list
<i>collection</i>	list

ARGUMENTS

- name *slct_bus***
Specifies to change the selection bus by using the value of *slct_bus*. By default, the command changes the selection bus by using the name *global*.
- replace**
Replaces the current selection with the objects in the collection. This is the default behavior of the command if you do not specify any optional parameter.
- add**
Adds the objects in the collection to the current selection. By default, this option is off.
- remove**
Removes the objects that are specified in the collection from the current selection. By default, this option is off.
- toggle**
Adds each item that is specified in the collection to the selection bus if it is not currently contained there. If it is currently contained in the selection bus, remove it. By default, this option is off.
- type *object_type***
Specifies the type to change. Only those items from the collection that are of the type specified by *object_type* are used to change the selection. The valid values are **design**, **port**, **net**, **cell**, **pin**, and **path** (timing path). By default, the command uses the entire collection.

`-clock_trees clock_tree_list`

Specifies a list of clock trees for changing the selection. By default, this option is off.

`collection`

Specifies the collection of objects to use to change the selection. The type of change that is applied to the current selection with the *collection* is specified by the optional parameters listed above. By default, this option is off.

DESCRIPTION

The **change_selection** command changes the selection in the GUI. When selections are changed, the GUI updates all relevant windows to reflect it.

A collection of objects and the type of change are given as input to the command. The collection of objects might be returned as the result of another command such as the **get_designs** command. If the collection is empty and you use the **-replace** option (or let the command default by specifying no option), the current selection is cleared.

If you use the **-type** option, only the type of objects specified are used to change the current selection. For example, if you use **-type design**, the command uses only the design objects in the collection to change the current selection. If you do not use **-type** option, all objects in the collection are used to change the current selection. For example, if you use the **-add** option without using the **-type** option, all objects, regardless of their type, are added to the current selection.

For information about collections, see the **collections** man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example replaces the current selection with the collection of design objects, regardless of type.

```
prompt> change_selection [get_designs]
```

The following example adds a cell named U5 to the selection made in the example above.

```
prompt> change_selection -add [get_cells U5]
```

The following example removes all pin objects from the current selection.

```
prompt> change_selection -remove -type pin [get_selection]
```

The following example clears the selection.

```
prompt> change_selection ""
```

The following example creates a new selection bus and adds a net named n1 to the selection.

```
prompt> set slct [create_selection_bus]  
prompt> change_selection -name $slct [get_nets 1]
```

SEE ALSO

```
collections(2)  
filter_collection(2)  
get_selection(2)  
query_objects(2)
```


change_tie_connection

Changes tie connection of the specified signal pins and ports.

SYNTAX

```
status change_tie_connection  
[-net net]  
-port port_list  
-pin pin_list
```

Data Types

<i>net</i>	list
<i>port_list</i>	list
<i>pin_list</i>	list

ARGUMENTS

-net *net*

Specifies the net to connect. The net must be a power or ground net, a scalar (single bit) net, and must exist in the current design.

This argument is optional. If no net is provided, and the pins or ports are currently connected to a PG net, they will be disconnected.

-port *port_list*

Specifies a list of signal ports to which the PG net is connected.

A port can be at top level or any other hierarchical level. Use "-port" to change the connection for a hierarchical port inside the module.

For example, "-port ma/mb/Q" means Q's connection will be changed inside hierarchical scope "ma/mb". While, "-pin ma/mb/Q" means Q's connection will be changed inside hierarchical scope "ma".

If a specified port is already connected to a signal net, the tool issues an error message and reminds the user to use `disconnect_net` to disconnect first. If a specified port is already connected to a PG net, the tool automatically reconnects the port.

This argument is optional. If not specified, "-pin" argument must be presented.

-pin *pin_list*

Specifies a list of signal pins to which the PG net is connected.

If a specified pin is already connected to a signal net, the tool issues an error message and reminds the user to use `disconnect_net` to disconnect first. If a specified pin is already connected to a PG net, the tool automatically reconnects the pin.

This argument is optional. If not specified, "-port" argument must be presented.

DESCRIPTION

This command changes tie connection on a list of signal pins and signal ports to the specified PG nets. PG nets include nets specified by **mw_logic1_net** and **mw_logic0_net**

variables.

Both "-port" and "-pin" can be specified at the same time. At least one of the two must be specified.

The net should be at top level or be at the same hierarchy level as the pins or ports. A net can be connected to many pins or ports; however, you cannot connect a pin or port to more than one net.

To disconnect objects on a net, do not use the **-net** option.

To display pins and ports on a net, use **all_connected** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses the **change_tie_connection** to change signal pin U7/A connection from tie low to tie high. The **all_connected** command is used to verify the Milkyway net connection before and after the command.

```
prompt> all_connected [get_pin U7/A]
{VSS}
prompt> change_tie_connection -net VDD -pin [get_pin U7/A]
Connecting power net 'VDD' to pin 'U7/A'.
prompt> all_connected [get_pin U7/A]
{VDD}
```

The following example shows the error message generated when you attempt to disconnect a PG pin.

```
prompt> change_tie_connection -pin U7/VDD
Error: PG leaf pin is not supported by this command. (UIED-65)
```

SEE ALSO

```
all_connected(2)
create_net(2)
current_design(2)
connect_net(2)
disconnect_net(2)
remove_net(2)
```

change_tie_connection

characterize

Captures information about the environment of specific cell instances, and assigns the information as attributes on the design to which the cells are linked.

SYNTAX

```
int characterize  
cell_list  
[-no_timing] [-constraints]  
[-connections] [-power]  
[-verbose]
```

Data Types

cell_list list

ARGUMENTS

cell_list
A list of cells in the current design whose designs are to be characterized.

-no_timing
Indicates not to characterize the timing characteristics of the design. Attributes representing the timing environment or requirements will not be placed on the subdesigns.

-constraints
Indicates to place area, power, connection class, and design rule constraint information on the subdesigns.

-connections
Indicates to characterize the connection attributes of the design.

-power
Indicates to use the switching activity information of the cells in *cell_list* to annotate the corresponding subdesigns.

-verbose
Indicates to echo equivalent commands that are applied to the subdesign being characterized.

DESCRIPTION

This command places on a design the information and attributes that characterize its environment in the context of a specified instantiation in another design.

The primary purpose of **characterize** is to capture the timing environment of the subdesign. This happens if **characterize** is used with no arguments, and if **-constraint**, **-connection**, or **-power** is used.

The command derives and asserts the following on the design to which the instance is

linked:

Unless **-no_timing** is specified, **characterize** places on the subdesigns any timing characteristics previously set by the following commands:

```
create_clock
group_path
read_sdf
read_clusters
set_annotated_check
set_annotated_delay
set_auto_ideal_net
set_disable_timing
set_drive
set_driving_cell
set_false_path
set_ideal_latency
set_ideal_net
set_ideal_transition
set_input_delay
set_load
set_max_delay
set_min_delay
set_multicycle_path
set_operating_conditions
set_output_delay
set_resistance
set_timing_ranges
set_wire_load_model
set_wire_load_mode
set_wire_load_selection_group
set_wire_load_min_block_size
```

If **-constraint** is specified, **characterize** places on the subdesigns any area, power, connection class, and design rule constraints previously set by the following commands:

```
set_max_area
set_max_power
set_dont_touch_network
set_ideal_network
set_ideal_net
set_connection_class
set_fanout_load
set_max_capacitance
set_max_fanout
set_min_porosity
set_fanout_load
set_max_transition
set_min_capacitance
set_cell_degradation
set_fix_multiple_port_nets
```

If **-connections** is specified, **characterize** places on the subdesigns any connection

attributes previously set by the following commands:

```
set_equal
set_opposite
set_logic_one
set_logic_zero
set_logic_dc
set_unconnected
```

NOTE: connection_class information is applied only if **-constraint** is used.

If **-power** is specified, **characterize** places on the subdesigns any switching activity information, toggle rates, and static probability previously set, calculated, or saved by the following commands:

```
report_power
set_switching_activity
```

In most cases, characterizing a design removes the effects of any previous characterization and replaces all the relevant information. However, in the case of back-annotation, (**set_load**, **set_resistance**, **read_sdf**, **set_annotated_delay**, **set_annotated_check**), the annotations are moved during the characterize step, and cannot overwrite existing annotations made on the subdesign. In this case, annotations must be explicitly removed from the subdesign (using **reset_design**) before running **characterize** the next time.

characterize can be used with **set_dont_touch** to maintain hierarchy during optimization. This method of optimization is known as bottom-up optimization. It can be applied using a golden instance or a uniquify approach. An alternative to bottom-up optimization is top-down optimization, otherwise known as hierarchical compile. In top-down optimization, the tool performs characterization and optimization of subdesigns automatically.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example shows the golden instance approach to bottom-up optimization. This is useful when the same design is used in several places, with a similar environment. **characterize** and **set_dont_touch** are used to optimize the subdesign once and reference that optimized design in several places. The characterization is done using a golden instance (U1 in this case) and reflects the environment of that instance. Assume that U1 and U2 both reference design ADDER and have similar environments.

```
Characterize the ADDER design with respect to instance "U1" and compile it.
prompt> current_design TOP
prompt> characterize U1
prompt> current_design ADDER
```

```
prompt> compile
```

Compile the top level while preserving "U1" and "U2".

```
prompt> current_design TOP  
prompt> set_dont_touch {U1 U2}  
prompt> compile
```

The following example shows the unify approach to bottom-up optimization. If a design is used in more than one place with different environments, use the **unify** command to create a design for each instance. You can now characterize the new designs and optimize each separately.

Unify the hierarchy so that all hierarchical cells reference different designs. This creates new design names such as ADDER_1 and ADDER_2.

```
prompt> current_design TOP  
prompt> unify
```

Characterize the unique designs with respect to their environments, and compile each.

```
prompt> characterize {U1 U2}  
prompt> current_design ADDER_1  
prompt> compile  
prompt> current_design ADDER_2  
prompt> compile
```

Compile the top-level design while preserving U1 and U2.

```
prompt> current_design TOP  
prompt> set_dont_touch {U1 U2}  
prompt> compile
```

The following example shows top-down optimization. In top-down optimization, the tool implicitly performs characterization for each subdesign.

```
prompt> current_design TOP  
prompt> unify  
prompt> compile
```

SEE ALSO

```
create_clock(2)  
current_design(2)  
group_path(2)  
link(2)  
read_sdf(2)  
remove_annotated_check(2)  
remove_annotated_delay(2)  
reset_design(2)  
set_annotated_check(2)  
set_annotated_delay(2)  
set_connection_class(2)  
set_disable_timing(2)  
set_dont_touch(2)  
set_dont_touch_network(2)  
set_drive(2)
```

characterize

set_driving_cell(2)
set_equal(2)
set_false_path(2)
set_fanout_load(2)
set_ideal_latency(2)
set_ideal_net(2)
set_ideal_network(2)
set_ideal_transition(2)
set_input_delay(2)
set_load(2)
set_logic_one(2)
set_logic_zero(2)
set_logic_dc(2)
set_max_area(2)
set_max_capacitance(2)
set_min_capacitance(2)
set_max_delay(2)
set_max_fanout(2)
set_max_transition(2)
set_cell_degradation(2)
set_fix_multiple_port_nets(2)
set_min_delay(2)
set_multicycle_path(2)
set_operating_conditions(2)
set_opposite(2)
set_output_delay(2)
set_resistance(2)
set_timing_ranges(2)
set_unconnected(2)
uniquify(2)

check_ccs_lib

Runs validation and correlation on the specified libraries.

SYNTAX

```
check_ccs_lib  
    -f config_file_name
```

Data Types

config_file_name string

ARGUMENTS

```
-f config_file_name  
    Specifies a configuration file name.
```

DESCRIPTION

This command runs validation and correlation on the specified libraries.

Usage: check_ccs_lib -f <config_file>
 -help Prints this help message.

config_file syntax for validation:

```
set cross_library      [ no | yes ]  
set precision          [ 5 | 6 | 7 | ... ]  
set generateCSV        [ none | failures | all ]  
set single_file_report [ no | yes ]  
set validation_type    [ ccs_timing | ccs_noise |  
                        { ccs_timing ccs_noise } ]  
set lib_list           [ test.lib | { test1.lib test2.lib ... } ]  
set delay_abs_tol      [ 15ps | 5ps | ... ]  
set delay_rel_tol      [ 0.04 | 0.03 | ... ]  
set slew_abs_tol       [ 30ps | 20ps | ... ]  
set slew_rel_tol       [ 0.10 | 0.05 | ... ]  
set [ NCVC-015 | NCVC-016 | ... ] ignore [ cell1 | { cell1 | cell2 | ... } ]  
set [ NCVC-015 | NCVC-016 | ... ] ignore  
set pt_shell_path      [ pt_shell_path ]  
set db_file            [ test.db | { test1.db test2.db ... } ]  
run validation
```

config_file syntax for correlation:

```
set library_file       [ lib_file_path ]  
set db_file            [ db_file_path ]  
set pt_shell_path      [ pt_shell_path ]  
set simulator          [ SPICE_simulator_path ]  
set spice_header_path  [ SPICE_common_header_path ]  
set netlist_path       [ SPICE_cell_netlist_directory ]  
set spice_include_file [ { 'file_name' 'file_name' ... } ]  
set spice_model_file   [ { 'file_name' option 'file_name' option ... } ]
```

check_ccs_lib


```

set tran_timestep      [ SPICE transient analysis interval ]
set correlation_type    [ ccs_timing | nldm_timing |
                        { ccs_timing nldm_timing } ]
set correlation_arcs    [ min | most | all ]
set waveform           [ predriver | pwl | real | user ]
set waveform_dir        [ waveform_directory ]
set waveforms_rise      [ { waveform_rise_list } ]
set waveforms_fall      [ { waveform_fall_list } ]
set sample_density      [ min | max | corners | all | NxN n ]
set sample_location     [ grid | interp ]
set driver_model_only   [ no | yes ]
set lsf_correlation     [ no | yes ]
set lsf_queue_name      [ lsf_queue_name ]
set lsf_maxruntime      [ lsf_run_time_limit ]
set lsf_maxrunjobs      [ lsf_running_job_limit ]
set vdd_name            [ VDD_name ]
set vss_name            [ VSS_name ]
set ad_cellname         [ active_driver_cellname input_pin output_pin ]
set ad_inslew           [ active_driver_input_ramp ]
set ad_inverting        [ no | yes ]
set spice_file_suffix   [ net | sp | typ | ... ]
set cells               [ { cell_list } ]
run correlation

```

EXAMPLES

The following example is used for library validation:

```

prompt> check_ccs_lib -f test.cfg
prompt> cat test.cfg
set lib_list test.lib
run validation

```

The following example is used for library correlation:

```

prompt> check_ccs_lib -f test.cfg
prompt> cat test.cfg
set library_file      test.lib
set db_file           test.db
set pt_shell_path     /usr/local/pt_shell
set simulator         /usr/local/hspice
set vdd_name          VDD
set vss_name          VSS
set correlation_mode   all
set netlist_path      /home/cae/netlists
set spice_header_path /home/cae/model.best
set spice_include_file "/home/cae/options.spi"
set correlation_type   { ccs_timing nldm_timing }
set waveform          predriver
set tran_timestep_in_ps 10
set sample_density     all
set sample_location    grid

```

```
set driver_model_only no
set debug_correlation yes
set lsf_correlation yes
run correlation
```

SEE ALSO

`check_library(2)`

check_clock_tree

Checks the clock trees of the current design for common problems that can adversely impact clock tree synthesis.

SYNTAX

```
status check_clock_tree  
[-clocks clock_list]
```

Data Types

clock_list list

ARGUMENTS

```
-clocks clock_list  
    Causes the command to only check those clock-trees whose names are in the  
    specified list. If the -clocks option is not provided, all clocks in the  
    current design are checked.
```

DESCRIPTION

The **check_clock_tree** command can be used at any time for a linked design to check for common problems that adversely impact clock-tree synthesis. The following classes of checks are performed:

CTS-810 exceptions defined on hierarchical pins CTS-811 clocks defined on hierarchical pins CTS-821 a master clock does not propagate to a generated clock CTS-822 improperly specified master clock CTS-823 a master clock terminates at a multi-clock pin CTS-831 a clock has no synchronous pins CTS-832 a clock loops to itself CTS-833 cascading master clocks CTS-834 multiple clocks per register CTS-835 exceptions defined on output pins CTS-836 unclocked sink pins CTS-840 multiple exceptions CTS-841 ignored exceptions

Some checks are specific to overlapping clock domains; these require that the shell variable **timing_enable_multiple_clocks_per_reg** be set true in order for the timing-engine to propagate multiple clocks per register. If the variable is set false when **check_clock_tree** is invoked, an extra check will be performed to detect and warn about overlapping clocks (CTS-834).

When **check_clock_tree** encounters a problem, a warning or informational message will be printed, sometimes followed by additional information to make solving the problem easier. There is a man page for each message type that explains the problem in detail and what a user can do to solve it.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

In the following example, the **check_clock_tree** command is used to check all of the clocks in the currently linked design:

```
prompt> check_clock_tree
```

In the following example, the **check_clock_tree** command is used to check a specific list of clocks in the currently linked design:

```
prompt> check_clock_tree -clocks [list clock1 clock2]
```

SEE ALSO

```
report_clock_tree(2)  
report_clock_timing(2)  
set_clock_tree_exceptions(2)  
compile_clock_tree(2)
```

check_database

Checks the integrity of the design database to ensure that design information has been correctly stored.

SYNTAX

```
status check_database
[library_name]
[-verbosity low | medium | high]
[-physical]
[-netlist]
```

Data Types

library_name string

ARGUMENTS

library_name
Specifies the name of the Milkyway design library to check. If it's not specified, check current opened cell in the library.

-verbosity low | medium | high
Specifies the verbosity level. Default verbosity is low. Higher verbosity will result in more displayed messages.

-physical
Specifies that physical information should be checked. Physical information includes, but is not limited to, floorplan, physical wiring, and UPF data. If neither "-physical" nor "-netlist" is specified, all information is checked.

-netlist
Specifies that logical netlist information should be checked. Logical netlist information includes, but is not limited to, netlist connectivity data. If neither "-physical" nor "-netlist" is specified, all information is checked.

DESCRIPTION

This command checks the database integrity and reports database problems. This command does not evaluate if the design information is correct, or that a legal, valid design can be constructed. It evaluates how the data is stored and whether this meets the database rules of construction.

EXAMPLES

```
prompt> check_database design
Warning: The hierarchical net 'VDD' is inconsistent with the flat net 'VDD'.
Warning: The via "vial2Fat_3_4" data is corrupted.
```

```
prompt> check_database design -verbosity high
Warning: The hierarchical net 'VDD' is inconsistent with the flat net 'VDD'.
The hierarchical net 'VDD' is power, while the flat net 'VDD' is signal.
The hierarchical net 'VDD' is connected to leaf pin 'U1/A', but the flat net 'VDD' is not.
The hierarchical net 'VDD' is connected to leaf pin 'U2/A', but the flat net 'VDD' is not.
Warning: The via "via12Fat_3_4" data is corrupted.
Via instance 0xF0 has a lower enclosure wire 0xC0 that does not match the via master enclosure size.
Via instance 0xF8 has a lower enclosure wire 0xC8 that does not match the via master enclosure size.
```

```
prompt> check_database design -verbosity high -physical
Warning: The via "via12Fat_3_4" data is corrupted.
Via instance 0xF0 has a lower enclosure wire 0xC0 that does not match the via master enclosure size.
Via instance 0xF8 has a lower enclosure wire 0xC8 that does not match the via master enclosure size.
```

```
prompt> check_database design -verbosity high -netlist
Warning: The hierarchical net 'VDD' is inconsistent with the flat net 'VDD'.
The hierarchical net 'VDD' is power, while the flat net 'VDD' is signal.
The hierarchical net 'VDD' is connected to leaf pin 'U1/A', but the flat net 'VDD' is not.
The hierarchical net 'VDD' is connected to leaf pin 'U2/A', but the flat net 'VDD' is not.
```

SEE ALSO

```
open_mw_cel(2)
check_design(2)
check_physical_design(2)
report_design(2)
```

check_design

Checks the current design for consistency.

SYNTAX

```
status check_design
[-summary]
[-no_warnings]
[-one_level]
[-multiple_designs]
[-no_connection_class]
[-post_layout | -only_post_layout]
```

ARGUMENTS

-summary

Displays a summary of the warning messages instead of one message per warning. This does not affect the way error messages are issued. The **-summary** option used along with the **-post_layout** or the **-only_post_layout** option only displays a summary of the missing back-annotation information.

-no_warnings

Suppresses warning messages, so only error messages are printed.

-one_level

Performs checks at only the current level of hierarchy. By default, **check_design** checks the current level and all designs below it.

-multiple_designs

Reports warning messages related to multiply-instantiated designs. With this option, the tool lists all multiply-instantiated designs along with instance names and associated attributes, such as `dont_touch`, `black_box`, and `ungroup`. By default, these messages are suppressed.

-no_connection_class

Suppresses connection class warning messages. This switch is useful while working on GTECH design/netlist, on which connection class violations are expected. By default, these messages are reported.

-post_layout

Checks the design for annotated information in a links-to-layout flow. The information is annotated on the design after the design has been placed and routed by the backend tool. This includes delay back-annotation, resistance back-annotation, capacitance back-annotation, and PDEF back-annotation; for example, clusters, cell locations, and so forth. The **-post_layout** option lists designs or instances that have any of these annotations missing. The **-post_layout** option must be used at least once for a design flow to validate the back-annotation part of the links-to-layout flow.

-only_post_layout

Checks only the annotated information in a links-to-layout flow. The information includes delay back-annotation, resistance back-annotation,

capacitance back-annotation, and PDEF back-annotation; for example, clusters, cell locations, and so forth. This option lists designs or instances that have any of these annotations missing. The **-only_post_layout** option can be used instead of the **-post_layout** option to validate the back-annotation part of the links-to-layout flow.

DESCRIPTION

The **check_design** command checks the internal representation of the current design for consistency, and issues error and warning messages as appropriate.

Error messages indicate design problems of such severity that **compile** does not accept the design; for example, recursive hierarchy in a design (when a design references itself) is an error. Warning messages are informational and do not necessarily indicate design problems. However, these messages should be investigated.

The **check_design** command flags multiple instances of a design within a system. If a design is instantiated in two different designs, a warning message is issued. For information on how to respond to error messages dealing with multiple instances, see the **uniquify** command man page.

The **check_design -summary** command automatically runs on every design that is compiled. However, you can use the **check_design** command explicitly to see warning messages.

Potential problems detected by this command include unloaded input ports or undriven output ports, nets without loads or drivers or with multiple drivers, cells or designs without inputs or outputs, mismatched pin counts between an instance and its reference, tristate buses with non-tristate drivers, and so forth.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command checks the current design for problems and issues error and warning messages:

```
prompt> check_design
```

The following command checks only the current level of the design:

```
prompt> check_design -one_level
```

SEE ALSO

[check_library\(2\)](#)
[check_timing\(2\)](#)


```
current_design(2)
set_dont_touch(2)
uniquify(2)
check_design_allow_non_tri_drivers_on_tri_bus(3)
```

check_error

Prints extended information on errors from the last command.

SYNTAX

```
status check_error  
[-verbose] [-reset]
```

ARGUMENTS

-verbose

Displays the list of errors found during previous commands.
By default, the command only returns the error status.

-reset

Resets the list of previously found errors. After resetting the error list, the **check_error -verbose** command returns an empty list.
By default, the command does not modify the error list.

DESCRIPTION

The **check_error** command is used to compare errors issued by previous commands to the list of error codes specified by the **check_error_list** variable.

If the **check_error** command finds errors that are specified in the **check_error_list** variable, the command returns 1. If the **check_error** command does not find any errors that are specified by the **check_error_list** variable, the command returns 0.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows how to use the **check_error** command to check if execution errors are specified by the **check_error_list** variable.

```
prompt> printvar check_error_list
```

```
"UID-3 EQN-18 EQN-19"
```

```
prompt> check_error
```

```
0
```

```
prompt> link
```

```
Warning: Can't read link_library file 'your_library.db'. (UID-3)
```

```
Linking design:
```

```
top
```

```
Using the following designs and libraries:
```

```
top, left, mux4_0, reg4_1, reg4_0, right, mux4_1, reg4_3, reg4_2, lsi_10k
```

```
(library)
```

```
Loading db file '/usr/libraries/syn/gtech.db'
```

```
check_error
```

```
208
```

```
Loading db file '/usr/libraries/syn/standard.sldb'
1
prompt> check_error
1
prompt> check_error -verbose
{"UID-3"}
1
```

The following command resets the error list:

```
prompt> check_error -reset -verbose
{"UID-3"}
1
prompt> check_error -verbose
{}
0
```

To use the **check_error** command in a script to see if specified errors occurred in the previous commands, use the following command:

```
prompt> if {[check_error]} {
    echo failure_message
    exit
} else {
    echo success_message
}
```

SEE ALSO

`check_error_list(3)`

check_fp_budget_result

Performs post-budgeting timing analysis.

SYNTAX

```
status check_fp_budget_result
[-blocks block]
[-pins pin]
-file_name output_design_report
```

Data Types

<i>block</i>	string
<i>pin</i>	string
<i>output_design_report</i>	string

ARGUMENTS

-blocks *block*
Specifies name of a plan group in the design. It reports budgeting analysis information related to that particular plan group in the design. You can specify the value of *block* by using a collection generated by the **get_cells** command. By default, the command generates the budgeting analysis report for all the pins on all the plan groups.

-pins *pin*
Specifies name of a pin on a plan group in the design. It reports budgeting analysis information related to that particular pin in the design. You can specify the value of *pin* by using a collection generated by the **get_pins** command. By default, the command generates the budgeting analysis report for all the pins on all the plan groups.

-file_name *output_design_report*
Specifies the name of the output design report file to which design information will be written.

DESCRIPTION

This command allows you to generate a report containing budgeted and actual delays through a hierarchical block.

A timing path starts at a start point and ends at an end point. For this report, hierarchical pins divide timing paths into timing path segments. This report prints the actual and budgeted delay for each of these path segments. This helps you to analyze the process of generating budgets for the design.

For each pin on the block, the report lists the worst-case timing paths per clock domain going through the pin. The report shows the budgeted and actual delays for each path segment on the timing path.

This command can be run only after calling the IC Compiler Timing Budgeter.

If you do not specify any of the optional options, the command writes out the information for the entire design. You can specify the **-blocks** or **-pins** option to restrict the information written.

In the report file, the information is divided by soft macro or plan group. All the timing path information related to one soft macro or plan group is printed together.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example creates a report file named *b1.rpt* containing information about a block named *B1* in the design.

```
prompt> check_fp_budget_result -blocks B1 -file_name b1.rpt
```

The following example creates a report file named *all.rpt* containing information about the entire design.

```
prompt> check_fp_budget_result -file_name all.rpt
```

SEE ALSO

```
allocate_fp_budgets(2)  
check_fp_timing_environment(2)  
get_cells(2)  
get_pins(2)
```

check_fp_pin_alignment

Checks the soft macro pin alignment QoR.

SYNTAX

```
status check_fp_pin_alignment  
[-detour]  
[-tolerance real]  
[-report_nets]  
[-nets collection]
```

ARGUMENTS

-detour

Generates a detour report for soft macro pins.

Note that unlike pin alignment checking, pin detour checking is performed on all two-pin and multi-pin nets connected to soft macros.

To determine the detour measure for a net, two bounding boxes are computed. The first bounding box minimally encompasses all the physical pins connected to the net, both inside and outside the soft macros. The second bounding box only minimally encompasses all the standard cell pins connected to the net (pins on soft macros are excluded). By definition, the first bounding box is at least as large as the second bounding box, and a detour exists if the first bounding box is larger than the second bounding box.

If you specify this option, the tool computes a count of the number of nets in which such a detour exists. When used in conjunction with the **-report_nets** option, the tool generates an actual list of nets with detours. To suppress reporting of nets that are slightly detoured, use the **-tolerance** option.

-tolerance *real*

Specifies the percentage tolerance for the detour report.

This option allows you to specify the percentage in which the perimeter of the first bounding box is allowed to be larger than the perimeter of the second bounding box, before the net is flagged as having a detour. It is useful for suppressing reporting of slightly detoured nets by supplying it with a small tolerance value, such as 0.1.

The valid range for tolerance is from 0 through 100 percent. A tolerance value greater than 100 is reset to 100. A tolerance value less than 0 is reset to 0. The default value for tolerance is 0.

-report_nets

Prints out the individual net names in the pin detour and pin alignment report.

By default, the tool outputs only the detoured net count.

-nets *collection*

Specifies the nets for which soft macro pin alignment and pin detour checks are performed.

DESCRIPTION

This command checks the soft macro pin alignment QoR.

If you do not specify the **-detour** option, the tool performs normal alignment checking. Alignment checking is limited to two-pin nets with at least one connection to a soft macro. Nets not connected to a soft macro or those with more than two connections of any kind are not checked. In particular, a net connected to a top-level pin (that is, a terminal) is not checked unless it is connected to a soft macro on the other end. A text report like the following is produced at the end of the alignment checking:

```
*****
Report: check_fp_pin_alignment pin alignment check
Design:  RISC_CORE
Version:  C-2009.06
Date:    Thu Apr 23 20:10:35 2009
*****
All numbers cited pertain only to two-connection nets with at least one
soft macro connection
186 total two-connection nets
176 alignable two-connection nets (176 non-blocked)
23 unaligned alignable two-connection nets (23 non-blocked)
66 aligned two-connection nets on different metal layers
0 aligned two-connection nets on different metal layers (soft macro ->
soft macro connections)
```

The definitions for terms used in this report are as follows:

alignable nets A two-pin net with only a single soft macro pin is considered alignable if the nonsoft macro pin (a terminal, a standard cell pin, or a hard macro pin) is located within the projection of the soft macro edge on which the soft macro pin is located. A two-pin net connected to soft macros at both ends is considered alignable if the projections of the soft macro edges on which the soft macro pins sit overlap.

non-blocked A two-pin net is considered "non-blocked" if the flyline connecting the two pins is not interrupted by any soft macro or hard macro not connected to the net.

If you specify the **-report_nets** option, the names of all nets that are alignable but are unaligned are printed out.

If you specify the **-detour** option, a detour report is generated. A detour is caused by an invalid soft macro pin placement, which causes routing from the source to targets to be longer than is necessary. The detour report consists of a simple count of the number of nets with a detour measure exceeding the tolerance specified by the **-tolerance** option. Detour measures of interconnected feedthrough nets created by IC Compiler during pin cutting or top-down pin placement are computed as a single net.

If you specify both the **-report_nets** option and the **-detour** option, the individual nets that fail the detour test are listed in the detour report.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example checks the soft macro pin alignment using a detour report.

```
prompt> check_fp_pin_alignment -detour
*****
Report: check_fp_pin_alignment detour check
Design: TOP
Version: C-2009.06
Date: Thu Apr 23 20:10:36 2009
*****
106 pin detours.

prompt> check_fp_pin_alignment -report_nets -detour
*****
Report: check_fp_pin_alignment detour check
Design: RISC_CORE
Version: C-2009.06
Date: Thu Apr 23 20:10:36 2009
*****
2 pin detours.
/***** Nets with pin detour, tolerance = 0.000000 *****/
OUT_VALID
Oprnd_A[11]
*****
```

SEE ALSO

`check_fp_pin_assignment(2)`

check_fp_pin_assignment

Performs pin placement checks.

SYNTAX

```
status check_fp_pin_assignment
[-pin_spacing]
[-pin_preroute_spacing]
[-shorts]
[-no_stacking]
[-layers]
[-wiretrack]
[-missing]
[-single_pin {connected | unconnected | all}]
[-off_edge]
[-outside_pin_guide]
[-block_level]
[-match_pin_shapes]
[-net_routing_constraints]
[-pin_type {signal_pins | pg_pins | all}]
[-macro_type {soft_macros | hard_macros | all}]
[-nets net_collection]
[objects]
```

Data Types

objects collection

ARGUMENTS

-pin_spacing

Determines if any two signal pins are closer than the number of wire tracks specified by the **-pin_spacing** option of the **set_fp_pin_constraints** command. The command does not perform this test by default.

-pin_preroute_spacing

Determines if any signal pin is closer to a preroute than the number of wire tracks specified by the **-pin_preroute_spacing** option of the **set_fp_pin_constraints** command. The command does not perform this test by default.

-shorts

Checks for shorts. A short occurs if two pins on the same layer touch or overlap each other. The command does not perform this test by default.

-no_stacking

Determines if pins are placed on legal layers. If the **-no_stacking** option is used in an earlier call to the **set_fp_pin_constraints** command to avoid pin overlaps on different layers, the **-no_stacking** option determines if the rule specified by **-no_stacking** in **set_fp_pin_constraints** is honored. The available **no_stack_type** rules in

set_fp_pin_constraints are **all**, **pg_pins_only**, and **signal_pins_only**.

It is important to note that the types of checks performed on individual macros depends on the specific stacking setting associated with each macro. For example, if a design has macros A, B, C with the following stacking rules:

A -Pin stacking is allowed for both PG and signal pins.

B -Pin stacking is not allowed on either PG or signal pins.

C -Pin stacking is allowed only for signal pins.

Then, if you use the **-no_stacking** option, the actual tests performed on these macros are as follows:

A -No check for pin stacking is performed, since pin stacking of both kinds is allowed.

B -Pin stacking checks are performed on both PG and signal pins.

C -Pin stacking checks are performed only on PG pins.

If the **-block_level** option is used, terminals are checked based on the stacking setting used in an earlier call to **set_fp_pin_constraints -block_level**.

If the stacking setting was not explicitly set through **set_fp_pin_constraints**, no check is performed, since pin stacking of both kinds is allowed.

Finally, using the **-no_stacking** option implicitly turns on the **-shorts** option. For more information, see the **set_fp_pin_constraints** man page. The command does not perform this test by default.

-layers

Determines if pins are placed on legal metal layers as specified by the **set_fp_pin_constraints** command used with the **allow_layer** option. For the default minimum and maximum metal layers for pin placement, see the **set_fp_pin_constraints** man page. The command does not perform this test by default.

-wiretrack

Determines if pins are centered on wire tracks. The command does not perform this test by default.

-missing

Determines if any pins are missing. The command does not perform this test by default.

-single_pin {connected | unconnected | all}

Checks for unroutable pins on abutted edges. If given the value "connected" or "all", a macro pin is flagged as a single (unroutable) pin if any one of the following scenarios occurs:

1. The macro pin is placed on a part of a macro edge that abuts another macro edge and the macro pin does not abut the other pin on the same metal layer to which it is connected.

2. The macro pin is placed on a part of a macro edge that coincides with the top-level cell boundary and the macro pin is connected to anything other than a terminal (top-level pin), unless the pin is connected solely to other pins on the same macro and those pins are stacked on top of the pin.

3. The macro pin is placed on a part of a macro edge that coincides with the top-level cell boundary and the macro pin is connected solely to a terminal but the terminal is not stacked on top of the macro pin on the same metal layer.

If given the value "unconnected" or "all", a macro pin is flagged as a single (unroutable) pin if either one of the following scenarios occurs:

1. The macro pin is logically unconnected to any pin or terminal and is placed on a part of a macro edge that abuts another macro edge.
2. The macro pin is logically unconnected to any pin or terminal and is placed on a part of a macro edge that coincides with the top-level cell boundary. This option also performs checks on terminals if given the value "unconnected" or "all". A terminal is flagged if it is logically unconnected and sits on the part of the top cell boundary that abuts a macro edge. The command does not perform this test by default. This option is ignored if the `-block_level` option is used.

`-off_edge`

Checks if part of a pin outline coincides with an edge.
A soft macro pin will be flagged if none of its edges coincide with an edge of the soft macro to which it belongs. If the **`-block_level`** option is used, a terminal will be flagged if none of its edges coincide with an edge of the top-level cell.
The command does not perform this test by default.

`-outside_pin_guide`

Checks if the center of a pin is not inside the associated pin guide. The command does not perform this test by default.

`-block_level`

Checks block-level pin placement. The tests you specify are performed on terminals (top-level pins). By default, the tests are performed on soft macros pins.

`-pin_type {signal_pins | pg_pins | all}`

Specifies the types of pins to be checked. This option takes as an argument one of *signal_pins*, *pg_pins*, or *all*. Signal pins are checked if "signal_pins" is selected. Power and ground pins are checked if "pg_pins" is selected. Both signal and PG pins are checked if "all" is selected. However, PG pin checks are limited to the **`,Äismatch pin shapes`**, **`,Äisingle pin`**, **`,Äishorts`**, **`-layer`**, **`-missing`**, and **`-off_edge`** options. All other checks specified in conjunction with the **`-pin_type pg_pins`** option are silently ignored. Only signal pins are checked if this option is not used.

`-macro_type {soft_macros | hard_macros | all}`

Specifies the types of macros whose pins are to be checked. This option takes as an argument one of *soft_macros*, *hard_macros*, or *all*. Soft macro pins are checked if "soft_macros" is selected. Hard macro pins are checked if "hard_macros" is selected. Both soft macro pins and hard macro pins are checked if "all" is selected. However, hard macro pin checks are limited to the **`,Äismatch pin shapes`**, **`,Äisingle pin`**, **`,Äishorts`**, **`-missing`**, and **`-off_edge`** options. All other checks specified in conjunction with the **`-macro_type hard_macros`** option are silently ignored. Except for the **`-single_pin`** option, only soft macro pins are checked if this option is not used. In the case of the **`-single_pin hard macros`** option, signal pins, PG pins or both, depending on whether or how the **`-pin_type`** option is used in conjunction, are always checked.

`-nets net_collection`

Specifies a collection of nets for which to check pin placement. If you specify this option, the command checks pin placement only on soft macro pins and relevant hard macro pins connected to the specified nets. All relevant

pins on specified soft and hard macros are checked if this option is not used. If this option is not used and no macros are specified, checks are performed on all relevant soft macro pins and hard macro pins.

objects

Specifies the collection of macros for which to check pin placement. The command checks pin placement only on the macros you specify. By default, if no macros are specified, the checks are applied on pins on all soft and relevant hard macros.

DESCRIPTION

This command performs pin placement checks as specified. It reads in relevant constraint values that were set earlier by using the **set_fp_pin_constraints** command to perform the pin placement checks. Checks will be performed against default pin constraints if no pin constraints have been set. You can review pin constraints with **report_fp_pin_constraints**.

Checks are performed on terminals (top-level pins) if the **-block_level** option is used. Routability checks are also performed on hard macro pins if the **-single_pin** option is used. No other checks are performed on hard macro pins unless the **-macro_type hard_macro** is specified. See explanations for the **-macro_type** and **-single_pin** options for details.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example checks a soft macro named A for shorted and missing pins. It also determines if any two signal pins are closer than the number of wire tracks previously specified by using the **-pin_spacing** option of the **set_fp_pin_constraints** command.

```
prompt> check_fp_pin_assignment -short -missing \  
-pin_spacing [get_cells A]
```

SEE ALSO

```
check_fp_pin_alignment(2)  
report_fp_pin_constraints(2)  
set_fp_pin_constraints(2)  
set_net_routing_rule(2)  
set_net_routing_layer_constraints(2)
```

check_fp_rail

Checks the integrity of the power network created by power network synthesis, early in the design planning stage.

SYNTAX

```
status check_fp_rail  
-nets nets  
[-ring]  
[-floating_segment]  
[-power_switch_connection]
```

Data Types

nets collection or list

ARGUMENTS

-nets *nets*
 Specifies the net name(or names) of the power network for integrity checking. If no power straps exist with the specified net name, the command fails and issues an error message. You can specify a single net name or multiple net names separated by a comma or a space.

-ring
 Checks only the core rings of the power network. If a core ring segment is broken, a warning message is issued. By default this option is off.

-floating_segment
 Checks only the floating segments in the power network. If a floating segment is detected, a warning message is issued. By default, this option is off.

-power_switch_connection
 Checks only the connection between power switch cell input pin and the permanent power/ground straps. If there are unconnected power switch input pins, a warning message is issued. Use this option in a multivoltage design with power-down voltage areas. By default, this option is off.

DESCRIPTION

This command allows you to check the integrity of the power network created by the **synthesize_fp_rail** and the **commit_fp_rail** commands early in the design stage. If there are broken core rings, floating segments, or unconnected power switch input pins in the power mesh, a warning message is issued.

EXAMPLES

The following example shows how to check the VDD ring of a power plan.

```
prompt> check_fp_rail -net {VDD} -ring
```

WARNING: Core ring is broken at (123.400 227.900)

SEE ALSO

`synthesize_fp_rail(2)`
`commit_fp_rail(2)`

check_fp_timing_environment

Performs timing environment analysis.

SYNTAX

```
integer check_fp_timing_environment
[-block_pin_stats]
[-unbudgetable_pins]
[-unconstrained_pins]
[-exception_pins]
[-static_logic_pins]
[-delay_violators percent]
[-num_pin_connections connections]
[-block_name string]
[-pin_name string]
[-zero_wire_delay slack_percent]
[-bottleneck slack_limit]
[-bottleneck_max_cell num_cells]
[-bottleneck_no_vipo]
[-vipo_timing slack_percent]
[-format_report]
```

Data Types

<i>percent</i>	float
<i>connections</i>	integer
<i>string</i>	string
<i>slack_percent</i>	float
<i>slack_limit</i>	float
<i>num_cells</i>	integer

ARGUMENTS

-block_pin_stats

Specifies whether to print block pin statistics. It prints the number of budgetable pins, the number of unbudgetable pins, and the total number of pins on the block.

-unbudgetable_pins

Specifies whether to print block pins that cannot be budgeted. It prints why each pin cannot be budgeted:

- * Pin is not connected to net
- * Pin is not connected to a top-level net, but is connected to an internal net
- * Pin is not connected to an internal net, but is connected to a top-level net
- * Only fixed delay exists on one side of the pin
- * Timing paths through the pins are not constrained
- * Pin is an input or output clock port
- * Pin is connected to power or ground net, or has set_case_analysis on it

`-unconstrained_pins`
 Specifies whether to print unconstrained block pins. A pin is unconstrained if it has no timing paths through it.

`-exception_pins`
 Specifies whether to print block pins touched by an exception. The exception can be `set_false_path`, `set_multicycle_path`, `set_min_delay`, or `set_max_delay`.

`-static_logic_pins`
 Specifies whether to print block pins that are set to static logic state. It prints whether a block pin is tied to high or low by a power or ground net in the input netlist. It also reports if a block pin is set to specific state by `set_case_analysis`, `set_logic_one`, or `set_logic_zero`.

`-delay_violators percent`
 Where *percent* ≥ 0 . Specifies whether to print cells with delays greater than the specified percentage of the capture clock period. The *percent* argument must be greater than or equal to zero. It traverses all the critical paths per clock domain through a block pin.

`-num_pin_connections connections`
 Where *connections* > 0 . Specifies whether to print number of net connections from this block to other blocks, top-level standard cells, pad cells, or to top-level ports. It also specifies the number of connections to be displayed in the GUI for each type of net connection.

`-block_name string`
 Specifies whether to print a budgeting report for a particular block only. The name of block could be a plan group, instance name, or master name.

`-pin_name string`
 Specifies whether to print a budgeting report for a particular pin only. The name of pin must exactly match a pin on the plan group.

`-zero_wire_delay slack_percent`
 Where *slack_percent* < 0 . Specifies whether to perform timing analysis with zero wire delays and print the paths that have slack less than the negative percentage of the capture clock period. Both the path and the number of logic levels on the path are printed.
 The *slack_percent* is the percentage of the clock period that is used as the threshold for printing negative slack paths. It must be greater than or equal to zero.

`-bottleneck slack_limit`
 Specifies whether to report bottleneck cells. It prints the leaf cells that contribute to multiple timing violations. It lists the leaf cell, its reference, and the number of paths through the leaf cell that have timing violations. Based on this report, you can check the fanin and fanout logic to determine the possible cause of the timing bottleneck.
 It also specifies the slack limit for reporting bottleneck cells. Only timing paths having slack less than *slack_limit* are explored for finding bottleneck cells.

`-bottleneck_max_cell num_cells`
 Where *num_cells* > 0 . Specifies the number of bottleneck cells to report. You

must specify a positive integer. If this option is not specified, a maximum of 20 bottleneck cells are reported.

-bottleneck_no_vipo

Specifies whether to perform virtual in-place optimization (vipo) before reporting bottleneck cells. The default is to run virtual in-place optimization.

-vipo_timing *slack_percent*

Specifies whether to report timing analysis results based on virtual in-place optimization (vipo). This report can help visualize the timing analysis report after the actual in-place optimization has been run.

It also specifies the slack limit for timing paths in the virtual in-place optimization timing report. Only the paths having slack less than *slack_percent* of the capture clock period are reported.

-format_report

Specifies whether to format the report in table form. If this option is not specified, the report is printed in a single line format. The report in table form is easier to look at, but may be difficult to parse using automated tools.

DESCRIPTION

This command allows you to generate a report that contains additional design information that helps you analyze the budgets assigned to each block. The block can be a plan group or a soft macro. The report can be used to debug the budgets generated by the IC Compiler design planning timing budgeter.

The additional design information includes, but is not limited to, timing constraints in the design, timing exceptions in the design, delay distribution on the worst timing path through a block port, pins tied to high or low, high negative slack on a timing path, and so on.

This command can be run before or after calling the IC Compiler design planning timing budgeter.

All of the command options are optional. The options determine the type of information put in the output report.

If no options are specified, no information is written out. You must provide at least one option to have something in the output report.

In the output, the following information is printed for the whole design:

- * zero wire delay based negative slack paths
- * bottleneck cells
- * virtual in-place optimization timing report

The rest of the report is divided by soft macro or plan group. All the timing environment information related to one soft macro or plan group is printed together.

The **-bottleneck_max_cell** and **-bottleneck_no_vipo** options are ignored if the **-bottleneck** option is not specified.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following command creates a report file, stat.rpt, that contains summary information about budgetable, unbudgetable and total block pins in the design.

```
prompt> check_fp_timing_environment -block_pin_stats > stat.rpt
```

The following command creates a report file, violators.rpt, that contains cell delay violators. The cells are printed if they have a delay of more than 50% of the related clock period.

```
prompt> check_fp_timing_environment -delay_violators 50 > violators.rpt
```

The following example reports block pins that are unconstrained or have timing exceptions on them.

```
prompt> check_fp_timing_environment -unconstrained_pins \  
-exception_pins
```

The output has the following format:

```
=====
Plan Group Name: <PG Name>, Master Name: <Reference Name>
=====
```

```
##### Information regarding block pins without constraint or delay informa  
tion
```

```
Plan Group Port = <port name1>: is not constrained.
```

```
Plan Group Port = <port name2>: is not constrained.
```

```
##### Information regarding blocks pins that are touched by an exception
```

```
Plan Group Port = <port name3>: Port touched by set_false_path exception.
```

```
Plan Group Port = <port name4>: Port touched by set_multicycle_path except  
ion.
```

SEE ALSO

```
allocate_fp_budgets(2)  
check_fp_budget_result(2)
```

check_isolation_cellsfp

Reports the existing isolation cells in the current design. It also reports if any isolation cell is redundant or might be required.

SYNTAX

```
status check_isolation_cells
[-input]
[-output]
[-inside]
[-outside]
[objects]
```

Data Types

objects list

ARGUMENTS

-input
Checks the isolation cells only on the input nets of the top-level logical hierarchies or voltage areas.

-output
Checks the isolation cells only on the output nets of the top-level logical hierarchies or voltage areas.

-inside
Checks the isolation cells that are inside the top-level logical hierarchies or voltage areas.

-outside
Checks the isolation cells that are outside the top-level logical hierarchies or voltage areas.

objects
Specifies a list of voltage areas (in physical context) or top-level logical hierarchies (in logical context) that are for which the query is run.

DESCRIPTION

The **check_isolation_cells** command reports the isolation cells that are associated with the top-level logical hierarchies or voltage areas in the design. The top-level logical hierarchies can be given as inputs only from `dc_shell` and voltage areas can be given as inputs only from `psyn_shell`.

This command also reports if there are possible redundant isolation cells in the design. When in `dc_shell`, this command reports about the possible requirement of an isolation cell on a net if that nets crosses two logical hierarchies without any isolation cell. In `psyn_shell`, if a net is sourced in a voltage area and sinked into another, the command reports that there might be a requirement of an isolation cell

on that net.

If no logical hierarchies are given in this command in `dc_shell`, the command looks all of the top-level logical hierarchies and reports on the isolation cells based upon the given switches.

In `psyn_shell`, if no voltage areas are specified, the command looks at all existing voltage areas, except the default voltage area, and reports about the isolation cells according to the given switches. The command does not accept default voltage area as an input from `psyn_shell`.

This command works both for multisupply and multivoltage designs. In the case of a multivoltage design, this command treats enabled level shifter cells also as isolation cells.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following is an example report from the **`check_isolation_cells`** command:

```
prompt> check_isolation_cells

*****
Information : Output vdd_region_2/state_out[31] has an isolation cell
state_out_iso31 outside vdd_region_2
Information : Output vdd_region_2/state_out[30] has an isolation cell
state_out_iso30 outside vdd_region_2
Information : Output vdd_region_2/state_out[29] has an isolation cell
state_out_iso29 outside vdd_region_2
Information : Output vdd_region_2/state_out[28] has an isolation cell
state_out_iso28 outside vdd_region_2
Information : Output vdd_region_2/state_out[27] has an isolation cell
state_out_iso27 outside vdd_region_2
Information : Output vdd_region_2/state_out[26] has an isolation cell
state_out_iso26 outside vdd_region_2
Information : Output vdd_region_2/state_out[25] has an isolation cell
state_out_iso25 outside vdd_region_2
Information : Output vdd_region_2/state_out[24] has an isolation cell
state_out_iso24 outside vdd_region_2
Information : Output vdd_region_2/state_out[23] has an isolation cell
state_out_iso23 outside vdd_region_2
Total number of isolation cell(s) : 9
*****
1
```

SEE ALSO

`check_level_shifters(2)`
`check_library(2)`

`check_isolation_cellsfp`
226

check_legality

Checks the legality of the current placement.

SYNTAX

```
int check_legality  
[-verbose]
```

ARGUMENTS

-verbose

Prints a detailed report of all possible violations. Without this option, only a summary of all violations is printed.

DESCRIPTION

The **check_legality** command checks the legality of current placement, and prints out a report of statistics about violations. The violations can be classified as:

1. Cells that are not on rows.
2. Cells overlapping each other.
3. Cells overlapping with blockages in the design.
4. Cells with orientation that is not allowed for the row on which a cell is placed.
5. Cells with a core site type not equal to that of the row on which cells are placed.
6. Cells overlapping with power straps in the design.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows the command for printing out a summary of all violations.

```
prompt> check_legality
```

SEE ALSO

```
create_placement(2)  
legalize_placement(2)
```

check_level_shifters

Checks the design for all existing level shifters and nets against the specified level shifter strategy and threshold.

SYNTAX

```
status check_level_shifters  
[-verbose]
```

ARGUMENTS

-verbose
Prints a detailed report of all possible violations. By default, only a summary of all violations is printed.

DESCRIPTION

This command checks the design for all existing level shifters and nets against the specified level shifter strategy and threshold. It prints out complete statistics of the design regarding existing and required voltage adjustments containing the following information:

- Level shifter strategy and threshold
- Nets that need level shifters
- Nets that need level shifters but are dont_touched
- Nets that have multiple driver pins at different operating voltages
- Total number of level shifters in the design
- Total number of violating level shifters in the design
- Total number of violating nets in the design.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example prints out a summary of all violations:

```
prompt> check_level_shifters
```

```
*****
Check Level Shifters and Nets Summary
*****
Level shifter strategy           :low_to_high
Level shifter threshold voltage  :0
Level shifter threshold percent  :0
Number of violating nets (dont_touch) :2
Number of violating nets (multiple driver) :0
Number of violating nets (total) :2
Number of violating level shifters :32
*****
```

The following example shows the command with verbose option:

```
prompt> check_level_shifters -v
```

```
-----
Net           Driver Pin   Input Volt   Load Pin   Output Volt
-----
n1            U1/out1      0.9          LS1/A        1.08

=====
Level Shifter  Reference   Input  Output Violation Reason   Opcond
                Volt    Volt
=====
LS1            LVL2       0.9    0.7    TRUE      Lib Cell  max_v9
=====
```

SEE ALSO

```
check_library(2)
insert_level_shifters(2)
remove_level_shifters(2)
set_level_shifter_strategy(2)
set_level_shifter_threshold(2)
```

check_library

Performs consistency checks between logic and physical libraries, cross logic libraries, and intra-physical libraries.

SYNTAX

```
status check_library  
[-mw_library_name phys_library_name_list]  
[-logic_library_name logic_library_name_list]  
[-cells cell_list]
```

Data Types

<i>phys_library_name_list</i>	list
<i>logic_library_name_list</i>	list
<i>cell_list</i>	list

ARGUMENTS

-mw_library_name *phys_library_name_list*
Specifies Milkyway reference library names to be checked. By default, the reference libraries set by the **mw_reference_library** variable are checked. Use the main or design library to check for technical consistency and cells of the same name in physical libraries.

-logic_library_name *logic_library_name_list*
Specifies logic library file names to be checked. By default, the link libraries set by the **link_library** variable are checked. For logic versus logic library checking, this option specifies two or more libraries; for example, minimum and maximum libraries. If the **-logic_library_name** option is not explicitly specified but **set_min_library max_library** is specified, the command checks the minimum and maximum libraries as a pair. If neither **-logic_library_name** nor maximum and minimum libraries are specified, but **link_library** and **search_path** are set, the command first groups the libraries in the list by cell set, and within each group or family the command checks consistencies across all libraries in the same group. Libraries without grouping, that is, libraries that cannot be paired, are not checked. For **-scaling**, specify the scaling libraries by **-logic_library_name** option that are defined in **define_scaling_lib_group**, or specify them by **define_scaling_lib_group** command. If **-logic_library_name**, **set_min_library**, **define_scaling_lib_group** and **\$link_library** are all specified, the priority order is also this sequence with **-logic_library_name** the highest. For **-compare** and **-validate**, specify only two libraries. This option is ignored for physical library checking.

-cells *cell_list*
Specifies a list of cell names to be checked. If not specified, all cells in the libraries are checked.

DESCRIPTION

The **check_library** command checks and reports the items described below, based on the settings of the **set_check_library_options** command.

This command checks library qualities in three main areas:

- Physical library quality
- Logic versus physical library consistency
- Logic versus logic consistency.

The command performs the following consistency checks between logic and physical libraries:

- Missing cells in the logic or physical library
- Missing or mismatched pins (pin names, directions, and types) in the logic and physical library. Note that the pin direction keyword *input* in a logic cell versus *Input* in a physical cell is not a mismatch.
- Area values of each standard cell between .lib/.db model and FRAM view (PRBoundary and CellBoundary)
- Cell footprint (**cell_footprint** attribute)
- Bus character naming style in logic and physical libraries.

Physical library qualities are checked as follows:

- Technology data consistency between the specified main library and each linked reference library
- Cell view versus FRAM view in the library for missing views and mismatched views
- Cells with identical names in different reference libraries and the specified main library
- Signal EM rule
- Antenna rules and missing antenna properties
- Rectilinear cells
- Physical-only cells
- Physical properties for place and route including unit tiles for the libraries
- Pin routability

- Technology data quality
- DRC for each cell in the library.

By default, missing cells and pins, and mismatched pins are checked.

Logic versus logic library consistency checks include:

- General checking at library, cell, pin and timing group levels: Missing cells, missing and mismatched pins or `pg_pins` and timing arcs.
- Special checking for timing, noise and power scaling
- Special checking for UPF/MV flow (`pg_pin`, power management cells and power data etc.)
- Special checking for MCMM flow (operating conditions and `power_down_function`, etc.)
- Library characterization and validation.

All of the checking functions are available in both IC Compiler and Design Compiler, and the functions of logic versus logic library checking are also available in Library Compiler. If multiple logic libraries are specified for default checking, and if no cells are found in any of the libraries that match the physical library, the cells are counted as missing in the logic library unless these are physical-only cells. Otherwise, if the cells are found in any of the logic libraries, such as the first library, they are counted as consistent in cell names.

If no libraries are specified by the `-mw_library_name` and `-logic_library_name` options, the command checks logic versus physical libraries that are already set up for the design by the `mw_reference_library` and `link_library` variables. If there is no design set up, the command does not perform any checks.

Explicitly specified libraries will override the defaults. If both physical and logic libraries are specified, no design library is required. However, if a design library is open, the explicitly specified libraries in `check_library` are checked and the libraries in the design are ignored.

Specify a design library when checking cells of the same name and technical consistency in Milkyway libraries. To check logic versus physical libraries for the current design, use the `set search_path`, `set link_library`, `set mw_reference_library`, or `open_mw_lib` command to set the libraries to be checked before issuing `check_library`. For logic versus physical library checking, at the end of checking, the command reports a cross-check summary that includes the following:

- Number of cells missing in the logic library (excluding physical-only cells)
- Number of cells missing in the physical library
- Number of cells with missing or mismatched pins in libraries, if any
- The logic versus physical library quality checking PASSED, or Logic library is INCONSISTENT with the physical library.

For logic versus logic library checking, at the end of checking, the command reports

a summary for each specified option. For example, for -mcmm, it reports:
Logic library consistency check FAILED for MCMM.

Use **check_library** before reading in a design.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

In the following example, the lib1.db and lib2.db logic libraries are checked against the phys_lib physical libraries for missing cells and pins, mismatched pins, the cell footprint and bus delimiter.

```
prompt> set_check_library_options -cell_footprint -bus_delimiter
1
```

```
prompt> check_library -mw_library_name {phys_lib} \
               -logic_library_name {lib1.db lib2.db}
1
```

```
#BEGIN_XCHECK_LIBRARY
```

```
Logic Library:      lib1.db
                   lib2.db
Physical Library:   phys_lib
check_library options: -cell_footprint -bus_delimiter
Version:           A-2007.12
Check date and time: Thu Jul 26 16:59:54 2007
```

```
#BEGIN_XCHECK_LOGICCELLS
```

```
Number of cells missing in logic library:      3 (out of 3348)
```

List of cells missing in logic library

Cell name	Cell type	Physical library
AND2	Core	phys_lib
NOR1	Core	phys_lib
XOR3	Core	phys_lib

List of physical only cells

Cell name	Cell type	Physical library
GFILL	Filler	phys_lib
GFILL10	Filler	phys_lib
FILL8	Filler	phys_lib

#END_XCHECK_LOGICCELLS

#BEGIN_XCHECK_PHYSICALCELLS

Number of cells missing in physical library: 0 (out of 846)

#END_XCHECK_PHYSICALCELLS

#BEGIN_XCHECK_PINS

Number of cells with missing or mismatched pins in libraries: 1

List of pins mismatched in logic and physical libraries

Logic library: lib1.db

Physical library: phys_lib

Cell name	Pin name	Pin direction		Pin type	
		Logic	Physical	Logic	Physical
pn1123	VSSO	output	Output	signal	ground
	SGND	input	Input	signal	ground

#END_XCHECK_PINS

#BEGIN_XCHECK_BUS

List of bus naming styles

Library name	Library type	Bus naming style
phys_lib	Physical library	__<%d>
lib1.db	Logic library	

#END_XCHECK_BUS

#BEGIN_XCHECK_FOOTPRINT

Number of footprints: 1

List of cells with cell_footprint attribute

Footprint	Logic library name	Cell name	PR boundary
TIEH	lib1.db	GTIEH	(0,0) (0.8,1.8)
	lib1.db	TIEH	(0,0) (0.6,1.8)

#END_XCHECK_FOOTPRINT

Cross check summary:

Number of cells missing in logic library: 3

```
Number of cells with missing or mismatched pins in libraries:    1
Logic library is INCONSISTENT with physical library.
```

```
#END_XCHECK_LIBRARY
```

```
1
```

SEE ALSO

```
report_check_library_options(2)
set_check_library_options(2)
link_library(3)
mw_reference_library(3)
```

check_license

Checks the availability of a license for a feature.

SYNTAX

```
status check_license  
feature_list
```

Data Types

```
feature_list      list
```

ARGUMENTS

```
feature_list  
    Specifies the list of features to be checked. If more than one feature is  
    specified, they must be enclosed in braces ({}). By looking at your key file,  
    you can determine all of the features licensed at your site.
```

DESCRIPTION

This command checks on a license for the named features. It does not check out the license.

The **list_licenses** command provides a list of the features that you are currently using.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

This example checks on the license for the multivoltage feature:

```
prompt> check_license {Galaxy-MV}
```

SEE ALSO

```
get_license(2)  
license_users(2)  
list_licenses(2)  
remove_license(2)
```

check_mpc

Checks the result of the minimum physical constraints options on the design after running the design through the minimum physical constraints flow.

SYNTAX

```
int check_mpc  
[-macros]  
[-ports]  
[-conflicting]  
[-verbose]  
[object_list]
```

Data Types

object_list collection or list

ARGUMENTS

-macros

Checks the constraints for macro objects only. Set the constraints with the **set_mpc_macro_options** command and the **set_mpc_macro_array** command.

-ports

Checks the constraints for port objects only. Set the constraints with the **set_mpc_port_options** command.

-conflicting

Reports only the conflicting constraints, such as constraints that are failing.

-verbose

Includes the values used to check the minimum physical constraints against the final placement in the report. This option produces a more detailed than without the switch.

object_list

Specifies a list of designs, macros, ports, or group_names that are specified in the design.

DESCRIPTION

The **check_mpc** command verifies the design level physical constraints against the actual floorplan created after the minimum physical constraints flow, which consists of the **create_placement -mpc**, **compile_physical -mpc**, and the **physopt -mpc** flows. Use **check_mpc** to verify and debug the constraints, and to determine if any constraints were or were not met and the reason why.

The constraints and the actual floorplan are read in from the .db file.

Multicorner-Multimode Support

This command is not supported in a multi-scenario design flow.

EXAMPLES

After reading in a .db file that contains the constraints and placement information, you can check the constraints on the *current_design*, as shown in the following example:

```
prompt> check_mpc [current_design]
Current design is 'MRISC'.
set_mpc_options
  -aspect_ratio      1.500      1.510      Not Met
  -left_port_limit   10         10         Met
  -right_port_limit  10         10         Met
  -top_port_limit    10         10         Met
  -bottom_port_limit 100        67         Met
  -min_port_pitch    1          560        Met
1
```

The following example checks the constraints on port objects only:

```
prompt> check_mpc -port
set_mpc_port_options Instrn[8]
  -side      top      top      Met
set_mpc_port_options Instrn[7]
  -side      top      top      Met
set_mpc_port_options Instrn[6]
  -side      top      top      Met
1
```

The following example checks constraints on macro objects only:

```
prompt> check_mpc -macro
set_mpc_macro_options I_MBOX_QTM_core1
  -legal_orientations N S FN FS      N      Met
  -anchor_bound       r      (215.520 176.360 315.520 276.360) Met
  -x_offset           2.000      2.000      Met
  -align_pins         {PSW[10] Y[0]}      Met
set_mpc_macro_options I_MBOX_QTM_core2
  -legal_orientations N S FN FS      N      Met
  -anchor_bound       t      (3.170 376.360 103.170 476.360) Met
  -y_offset           3.000      3.000      Met
1
```

The following example includes the values used to check the minimum physical constraints of the I_MBOX_QRT_core1 object against the final placement in the report:


```

prompt> check_mpc -verbose I_MBOX_QTM_core1
set_mpc_macro_options      I_MBOX_QTM_core1
-legal_orientations  N S FN FS          N          Met
-anchor_bound        r                  (215.520 176.360 315.520 276.360) Met
The Bound for this region 'r' is [215.520 0.000 315.520 479.360]
-x_offset            2.000              2.000          Met
-align_pins          {PSW[10] Y[0]}      Met
The location of the pin Y[0] on macro I_MBOX_QTM_core1 is {315.020 196.860}
The location of the port PSW[10] is {317.380 196.840}
1

```

The following examples show how run **check_mpc** on specific object lists:

```

prompt> check_mpc [all_macro_cells]

prompt> check_mpc [get_ports ALL_OUTS*]

prompt> check_mpc -verbose

prompt> check_mpc [list [current_design] [get_ports In*]] \
[all_macro_cells]

```

SEE ALSO

```

create_placement(2)
report_mpc_macro_options(2)
report_mpc_options(2)
report_mpc_port_options(2)
set_mpc_macro_array(2)
set_mpc_macro_options(2)
set_mpc_port_options(2)

```

check_mv_design

Checks for violations in a multivoltage design.

SYNTAX

```
status check_mv_design
[-verbose]
[-isolation]
[-target_library_subset]
[-opcond_mismatches]
[-connection_rules]
[-level_shifters]
[-power_nets]
[-max_messages message_count]
[-clock_gating_style]
```

Data Types

message_count unsigned integer

ARGUMENTS

-verbose

Reports violations in the design in detail. If this option is not specified, the command provides only a summary of the violations.

-isolation

Reports electrical isolation requirements on nets connecting power domains. This option enforces the following checks:

- Nets that require isolation but on which the isolation cell is missing
- Nets that do not require isolation but for which an isolation cell is added
- Isolation cells that do not cross power domain boundaries.

-target_library_subset

Checks inconsistent settings between the target library, target library subset, and operating conditions. This option enforces the following checks:

- Conflicts between the target library subset and the global **target_library** variable. The target library subset should be a subset of the library set specified by the **target_library** variable.
- Conflicts between the operating condition and target library subset. There should be at least one library from the target library subset that has the same process, voltage, and temperature as the operating condition being used on a block.
- Conflicts between a cell and target library subset specification on the parent design of the cell. This check ensures that cells from the specified target library subset are used.

`-opcond_mismatches`

Reports on technology cells instantiated in the design with incompatible operating conditions. The option checks for conflicts between a cell and the operating condition specified on the parent design of the cell. This check ensures that the operating condition at which the cell is characterized matches the operating condition specified on the design.

`-connection_rules`

Reports violations in always-on synthesis and pass gate connections. The option enforces the following checks:

- Always-on net driven by a normal cell
- Always-on net or a net from an always-on domain driving the pass gate
- Always-on cell driving normal net
- Two pass gates connected to each other.

`-level_shifters`

Reports level shifter related violations. The following types of violations are reported:

- Nets with missing level shifters
- Nets on which the level shifter cannot be added because either the nets are marked dont touch or the nets are driven by pins operating at different voltages.
- Level shifter cells shifting incorrect voltage differences
- Level shifters with an input pin driven by pins operating at different voltages
- Level shifters with an output pin driving pins operating at different voltages
- Level shifters that violate level shifter strategy settings.

`-power_nets`

Reports on power and ground pin connections. The report provides the following information:

- Mismatches of power and ground pins between logical libraries and FRAM libraries
- Power and ground connection summary
- Power and ground pins whose connection cannot be derived and why the derivation fails.
- Power and ground pins whose existing connection does not match with the derived connection from the power domains. This failure for power and ground pins may be caused by any of the following:
 - Unknown power pin type indicates either the power pin does not have a type

or the pin has a type without connection semantics. The automatic power connection supports the following power pin types: `primary_power_pin`, `primary_ground_pin`, `backup_power_pin`, `backup_ground_pin`, `internal_power_pin`, and `internal_ground_pin`.

- Invalid pin type for the cell or the cell's power domain indicates that the cell's power domain does not have a power net connection with the matching type for the power pin type. For example, a backup ground pin of a regular cell will have such an issue if the cell's power domain does not have a backup ground net connection.

- No signal pin that uses this pin as `related_power_pin`/`related_ground_pin` occurs only on level shifters or isolation cells. The power connection of a power pin on level shifters or isolation cells is obtained through tracing cells connected to related signal pins of the power pin. The power connection fails if a power pin has no related signal pin; such as if no signal pin uses the power pin as `related_power_pin` or `related_ground_pin` in the logical library cell definition.

- Cells connected to the related signal pins of the LS/ISO cell requiring different P/G nets: occurs only on level shifters or isolation cells. The tool cannot derive a proper power net for a given pin because different nets are needed for the same pin based on cells connected to related signal pins.

- Back-to-back connection of LS/ISO cells occurs when a level shifter or isolation cell is directly connected to another level shifter or isolation cell. The automatic connection of a power pin on such level shifters or isolation cells may fail if the tracing of related signal connections reaches only other level shifters or isolation cells.

The above issues can be resolved using the **`connect_power_net_info`** command.

`-max_messages message_count`

Sets the limit on the number of messages printed in the log file.

DESCRIPTION

The **`check_mv_design`** command checks the design, multivoltage constraints, electrical isolation requirements, and connection rules, and issues error and warning messages as appropriate.

The checker options can be combined to adjust the verbosity of the final report. If the command is specified without a checker option, a summary of all violations is reported.

If the **`-verbose`** option is specified, details of all violations are reported. The **`-max_messages`** option reports a specified number of violations. When other checker options are specified, the message count specified by **`-max_messages`** applies to each checker. If a checker is not specified, the message count is used to limit the number of messages printed by all checkers. If **`-max_messages`** is not specified, the command prints all messages.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following command prints all violations in the design:

```
prompt> check_mv_design -verbose
```

```
-----  
Target Library Subset Checks  
-----
```

No Errors/Warnings Found.

```
-----  
Power Domain Checks  
-----
```

Warning: Isolation cell is required on net A_INST/OUTPUT connecting A_INST and B_INST. (MV-042)

Warning: Isolation cell is required on net B_INST/OUTPUT connecting B_INST and top. (MV-042)

Warning: Isolation cell A_INST/iso1 is used to isolate a net that is not crossing power domain boundary. (MV-044)

```
-----  
Power Domain Checks Summary  
-----
```

Warning: Found 2 net(s) without isolation. (MV-046)

Warning: Found 1 isolation cell(s) on net(s) not crossing power domain boundaries. (MV-048)

```
-----  
Cell Operating Condition Checks  
-----
```

No Errors/Warnings Found.

```
-----  
Power Domain and Operating Condition Consistency Checks  
-----
```

No Errors/Warnings Found.

The following command prints a summary of all violations in the design:

```
prompt> check_mv_design
```

```
-----  
Target Library Subset Checks  
-----
```

No Errors/Warnings Found.

Power Domain Checks

Warning: Found 2 net(s) without isolation. (MV-046)

Warning: Found 1 isolation cell(s) on net(s) not crossing power domain boundaries. (MV-048)

Cell Operating Condition Checks

No Errors/Warnings Found.

Power Domain and Operating Condition Consistency Checks

No Errors/Warnings Found.

The following command prints violations related to electrical isolation:

prompt> **check_mv_design -verbose -isolation**

Power Domain Checks

Warning: Isolation cell is required on net A_INST/OUTPUT connecting A_INST and B_INST. (MV-042)

Warning: Isolation cell is required on net B_INST/OUTPUT connecting B_INST and top. (MV-042)

Warning: Isolation cell A_INST/iso1 is used to isolate a net that is not crossing power domain boundary. (MV-044)

Power Domain Checks Summary

Warning: Found 2 net(s) without isolation. (MV-046)

Warning: Found 1 isolation cell(s) on net(s) not crossing power domain boundaries. (MV-048)

SEE ALSO

check_library(2)
remove_target_library_subset(2)
set_target_library_subset(2)

check_mv_design

check_noise

Checks whether there are necessary data available to run noise analysis in the current design.

SYNTAX

```
status check_noise  
[-verbose]  
[-nosplit]  
[-include check_list]
```

Data Types

check_list list

ARGUMENTS

- verbose**
Enables verbose mode showing detailed information and pin names.
- nosplit**
Most of the design information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column. The **-nosplit** option prevents line-splitting and facilitates writing software to extract information from the report output.
- include *check_list***
Specifies the types of checking. Available values are **noise_driver** and **noise_immunity**. The default is **noise_immunity**.

DESCRIPTION

It is possible to have invalid noise models in a library and run noise analysis without detecting them. This may result in inaccurate results. If the design is large, and it takes long time to finish the noise analysis, it will also cause longer turn around time. Moreover, it is very important that all pins in the design have correct noise immunity information. Otherwise, when the noise analysis is over, violations wouldn't be reported even if noise bumps are large enough to create violations.

The **check_noise** command can be executed before the noise analysis to validate the correctness of a design with respect to noise analysis. It checks for the presence of any invalid noise model, and any pin without a model from the library and from the design. It verifies that all pins in the design are 'noise constrained', i.e., their noise immunity can be calculated.

In a verbose mode, the command shows the names of cells and pins that have no model or an invalid model.

EXAMPLES

The following example shows a noise immunity check:

```
prompt> check_noise
```

```
Information: Checking noise immunity on load pins...
```

```
Noise immunity type above_low below_high -----  
----- library immunity table 0 0 library immunity curve 0 0 library CCS  
noise immunity 3245 3245 global threshold 0 0
```

The following example shows a noise driver check:

```
prompt> check_noise -include {noise_driver}
```

```
Information: Checking noise models on driver pins...
```

```
Noise driver type above_low below_high -----  
----- library set iv curve 0 0 library set CCS noise 520 520 estimation set  
value 0 0
```

SEE ALSO

```
report_noise(2)
```


check_physical_constraints

Checks the physical constraints and provides feedback about possible errors in input.

SYNTAX

```
int check_physical_constraints  
[-narrow_placement_area no_of_sites]  
[-rc_check rc_variation_margin]  
[-verbose]  
[-analyze_legality [-nworst no_of_worst_cells]]  
[-design]  
[-lib_cell lib_cell_name]]
```

Data Types

<i>no_of_sites</i>	integer
<i>rc_variation_margin</i>	float
<i>no_of_worst_cells</i>	integer
<i>lib_cell_name</i>	string

ARGUMENTS

-narrow_placement_area *no_of_sites*

Issues warnings about narrow placement areas in the floorplan. It is defined in terms of base site widths. The default is the maximum width of the cells in the library. Using a value of 0 for this switch causes the narrow placement area check to be skipped. The narrow placement area threshold width is same for both the vertical and the horizontal direction.

-rc_check *rc_variation_margin*

Issues warnings if the resistance or capacitance varies by more than the *rc_variation_margin* across different metal layers. This option detects invalid layers, such as POLY. However, such a variation is possible because of differing metal widths in different layers. Also checks for width, pitch, and spacing for all route metal layers. Default is 5.0. Note that *rc_variation_margin* value is a multiplier and not an absolute value.

-verbose

Prints more information. When used with the **-analyze_legality** switch, it prints the legal site rates for each library cell and the reason for non-legal sites.

-analyze_legality

Analyzes each cell of a physical library against a floorplan (with power structures) and reports the success rate for legalization on all sites. The legal rate printed prints the legal sites for the library cell after evaluating them against the power straps. It shows the total number of available sites for the library cell so you can determine whether or not you want to use that cell. Default is off.

-nworst *no_of_worst_cells*
Prints *no_of_worst_cells* for all cells in the physical library in terms of legal site rates. This is the default mode of the **-analyze_legality** switch. If it is used with **-verbose** or **-lib_cell**, it is ignored. This option is only used with **-analyze_legality**. Default is 10.

-design
Performs **-analyze_legality** on the library cells that are in design only. Default is false.

-lib_cell *lib_cell_name*
Specifies a particular library cell for **-analyze_legality**.

DESCRIPTION

The **check_physical_constraints** command checks several physical constraints and provide information about possible errors in input. It checks for cell areas in hard bound, correct layers in the library against those in the floorplan, resistance and capacitance for different route layers, narrow placement areas in the floorplan, legal sites for library cells in floorplan, etc. Check all of the warnings and error messages to avoid erroneous input to the tool. Use this command early in the flow to detect errors in input.

Refer to the **-check_only** option to the **physopt** for further information.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example checks the physical constraints. The *rc_variation_margin* is 2.1.

```
prompt> check_physical_constraints -verbose -rc_check 2.1
```

SEE ALSO

`report_placement_utilization(2)`

check_physical_design

Checks the readiness of the current design for IC Compiler.

SYNTAX

```
status check_physical_design  
-stage stage_value | -input_log log_file  
[-output directory]  
[-display]
```

Data Types

<i>stage_value</i>	string
<i>log_file</i>	string
<i>directory</i>	string

ARGUMENTS

- stage *stage_value*
Specifies which stage to check. The checks performed by the **check_physical_design** command depend on the specified stage. Make sure the design contents make sense for the specified stage; otherwise, the report might be meaningless.
You can specify one of the following values:
- * pre_place_opt
This stage requires that the floorplan and netlist data are ready and the design constraints are set.
 - * pre_clock_opt
This stage requires the same items as pre_place_opt. In addition, the design must be placed and clock constraints must be set.
 - * pre_route_opt
This stage should be called when the design is ready for routing.
You must specify either this option or the **-input_log** option.
- input_log *log_file*
Specifies the log file to analyze. When you specify this option, the command only analyzes the specified log file; it does not invoke special checks.
You must specify either this option or the **-stage** option.
- output *directory*
Specifies the the directory in which to save the report.
By default, the report is saved in a directory called cpd_<stage>_<time>_<pid>. The generated HTML report is called index.html.
If the specified directory already exists, a unique directory named *directory.n* is created and used.
- display
Displays the result in the Web browser defined by the **gui_online_browser**

variable.

DESCRIPTION

The **check_physical_design** command checks the readiness of the current design for IC Compiler. It performs specific checks and analysis based on the specified options. If you specify the **-input_log** option, the tool skips the checking and analyzes the specified log file instead.

If you do not specify the **-input_log** option, you must load the floorplan and open the Milkyway design before running the command; otherwise, the checking will not work. In addition, some of the checks require the CEL view to be writable.

This command analyzes and reports all the error messages and other customizable messages from the checking result (or the input log file specified by the **-input_log** option) and provides the final report in an easy-to-use HTML format. The HTML file contains information about what to do next, hyperlinks to man pages, and the line number in the log file for each message. The command supports Web browsers such as Internet Explorer and Firefox. By default, this command tries to open the result in the Web browser.

Configuration File

You can customize the behavior of the **check_physical_design** command by using a configuration file, which is called `.icc_report_config`. The tool looks for this file in your home directory and the current working directory. Settings in the configuration file in the current working directory override those in the configuration file in your home directory.

The configuration file can contain the following commands:

`set maxMessageCount integer` This variable controls how many detailed messages are displayed for the same message ID in the HTML file. The default is 10.

`set linkToLogMode integer` This variable controls the algorithm used to link the log file in the HTML file. If set to 0, the HTML file uses JavaScript to automatically generate the partial log file and highlights the desired message line when you click on the hyperlink to the log file. If set to 1, multiple partial log files are generated disk during command execution and the links point directly to the saved files. A setting of 1 has a small runtime and disk space penalty, due to saving extra files on disk, but the speed of opening the links should be better than with a setting of 0. If your browser does not support JavaScript inside the HTML, use a setting of 1. The default is 0.

`add_keyword keyword` This command allows you to specify keywords that you want the tool to be aware of. The tool creates an HTML table with the results of "grepping" for this keyword. When you use this setting, your operating system path must be set up so that it can directly find the grep command.

`config_message options` This command defines how the **check_physical_design** command treats messages during analysis. Note that error messages are always considered important messages.

The supported options for the **config_message** command are:

- * **-id *message_id***
Specifies the message ID which will be reconfigured.
- * **-title *message_id_title***
Specifies the title of the message, which is displayed in the HTML table.
- * **-ignore**
Specifies that the **check_physical_design** command should treat this message ID as an unimportant message.
- * **-stage *stage_name***
Specifies the stage to which this configuration applies. If not specified, the configuration applies to all stages. This option allows you to treat the same message differently at different stages. For example, you can define the messages from checking ideal clock nets as unimportant at the *pre_clock_opt* stage, but define them as important at the *pre_route* stage.

config_check *options* This command adds user-defined checks to the **check_physical_design** command. The checks are executed one by one after the embedded checks.

The supported options for the **config_check** command are

- * **-cmd *tcl_procedure_name***
Specifies the Tcl procedure that contains the user-defined check.
- * **-stage *stage_name***
Specifies the stage to which this configuration applies. If not specified, the configuration applies to all stages. This option allows you to add the same check at different stages.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example checks the design readiness for placement and saves the result in the *cpd_result* directory. The log file lists the checks that were performed and the result summary. An HTML file is also generated, which you can read using the Web browser.

```
prompt> check_physical_design -stage pre_place_opt -output cpd_result
checking tluplus...
checking link...
checking logic objects...
```

```

checking physical objects...
checking timing...
checking HFN/ideal/dont_touch nets...
checking placement constraints...
*****
Report : check_physical_design
Design : RISC_CHIP
Version: B-2008.09-ICC-SP
Date   : Wed Oct  8 10:55:22 2008
*****
Total messages: 0 errors, 270 warnings

```

Other Warning Summary for check_physical_design

ID	Occurances	Title

FEAS-002	1	High Fanout Synthesis has not been performed
LINT-1	3	cell does not drive any nets
LINT-2	129	Net driven by pin has no loads
LINT-3	4	Net has no drivers.
LINT-5	4	In design '%s', output port '%s' is not driven.
LINT-8	88	Input port unloaded
LINT-		
32	5	In design '%s', a pin on submodule '%s' is conn...
LINT-		
33	1	In design '%s', the same net is connected to mo...
LINT-63	32	Net '%s' has a single tri-state driver.
PNR-156	2	macro cell has no keepout margin defined
PSYN-		
261	1	Capacitance of layer %s varies more than the sp...

```

dump check_physical_design result to file cpd_result/index.html
1
prompt> sh firefox cpd_result/index.html

```

The following example analyzes the results in the pre_place_log log file. It does not perform checks on the design.

```

prompt> check_physical_design -input_log pre_place_log

```

The following example shows a sample configuration file:

```

add_keyword drc
config_message -stage pre_clock_opt -id CTS-810 -ignore
config_check -stage pre_clock_opt -cmd "report_design -physical"
set maxMessageCount 5

```

SEE ALSO

check_design(2)

```
check_physical_constraints(2)
check_error(2)
get_message_info(2)
print_message_info(2)
print_suppressed_messages(2)
suppress_message(2)
unsuppress_message(2)
gui_online_browser(3)
```

check_route

Checks and reports the violations of a routed design.

SYNTAX

```
status check_route
[-drc]
[-opens]
[-antenna]
[-top_layer_probe_constraints]
[-num_cpus num]
```

ARGUMENTS

-drc
Check DRC Violations.

-opens
Check Opens.

-antenna
Check Charge-Collecting Antenna violations.

-top_layer_probe_constraints
Check Top-Layer Probe Constraints.

-num_cpus *num*
Number of cpus for DRC checking. Number must be ≥ 1 .

DESCRIPTION

This command checks and reports the following about a routed design: DRC Violations, Opens, Charge-Collecting Antenna violations and Top-Layer Probe Constraints. Design rule checking can also be done using the Distributed Routing option.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

```
prompt> check_route -drc
```

SEE ALSO

check_routeability

Verifies that the current design is routeable.

SYNTAX

```
integer check_routeability  
[-error_cell cell_name]
```

ARGUMENTS

`-error_cell cell_name`
Enter the error cell you want or use the default name provided. The default name is `<top_cell_name>.err`. When there are errors in the design, an error cell is generated with this given name. An error cell will not be generated unless errors exist.

DESCRIPTION

Checks pin access points, cell instance wire tracks, pin out of boundaries, min-grid and pin design rules and blockages to ensure they meet the design requirements. It performs a check of the design for optimization in order to substantiate any errors in the design that might need to be fixed or what could help to improve the design. This must currently be run on a placed design.

You can use this command at every stage between placement and detail routing. Verify errors in the generated error cell or log file.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example dump out the design error to a specified file, `dumpErr.err`.

```
prompt> check_routeability -error_cell dumpErr.err
```

The following example dump out the design error to a default cell.

```
prompt> check_routeability
```

SEE ALSO

`create_route_guide(2)`

check_rp_groups

Checks the relative placement constraints and reports the failures.

SYNTAX

```
collection check_rp_groups
{rp_groups | -all}
[-output filename]
[-verbose]
```

Data Types

<i>rp_groups</i>	list or collection
<i>filename</i>	string

ARGUMENTS

rp_groups
Specifies the relative placement groups that will be checked for failures. This option and the **-all** option are mutually exclusive.

-all
Specifies that all relative placement groups will be checked. This option and the *rp_groups* argument are mutually exclusive.

-output *filename*
Specifies the name of the output file for the report. If you do not specify this option, the report is written to the standard output.

-verbose
Reports all possible relative placement failures in the design.

DESCRIPTION

The **check_rp_groups** command checks for and reports any relative placement failures in the specified groups. This command is supported only for designs that do not contain multiply instantiated designs.

The command returns a collection containing the relative placement groups that are checked. If no groups are checked, the empty string is returned.

Relative placement usage is available with Design Compiler Ultra.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following examples use **check_rp_groups** to check for relative placement failures:

```
prompt> get_rp_groups
{compare::g1 compare::g2 compare::g4 compare::g7}

prompt> check_rp_groups -all -out failures
{compare::g7 compare::g4 compare::g2 compare::g1}

prompt> check_rp_groups compare::g4 -out g4_failures
{compare::g4}

prompt> check_rp_groups compare::g4

*****
Report: The relative placement groups not meeting all constraints but placed.
Version: B-2008.09
Date: Mon Sep 11 04:44:34 2008
Number of relative placement groups: 1
*****

relative placement GROUP: compare::g4
-----
Warning: Possible placement failure of relative placement group 'compare::g4'
(RPGP-027)
{compare::g4}

prompt> check_rp_groups compare::g1
*****
Report: The relative placement groups that could not be placed.
Version: B-2008.09
Date: Mon Sep 11 04:47:25 2008
Number of relative placement groups: 1
*****

relative placement GROUP: compare::g1
-----
Warning: Possible placement failure of relative placement group 'compare::g1'
(RPGP-027)
{compare::g14}
```

SEE ALSO

```
add_to_rp_group(2)
create_rp_group(2)
remove_rp_groups(2)
```

check_scan_chain

Allows scan chain structural consistency checking based on the scan chain information stored in the current design.

SYNTAX

```
status check_scan_chain  
[-chain_name name_of_the_scan_chain_string]
```

ARGUMENTS

DESCRIPTION

This command performs a scan chain structural checking based on the scan chain information stored in the current CEL. If the scan chain passes all the structural checks, then the scan chain status is "VALIDATED"/V and is physically design-for-test (DFT) optimized. If the scan chain fails the structural checks then the scan chain status is "FAILED"/F and is not physically optimized.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example checks for scan chain structural consistency.

```
prompt> check_scan_chain  
Checking SCANDEF...  
  Checking scan cell correspondence between SCANDEF and netlist...  
  Checking scan chain checking...  
All checks completed.
```

```
Report : Scan DEF check  
Design : E2080_cpu  
Version: Z-2007.03-ICC-BETA1  
Date   : Tue Jan  9 14:37:48 2007
```

```
Information from SCANDEF file:  
Number of SCANCHAINS: 2
```

```
Checking between SCANDEF file and design:  
Total SCANCHAINS checked: 2  
VALIDATED : 2  
FAILED    : 0
```

Chain name	Status	#cells	Scan IN	Scan OUT
-----	-----	-----	-----	-----
1	V	62	U10/C	alu1/U381/A
2	V	61	U11/C	U9/B

1

The following example checks for the scan chain structural consistency of scan chain 2.

```
prompt> check_scan_chain -chain 2
```

```
Checking SCANDEF...
```

```
Checking scan cell correspondence between SCANDEF and netlist...
```

```
Checking scan chain checking...
```

```
All checks completed.
```

```
*****
```

```
Report : Scan DEF check
```

```
Design : E2080_cpu
```

```
Version: Z-2007.03-ICC-BETA1
```

```
Date   : Tue Jan  9 14:38:37 2007
```

```
*****
```

```
Information from SCANDEF file:
```

```
Number of SCANCHAINS: 2
```

```
Checking between SCANDEF file and design:
```

```
Total SCANCHAINS checked: 2
```

```
VALIDATED : 2
```

```
FAILED    : 0
```

```
Information on SCANCHAIN 2:
```

```
STATUS: V
```

```
Examining design SCANCHAIN 2:
```

```
STOP: U9/B
```

```
thru: statemachine1/counter_reg_3_/SI
```

```
thru: statemachine1/counter_reg_2_/SI
```

```
thru: statemachine1/counter_reg_1_/SI
```

```
thru: statemachine1/counter_reg_0_/SI
```

```
START: U11/C
```

1

SEE ALSO

```
read_def(2)
```

```
optimize_dft(2)
```

```
report_scan_chain(2)
```

```
trace_scan_chain(2)
```

check_scenarios

Performs consistency checks between scenarios for scenario specific information such as TLU_plus, operating conditions on the libraries, clocks and so on.

SYNTAX

```
status check_scenarios  
[-output ]  
[-display]
```

ARGUMENTS

-output
Specifies the directory to store the output. By default output is stored in the html format in the directory design_dir/check_scenarios_mmddyyyy_pid

-display
Pops up a html browser to display the data. The browser that is to be used is defined by the tcl variable gui_online_browser.

DESCRIPTION

The **check_scenarios** command checks and reports issues with the scenarios. The command performs the following checks: TLU_plus, clock related information such as clock periods, default versus nominal PVT on libraries, PVT from ILMs versus parent, and PVT of library cells. This is a beta capability.

EXAMPLES

The following is an example of the **check_scenarios** command: prompt> **check_scenarios**
check_scenarios Warning: The complexity of the scenario (s1) might cause long runtime and high memory usage. (MCMM-212) Warning: The complexity of the scenario (s2) might cause long runtime and high memory usage. (MCMM-212) Warning: Scenario (s1) has instances which referenced library cells that are marked as "dont_use". (MCMM-225) Warning: Scenario (s2) has a library setup issue with regard to the PVT values and the operating condition. (MCMM-213) Warning: Scenario (s2) has instances which referenced library cells that are marked as "dont_use". (MCMM-225) Information: HTML report can be found at /WORKING_DIRECTORY/run/check_scenarios_7132009_21451/check_scenarios_0.html

SEE ALSO

gui_online_browser(3)

check_target_library_subset

Checks and prints out the inconsistent settings among target library, target library subset, and operating conditions.

SYNTAX

```
status check_target_library_subset
```

ARGUMENTS

The **check_target_library_subset** command has no arguments.

DESCRIPTION

This command checks and prints out the inconsistent settings among target library, target library subset, and operating conditions. It checks the following:

- Conflicts between the target library subset and the global target_library variable. Target library subset should be a subset of the library set being specified by the target_library variable.
- Conflicts between operating condition and target library subset. There should be at least one library from the target library subset that has the same process, nom_voltage, and temperature as the operating condition being used on that block.
- Conflicts between the library cell of the mapped cell and target library subset. Gives a warning if the library cell of a mapped cell is not from any of the libraries of the target library subset. Note that this is not an error, because the subset applies only to new cells being created during optimization, not to existing cells in the block.

Multicorner-Multimode Support

Depending on the options used, this command either uses the current scenario or has no dependency on scenario-specific information.

SEE ALSO

```
check_library(2)  
remove_target_library_subset(2)  
report_target_library_subset(2)  
set_operating_conditions(2)  
set_target_library_subset(2)  
target_library(3)
```

check_timing

Checks for possible timing problems in the current design.

SYNTAX

```
status check_timing
[-overlap_tolerance minimum_distance]
[-override_defaults check_list]
[-multiple_clock]
[-retain]
[-include check_list]
[-exclude check_list]
```

Data Types

<i>minimum_distance</i>	float
<i>check_list</i>	list

ARGUMENTS

-overlap_tolerance *minimum_distance*

Specifies the minimum distance allowed between the master close edge and the slave open edge. If the distance is less than this value, the tool issues a warning. Use this option to check for the master-slave clock overlap. By default, this option is off.

-override_defaults *check_list*

Overrides the checks in `timing_check_defaults` by using the argument specified in *check_list*. If the **-override_defaults** option is used with a check list, the final list of checks to be performed is the one in the *check_list* argument of the **-override_defaults** option.

-multiple_clock

Issues a warning when multiple clocks reach a register clock pin. If more than one clock signal reaches a register clock pin, and the **timing_enable_multiple_clocks_per_reg** variable is set to false, then the clock to use for analysis is undefined, and the tool generates a warning message. In this case, use either the **set_case_analysis** or **set_disable_timing** command so that only one clock can propagate from the sources to the register clock pin. By default, this option is off.

-retain

Provides a warning if any of the RETAIN values is larger than its corresponding delay value. The RETAIN values should be less than their corresponding delay values so that they can be considered for hold violations. Otherwise, the RETAIN values might be considered for setup violations. By default, this option is off.

-include *check_list*

Adds the checks listed in *check_list* to the checks defined by the **timing_check_defaults** variable.

`-exclude check_list`

Subtracts the checks listed in *check_list* from the checks defined by the **timing_check_defaults** variable.

DESCRIPTION

This command checks the timing attributes placed on the current design and issues warning messages as needed. The messages provide information that identifies and corrects potential errors. The warnings do not necessarily indicate design problems.

This command without any options performs the checks defined by the **timing_check_defaults** variable. To change the value of the variable, redefine it. If the **-override_defaults** option is not used, the final list of checks to be performed is the list of checks specified by the **timing_check_defaults** variable, plus the list of checks given by the **-include** option, minus the list of checks given by the **-exclude** option. If the **-override_defaults** option is used with a *check_list*, the final list of checks to be performed is the checks in the *check_list* given by the option.

The alphabetically ordered list below shows the meaning of each check:

loops

Warns of combinational feedback loops. If the feedback loop is not broken by the **set_disable_timing** command, it is automatically broken by disabling one or more timing arcs.

no_input_delay

Warns if no clock related delay is specified on an input port, where it propagates to a clocked latch or output port. If the **timing_input_port_default_clock** variable is set to true, a default clock will be assumed for the input port. Otherwise it will not be clocked, and the paths are unconstrained. In this case, if there is no input delay specified, **check_timing** will not generate warnings.

unconstrained_endpoints

Warns about unconstrained timing endpoints. This warning identifies timing endpoints that are not constrained for maximum delay (setup) checks. If the paths to the endpoint are all false paths, endpoints will not be reported as unconstrained endpoints.

generated_clock

Checks the generated clock network. Three types of issues are reported:

- The source pin (master point) is not the clock source.
- The definition point of the generated clock has no path to the source point.
- The generated clocks form a loop.

pulse_clock_cell_type

Warns if an instance cell has a mismatched pulse type defined from its library cell.

clock_crossing

Checks clock interactions when there are multiple clock domains. If a clock

launches one or more paths that are captured by other clocks, it will have an entry in the clock crossing report. If all paths between two clocks are false paths or they are exclusive or asynchronous clocks, the path is marked with an asterisk (*). If only some of the paths are set as false paths, the path is marked by a number sign (#).

`data_check_multiple_clock`

Warns if multiple clocked signals reach a data check register reference pin. The analysis will be done separately for each of the clocked domains. If the **timing_enable_multiple_clock_per_reg** variable is set to FALSE, only one of the clocked signals will be analyzed.

`data_check_no_clock`

Warns if no clocked signal reaches a data check register reference pin. In this case, no setup or hold checks are performed on the constrained pin.

`multiple_clock`

This item should also be specified by the **-multiple_clock** option.

`generic`

Warns about generic (unmapped) cells in the design. The timing of paths through generic cells is inaccurate because generic cells have zero delay.

`gated_clock`

Warns about the gated clocks. Disable the gating timing arcs only if the **propagated_clock** attribute is set on that clock.

`ideal_timing`

Warns user_defined ideal ideal transition or latency is set on a normal pin (not ideal).

`retain`

This check item should also be specified by the **-retain** option.

`clock_no_period`

Warns if clocks with no period are specified.

The warning messages that can occur when using this command are described below.

- The warning message shown below occurs when the waveforms applied to a master-slave register are overlapping or have different periods. Use the **create_clock** command to modify one of the waveforms.

The overlap check first determines the intended master-slave waveform relationships based on the ideal clock waveforms. Then the clock network delay and uncertainty is applied to the waveforms and if the distance between the related edges is less than the overlap tolerance, the tool issues a warning.

WARNING: The following master-slave registers have overlapping clock violations. The waveforms or periods may be invalid.

- The warning message shown below identifies timing endpoints (output ports and register data pins) that are not constrained. If the endpoint is a register data pin, use the **create_clock** command for the appropriate clock source to constrain the pin. Use the **set_output_delay** or **set_max_delay** command to constrain output ports.

`check_timing`

The **set_output_delay** command constrains only the path when the delay is relative to a clock.

WARNING: The following endpoints are not constrained for maximum delay.

- The warning message shown below identifies gated clocks. Disable the gating timing arcs only if the **propagated_clock** attribute is set on that clock.

WARNING: The clock network starting at "clk" is gated by the following input pins. The gating timing arcs might need to be disabled for clocks with the "propagated_clock" attribute.

- The warning message shown below occurs when the design contains unmapped cells (generic logic), which causes the timing of the paths through the generic cells to be inaccurate because generic cells have zero delay.

WARNING: Design "*design_name*" contains unmapped cells.

ideal_clocks

Warns any clock networks that are ideal. Generally, all clocks should be propagated so that the clock network timing is accurately calculated. Especially, in presence of crosstalk, the delay changes induced by other nets on the clock network will not be reflected in the calculated slacks in the design.

no_driving_cell

Warns any port that does not have a driving cell. The warning is issued only when the net connected to the port has parasitic. When no driving cell is specified, that net is assigned a strong driver for modeling aggressor effects, which can be pessimistic. Also, a port with no driving cell could act as a strong victim, which could underestimate the crosstalk effect.

partial_input_delay

Warns any ports have partially defined input delay, only the minimum or only the maximum delay defined with **set_input_delay**. As a result, some paths starting from the port with partially defined input delay may become unconstrained and some potential violations could be missed.

net_no_driving_info

Warns any net that does not have driving pins or there is no timing arcs on the driver pins. The warning is issued only when the net has coupled parasitic.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example checks the timing of the current design and issues warnings as needed:

```
prompt> check_timing
```

Information: Checking 'unexpandable_clocks'...
Information: Checking 'generic'...
Information: Checking 'latch_fanout'...
Information: Checking 'loops'...
Information: Checking 'generated_clocks'...

SEE ALSO

check_design(2)
check_library(2)
create_clock(2)
current_design(2)
set_case_analysis(2)
set_disable_timing(2)
set_max_delay(2)
set_output_delay(2)
timing_enable_multiple_clocks_per_reg(3)

check_tlu_plus_files

Checks the files used for TLUPlus extraction.

SYNTAX

status **check_tlu_plus_files**

ARGUMENTS

This command has no arguments.

DESCRIPTION

The **check_tlu_plus_files** command invokes consistency checks on the files used for virtual route and post route extraction using TLUPlus. The min and max TLUPlus files were specified with the **set_tlu_plus_files** command. If the tlu+ is from the Milkyway design library attachment, then this command will not do any checking because the attachment should be already checked when it is attached.

The following items are checked for consistency:

layer names

The consistency check ensures that the conductor and via layer names are consistent across the Milkyway, ITF, and mapping files. The tool also checks that the number of layers are consistent among the ITF and Milkyway files. Error messages are printed if there are inconsistencies.

etch values

The tool checks that the etch values in ITF files are consistent with the delta width in Milkyway technology files under min/nom/max conditions, respectively. The signs used to indicate expand and shrink are opposite in ITF files and Milkyway technology files. Warning messages appear if inconsistencies are detected.

min-width and min-spacing

The tool checks that min-width and min-spacing are consistent between ITF files and Milkyway technology files, and among different min/nom/max ITF files. Inconsistencies cause warning messages to be issued.

conductor thickness

The tool checks that the conductor thicknesses are consistent between ITF files and Milkyway technology files. Inconsistencies cause warning messages to be issued.

Multicorner-Multimode Support

This command uses information from both active and inactive scenarios.

EXAMPLES

The following example runs the consistency checks:

```
prompt> check_tlu_plus_files
```

SEE ALSO

```
check_library(2)  
set_tlu_plus_files(2)
```

clock_opt

Performs clock tree synthesis, routing of clock nets, extraction, optimization, and hold-time violation fixing on the design. There is also an option to execute interclock delay balancing.

SYNTAX

```
status clock_opt
[-only_psyn]
[-fix_hold_all_clocks]
[-inter_clock_balance]
[-update_clock_latency]
[-operating_condition min | max(default) | min_max]
[-only_cts]
[-optimize_dft]
[-no_clock_route]
[-only_hold_time]
[-area_recovery]
[-size_only]
[-in_place_size_only]
[-power]
```

ARGUMENTS

-only_psyn
Performs optimization only.

-fix_hold_all_clocks
Executes hold time violation fixing for all clocks. during incremental optimization. This option is off by default.

-inter_clock_balance
Execute interclock delay balancing.

-update_clock_latency
Updates the latencies on real and virtual clock objects after clock tree synthesis, clock tree optimization, interclock balance (if chosen) and clock tree detailed routing. This will effectively execute a `set_clock_latency` under-the-hood for the clock objects and use the insertion delay of the clock tree as the latency number. If `set_latency_adjustment_options` command is executed before `clock_opt`, then the directives will be obeyed. If no directives are given, then only the latencies of real clock objects will be update. The update mechanism uses the insertion delay of the clock tree as the latency number.

-operating_condition [min | max(default) | min_max]
Specify operating condition, default is max.

-only_cts
Perform CTS, CTO and clock routing.

`-optimize_dft`

Enables clock-aware scan reordering. The reordering aims to minimize the number of buffer crossings in the scan chain. Minimize the number of buffer crossings can reduce hold time violations in the scan chain. For best results you should first use `place_opt -optimize_dft` to perform placement aware scan reordering.

`-no_clock_route`

Don't perform clock routing of clock nets.

`-only_hold_time`

Perform hold time fixing only after CTS. If `-fix_hold_all_clocks` is specified, we will fix hold for all clocks. Otherwise, we only fix hold time violations for user specified hold time fix clocks.

`-area_recovery`

Enables area recovery for the cells not on the timing critical paths. By default, this option is off.

`-size_only`

Restricts post CTS optimization to sizing changes only. Optimization tricks that insert new cells and remove cells are disabled. The **-size_only** and **-in_place_size_only** options are mutually exclusive. See the description for the **-in_place_size_only** option for additional information on how to constrain sizing changes more.

`-in_place_size_only`

Restricts post CTS optimization to sizing changes only. Optimization procedures that insert new cells and remove cells are disabled. With the **-in_place_size_only** option, sizing changes are further constrained for minimal engineering change order (ECO) placement changes. For example, a cell is sized to improve timing or design rule costs only if the newly-sized cell can fit into any available space adjacent to the original cell location. The resulting transformation is verified to ensure it is legal.

`-power`

Performs pre or post CTS power optimization in **clock_opt**. The power optimization features are determined by **set_power_options**. During post CTS optimization phase, currently only leakage power optimization is supported. To enable pre CTS power optimization, at least one of `-low_power_placement` and `-clock_gating` must be enabled by **set_power_options**. Detailed options for pre CTS power optimizations are set by **set_optimize_pre_cts_power_options**. When **-low_power_placement** is enabled, switching-activity-based power-aware placement is utilized in an incremental way. For the maximum power optimization performance from this power-aware placement, the same coarse placement setups (e.g. `placer_max_cell_density_threshold`) used in **place_opt** should be specified before running **clock_opt**. When the **-power** option is used with the **-only_cts** option, pre CTS power optimization is done before CTS. When the **-power** option is used with the **-only_psyn** option, only post CTS power optimization is done during post CTS optimization.

DESCRIPTION

The **clock_opt** command performs clock tree synthesis, routing of clock nets,

extraction, optimization, and optionally hold-time violation fixing on the current design. If clock tree synthesis fails, or the routing of clock nets fail, the command returns with a value of 0. If `-only_psyn` option is specified, the start point will be from clock routing, then run extraction and optimization. This can be used in a user-customized CTS flow where user performs clock tree synthesis outside of `clock_opt`.

Prior to `clock_opt`, **`set_clock_tree_options`** can be used for controlling `compile_clock_tree`. **`set_latency_adjustment_options`** can be used to issue directives to control the adjustment of latency on clock objects belonging to virtual and real clocks. If no directives are given, the default `clock_opt` flow will update the latencies of real clocks with their insertion delays obtained after `compile_clock_tree`, `optimize_clock_tree`, and optionally `balance_inter_clock_delay`. **`set_inter_clock_delay_options`** can be used for controlling the behavior of interclock delay balancing within `clock_opt`.

Briefly, `clock_opt` performs: - `compile_clock_tree` - `optimize_clock_tree` - `set_propagated_clock` for all clocks from root pin, but keeps the clock object as ideal - perform interclock balance if so directed - set clock buffers as fixed - updates latency on clock objects if so directed

Multicorner-Multimode Support

This command uses information from the clock tree synthesis scenario, as well as the active scenarios.

EXAMPLES

The following command performs clock tree synthesis, routing of all clock nets, extraction, and optimization:

```
prompt> clock_opt
```

SEE ALSO

`extract_rc(2)`
`psynopt(2)`
`place_opt(2)`

close_distributed_route

Closes all of the sockets and shuts down the daemons.

SYNTAX

integer **close_distributed_route**

DESCRIPTION

Closes all of the sockets and shuts down the daemons on the slave machines.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

SEE ALSO

set_distributed_route(2)
report_distributed_route(2)
remove_distributed_route(2)

close_mw_cel

Closes the specified Milkyway designs.

SYNTAX

```
status close_mw_cel  
[-save]  
[-hierarchy]  
[-all_views]  
[-all_versions]  
[mw_cel_list]
```

Data Types

mw_cel_list list

ARGUMENTS

-save

Indicates that the specified Milkyway designs are to be saved before closing. By default, the command discards any changes made to the Milkyway design and closes the specified Milkyway design.

-hierarchy

Closes the specified top design and all opened child designs. By default, the command closes only the specified Milkyway designs.

When **-hierarchy** is specified, *mw_cel_list* can only specify the top design. The **-hierarchy** argument cannot be used with **-all_views** and **-all_versions**.

-all_views

Closes all opened views of specified Milkyway designs. By default, the command closes only the specified Milkyway designs. If this option is used with **-save**, only views supported by **save_mw_cel** will be saved first and then all opened views will be closed. If specifying this option without **-all_versions**, only the latest versions of all views will be closed. The **-all_views** argument cannot be used with **-hierarchy**.

-all_versions

Closes all versions of specified Milkyway designs. If specifying this option with **-all_views**, all views and all versions of the specified Milkyway designs will be closed, or else only all versions of the specified views will be closed.

The **-all_versions** argument cannot be used with **-hierarchy**.

mw_cel_list

Specifies Milkyway designs to be closed. You can specify Milkyway designs by name, name pattern, or by the Milkyway design collection's name. For example, specifying *top* matches a Milkyway design named *top* in the current library. Specifying *top** matches all Milkyway designs having names beginning with *top*. The command **close_mw_cel [get_mw_cels *]** closes all Milkyway designs in the current library. By default, the command uses the current Milkyway design.

DESCRIPTION

This command saves or discards specified Milkyway designs or the current Milkyway design and then closes them. If the closed Milkyway design is the current Milkyway design, the command automatically clears the Milkyway design.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example discards and closes the current Milkyway design.

```
prompt> close_mw_cel  
1
```

The following example saves and closes a list of Milkyway designs.

```
prompt> close_mw_cel -save {test1 test2}  
Information: Saved design named test1.CEL;1.  
Information: Saved design named test2.CEL;1.  
1
```

The following example discards and closes a list of Milkyway designs.

```
prompt> close_mw_cel {test1 test2}  
1
```

The following example saves and closes all views of the specified Milkyway design.

```
prompt> close_mw_cel -save -all_views test1  
Information: Saved design named test1.CEL;1.  
Information: Saved design named test1.FILL;2.  
1
```

The following example saves and closes all versions of the specified Milkyway design.

```
prompt> close_mw_cel -save -all_versions {test1.FILL}  
Information: Saved design named test1.FILL;1.  
Information: Saved design named test1.FILL;2.  
1
```

The following example saves and closes all views and all versions of the specified Milkyway design.

```
prompt> close_mw_cel -save -all_views -all_versions test1
Information: Saved design named test1.CEL;1.
Information: Saved design named test1.FILL;1.
Information: Saved design named test1.FILL;2.
1
```

The following example saves and closes the specified top design and its opened child designs.

```
prompt> close_mw_cel -hierarchy -save RISC_CORE
Information: Saved design named ALU.CEL;1. (UIG-5)
Information: Saved design named DATA_PATH.CEL;1. (UIG-5)
Information: Saved design named RISC_CORE.CEL;1. (UIG-5)
1
```

SEE ALSO

```
copy_mw_cel(2)
create_mw_cel(2)
current_mw_cel(2)
get_mw_cels(2)
open_mw_cel(2)
remove_mw_cel(2)
rename_mw_cel(2)
mw_cel_collection(2)
list_mw_cels(2)
```

close_mw_lib

Closes the current Milkyway library.

SYNTAX

```
status close_mw_lib  
[-save]
```

ARGUMENTS

-save

Saves all Milkyway cels opened in this library. By default, the command discards changes of Milkyway cels in this library.

DESCRIPTION

This command closes the current Milkyway library. It returns a status indicating success or failure.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example closes the current Milkyway library.

```
prompt> close_mw_lib  
1
```

SEE ALSO

open_mw_lib(2)

Collections_and_Querying

Describes the methodology for creating collections of objects and querying objects in the database.

DESCRIPTION

Synopsys applications build an internal database of the netlist and the attributes applied to it. This database consists of several classes of objects, including designs, libraries, ports, cells, nets, pins, and clocks. Most commands operate on these objects.

By definition

A collection is a group of objects exported to the Tcl user interface.

Collections have an internal representation (the objects), and sometimes, a string representation. The string representation is generally used only for error messages. A set of commands to create and manipulate collections is provided as an integral part of the user interface. The collection commands encompass two categories: those that create collections of objects for use by another command, and one that queries objects for your viewing. The result of a command that creates a collection is a Tcl object that can be passed along to another command. For a query command, although the visible output looks like a list of objects (a list of object names is displayed), the result of a query command is the empty string. An empty string "" is equivalent to the empty collection, that is, a collection with zero elements.

Homogeneous and Heterogeneous Collections

A homogeneous collection contains only one type of object. A heterogeneous collection can contain more than one type of object. Commands that accept collections as arguments can accept either type of collection.

Lifetime of a Collection

Collections are active only as long as they are referenced. Typically, a collection is referenced when a variable is set to the result of a command that creates it, or when it is passed as an argument to a command or a procedure. For example, if in PrimeTime, you save a collection of ports:

```
dc_shell-t> set ports [get_ports *]
```

then either of the following two commands deletes the collection referenced by the *ports* variable:

```
dc_shell-t> unset ports
dc_shell-t> set ports "value"
```

Collections can be implicitly deleted when they go out of scope. Collections go out of scope when the parent (or other antecedent) of the objects within the collection is deleted. For example, if our collection of ports is owned by a design, it is implicitly deleted when the design that owns the ports is deleted. When a collection is implicitly deleted, the variable that referenced the collection still holds a string representation of the collection. However, since the collection is gone, this value is useless, as illustrated in the following example from PrimeTime:

```
dc_shell-t> current_design
{"TOP"}

dc_shell-t> set ports [get_ports in*]
{"in0", "in1"}

dc_shell-t> remove_design TOP
Removing design 'TOP'...

dc_shell-t> query_objects $ports
Error: No such collection '_sel26' (SEL-001)
```

Iteration

To iterate over the objects in a collection, use the **foreach_in_collection** command. You cannot use the Tcl-supplied **foreach** iterator to iterate over the objects in a collection, because **foreach** requires a list, and a collection is not a list. In fact, if you use **foreach** on a collection, it will destroy the collection. The arguments to **foreach_in_collection** are similar to those of **foreach**: an iterator variable, the collections over which to iterate, and the script to apply at each iteration. Note that unlike **foreach**, **foreach_in_collection** does not accept a list of iterator variables. The following example is an iterative way to perform a query:

```
dc_shell-t> \
    foreach_in_collection s1 $collection {
        echo [get_object_name $s1]
    }
```

For details, see the **foreach_in_collection** man page or the user guide.

Manipulating Collections

A variety of commands are provided to manipulate collections.

- **add_to_collection** - This command takes a base collection and a list of element names or collections that you want to add to it. The base collection can be the empty collection. The result is a new collection. In addition, **add_to_collection** allows you to remove duplicate objects from the collection with the **-unique** option.

- **remove_from_collection** - This command takes a base collection and a list of element names or collections that you want to remove from it. For example, in PrimeTime:

```
dc_shell-t> set dports [remove_from_collection [all_inputs] CLK]
{"in1", "in2", "in3"}
```

- **compare_collections** - Verify that two collections contain the same objects (optionally, in the same order). Result is "0" on success.

- **copy_collection** - Creates a new collection containing the same objects (in the same order) as a given collection. Not all collections can be copied.

- **index_collection** - Extracts a single object from a collection and creates a new collection containing that object. Not all collections can be indexed.

- **sizeof_collection** - Returns the number of objects in a collection.

Filtering

You can filter any collection using the **filter_collection** command. This command takes a base collection and creates a new collection that includes only those objects that match an expression.

Further, many of the commands that create collections support a **-filter** option, which allows the objects to be filtered out of the collection before they are ever included in it. Frequently, this is more efficient than filtering after the fact. The following example from PrimeTime filters out all leaf cells:

```
dc_shell-t> filter_collection [get_cells *] "is_hierarchical == true"
{"i1", "i2"}
```

The basic form of a filter expression is a series of relations joined together with AND and OR operators. Parentheses are also supported. The basic relation contrasts an attribute name with a value through a relational operator. In the previous

example, *is_hierarchical* is the attribute, *==* is the relational operator, and *true* is the value.

The relational operators are

```
==    Equal
!=    Not equal
>     Greater than
<     Less than
>=    Greater than or equal to
<=    Less than or equal to
=~    Matches pattern
!~    Does not match pattern
```

The basic relational rules are

- String attributes can be compared with any operator.
- Numeric attributes cannot be compared with pattern match operators.
- Boolean attributes can be compared only with *==* and *!=*. The value can be only *true* or *false*.

See the appropriate man pages for complete details.

Sorting Collections

You can sort a collection using the **sort_collection** command. This command takes a base collection and a list of attributes as sort keys. The result is a copy of the base collection sorted by the given keys. For example, the following PrimeTime command will sort the ports by direction, then by full name.

```
dc_shell-t> sort_collection [get_ports *] {direction full_name}
{"in1", "in2", "out1", "out2"}
```

Sorting is ascending by default, or descending with the **-descending** option.

Implicit Query of Collections

All commands that create collections implicitly query the collection when the command is used at the command line. The number of objects displayed is controlled by the variable **collection_result_display_limit**. Consider the following examples from PrimeTime:

```
dc_shell-t> set_input_delay 3.0 [get_ports in*]
```

```

1
dc_shell-t> get_ports in*
{"in0", "in1", "in2"}
dc_shell-t> query_objects -verbose [get_ports in*]
{"port:in0", "port:in1", "port:in2"}
dc_shell-t> set iports [get_ports in*]
{"in0", "in1", "in2"}

```

In the first example, the **get_ports** command creates a collection of ports that is passed to the **set_input_delay** command. This collection is not the result of the primary command (**set_input_delay**), so it is not queried. The second example shows how a command which creates a collection automatically queries the collection when that command is used as a primary command. The third example shows the verbose feature of **query_objects**, which is not available with implicit query. Finally, the fourth example sets the variable **iports** to the result of the **get_ports** command. Only in this example does the collection persist to future commands until **iports** is overwritten, unset, or goes out of scope (for more information, see the Scope section).

Controlling Deletion Effort

In cases where a subset of objects in a design is removed, it is not always clear whether to remove the collection. PrimeTime supplies the **collection_deletion_effort** variable to control how much effort is expended to preserve collections. For complete details, see the **collection_deletion_effort** man page.

SEE ALSO

```

add_to_collection(2), compare_collections(2), copy_collection(2),
foreach_in_collection(2), index_collection(2), query_objects(2),
remove_from_collection(2), sizeof_collection(2); filter_collection(2),
sort_collection(2); all_clocks(2), all_connected(2), all_inputs(2),
all_instances(2), all_outputs(2), get_cells(2), get_clocks(2), get_designs(2),
get_generated_clocks(2), get_libs(2), get_lib_cells(2), get_lib_pins(2),
get_nets(2), get_path_groups(2), get_pins(2), get_ports(2), get_qtm_ports(2),
get_timing_paths(2), collection_deletion_effort(3),
collection_result_display_limit(3).

```

commit_fp_group_block_ring

Commits the power ground group block ring and/or straps based on the results from the **create_fp_group_block_ring** command.

SYNTAX

```
status commit_fp_group_block_ring
```

ARGUMENTS

none

DESCRIPTION

Generates a real power ground group block ring and/or straps based on group block ring creation results from the **create_fp_group_block_ring** command. Vias along with wires are also created at each intersection of two wires (For example, a ring or strap) in opposite directions.

EXAMPLES

The following example commits the group block ring.

```
prompt> commit_fp_group_block_ring
```

SEE ALSO

`create_fp_group_block_ring(2)`

commit_fp_plan_groups

Transforms the specified plan groups into soft macros.

SYNTAX

```
status commit_fp_plan_groups  
[-push_down_power_and_ground_straps]  
[-new_top_cell new_cell_name]  
[plan_groups]
```

Data Types

<i>new_cell_name</i>	string
<i>plan_groups</i>	collection

ARGUMENTS

-push_down_power_and_ground_straps

Specifies that the command pushes down global power and ground preroutes into the soft macro and removes them from the top level. This option only pertains to power and ground nets that are outside the scope of the plan group. Any routing associated with a power or ground net that is defined within the plan group will be pushed down automatically along with any other net route objects that are owned by the plan group. The use of this option is equivalent to the combination of running `commit_fp_plan_groups` (without this option) and subsequently running `push_down_fp_objects` and specifying routing as the type and specifying a collection of power and ground nets.

-new_top_cell *new_cell_name*

Specifies that the command creates a new top cell where the top-level changes occur. When this option is used, a new top cell is created and the current cell is changed to the new top cell and plan groups within this new top cell are committed. This is equivalent to internally calling "`current_mw_cell new_cell_name`". When the user calls "`save_mw_cell -hier`" after commit when using this option, then the new top cell is saved and all soft macros created by commit are saved also.

plan_groups

Specifies the plan groups that are committed to soft macros.

If you do not specify this option, all plan groups are committed to soft macros.

DESCRIPTION

This command transforms the specified plan groups into soft macros. For each plan group processed, the tool creates a soft macro to replace the plan group. The soft macro uses the plan group's placement. The soft macro's port instances are connected to their corresponding nets. The plan group's cells are disconnected and deleted, along with the plan group. On the soft macro level, the cells that correspond to the plan group's master-level cells are created in the child, placed relative to their master-level placement, and connected to the corresponding macro-level nets. When

used with the "-new_top_cell" option, the user has the benefit of the changes made to a new top layout cell without modifying the currently open cell. If a UPF power domain is detected which is defined completely within the plan group's hierarchical cell instance, and the power domain lists the hierarchical cell instance as an element of that power domain, all UPF objects related to that power domain will be pushed into the soft macro and removed from the top cell. Any embedded UPF power domain and voltage area whose scope is a child of the plan group's hierarchical cell instance will also be pushed down into the soft macro and removed from the top cell. All routing associated with plangroup-owned nets (nets in the top cell that are internal to the plan groups) is automatically removed from the top cell and instantiated in the soft macro cell, both signal and power/ground (as long as they are owned by the plan group). Global power and ground nets do not get pushed down automatically, as they are not "owned" uniquely by the plan group. Such global power and ground nets must be pushed down after commit using `push_down_fp_objects`.

Hard move bounds and relative placement groups which overlap the plan group or which logically are members of the plan group are pushed down to the soft macro and removed from the top cell.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example commits the specified plan groups to soft macros.

```
prompt> commit_fp_plan_groups $PG_List
```

The following example commits all plan groups to soft macros and makes the changes in a new cell named TOP_NEW.

```
prompt> commit_fp_plan_groups -new_top_cell TOP_NEW
```

The following example commits all plan groups in the current design to soft macros and pushes down the global power and ground net straps.

```
prompt> commit_fp_plan_groups -push_down_power_and_ground_straps
```

SEE ALSO

`uncommit_fp_soft_macros(2)`
`push_down_fp_objects(2)`

commit_fp_rail

Commits the power network (power/ground wires and vias) based on power network synthesis (PNS) results.

SYNTAX

```
int commit_fp_rail
```

ARGUMENTS

none

DESCRIPTION

Generates a real power network (power/ground wires and vias) based on PNS results, when the IR (voltage) drop map meets target IR drop constraints. Power/ground pins might also be created on the chip boundary.

Multicorner-Multimode Support

This command is not supported in a multi-scenario design flow.

EXAMPLES

The following example commits a real power network after PNS is done.

```
prompt> commit_fp_rail
```

SEE ALSO

```
synthesize_fp_rail(2)
```

commit_skew_group

Checks the skew groups defined in the design for common problems that can adversely impact clock tree synthesis. If no problem is found, the command modifies clock tree structure to isolate skew groups.

SYNTAX

```
status commit_skew_group  
[-check_only]
```

ARGUMENTS

-check_only
Causes the command only to check skew groups for common problems but not modifies the clock tree structure to isolate skew groups.

DESCRIPTION

Use the **commit_skew_group** command after defining skew groups for a linked design to check for common problems that adversely impact clock tree synthesis. The following classes of checks are performed:

CTS-924 pins in a skew group from different master clocks CTS-925 pins in a skew group with upstream/downstream relation CTS-926 looping dependencies between skew groups CTS-927 more than one level of dependency between skew groups CTS-928 pins in a skew group with no clocks reaching them CTS-929 skew group spreading across more than one sub-tree CTS-930 incompatible options CTS-931 pins beyond exceptions

When **commit_skew_group** encounters a problem, the tool displays a warning or information message, which is sometimes followed by additional information to help solve the problem. Refer to the man page for each message that explains the problem in detail and shows how to solve it. When no problem is found, the command modifies clock tree structure to isolate skew groups.

EXAMPLES

In the following example, the **commit_skew_group** command checks all skew groups in the currently linked design and isolates skew groups if no problem is found:

```
prompt> commit_skew_group
```

In the following example, the **commit_skew_group** command checks all skew groups in the currently linked design. Clock tree structure is not modified to isolate skew groups:

```
prompt> commit_skew_group -check_only
```


SEE ALSO

`set_skew_group(2)`
`remove_skew_group(2)`
`report_skew_group(2)`
`compile_clock_tree(2)`

compare_collections

Compares the contents of two collections. If the same objects are in both collections, the result is 0 (zero), like string compare. If they are different, the result is nonzero. The order of the objects can optionally be considered.

SYNTAX

```
int compare_collections  
[-order_dependent]  
collection1  
collection2
```

Data Types

```
collection1      collection  
collection2      collection
```

ARGUMENTS

-order_dependent
Specifies that the order of the objects is to be considered. The collections are considered to be different if the objects are ordered differently.

collection1
Specifies the base collection for the comparison. The empty string (the empty collection) is a legal value for the *collection1* argument.

collection2
Specifies the collection with which to compare to *collection1*. The empty string (the empty collection) is a legal value for the *collection2* argument.

DESCRIPTION

The **compare_collections** command is used to compare the contents of two collections. By default, the order of the objects does not matter, so that a collection of the *u1* and *u2* cells is the same as a collection of the *u2* and *u1*. By using the **-order_dependent** option, the order of the objects is considered.

Either or both of the collections can be the empty string (the empty collection). If two empty collections are compared, the comparison succeeds (**compare_collections** considers them identical) and the result is 0 (zero).

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows a variety of comparisons. Note that a result of 0 (zero)

from **compare_collections** indicates success. Any other result indicates failure.

```
prompt> compare_collections [get_cells *] [get_cells *]
0
prompt> set c1 [get_cells {u1 u2}]
{"u1", "u2"}
prompt> set c2 [get_cells {u2 u1}]
{"u2", "u1"}
prompt> set c3 [get_cells {u2 u4 u6}]
{"u2", "u4", "u6"}
prompt> compare_collections $c1 $c2
0
prompt> compare_collections $c1 $c2 -order_dependent
-1
prompt> compare_collections $c1 $c3
-1
```

The following example builds on the previous example by showing how empty collections are compared.

```
prompt> set c4 ""
prompt> compare_collections $c1 $c4
-1
prompt> compare_collections $c4 $c4
0
```

SEE ALSO

`collections(2)`

compare_delay_calculation

Compares the Arnoldi-based delays with the Elmore delays in the current design.

SYNTAX

```
integer compare_delay_calculation
[-verbose]
[-ccs]
```

ARGUMENTS

- verbose
Creates histogram-style reports showing the distribution of the delay differences when using the Arnoldi-based and Elmore delay models.
- ccs
Creates histogram-style reports showing the delay differences when using Composite Current Source (CCS) models and nonlinear delay models (NLDM).

DESCRIPTION

The **compare_delay_calculation** command calculates and compares the timing results of the current design using the Arnoldi-based and Elmore delay models. This command reports pin-to-pin delays, max transition, several DRC violations and the worst slack within each path group for both delay models. In the verbose mode this command creates histogram-style reports showing the distribution of the Arnoldi-based and Elmore delay differences on the paths with negative slacks. The distribution of the differences is represented by the two histograms showing the delay differences in the library units and in the percentage.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following is an example of an Arnoldi-based and Elmore delay calculation report:

```
prompt> compare_delay_calculation -verbose
```

Percent of Arnoldi-based delays = 66.67%

	Elmore		Arnoldi	
Max Transition	0.195		0.521	
DRC Violations	1		1	
Group	WNS	TNS	WNS	TNS

CLK1	1.71	3.96	1.46	2.64
default	0.00	0.00	0.00	0.00

Histograms for Delays with Negative Slack

Difference (%)	Count	%
0 - 1	2	8.33
1 - 5	5	20.83
5 - 10	3	12.50
10 - 50	10	41.67
50 - 100	4	16.67
100+	0	0.00

Difference(lib_units)	Count	%
0.0 - 0.001	0	0.00
0.001- 0.01	3	12.50
0.01 - 0.05	7	29.17
0.05 - 0.1	0	0.00
0.1 - 1.0	14	58.33
1.0+	0	0.00

Histograms for Slews with Negative Slack

Difference (%)	Count	%
0 - 1	0	0.00
1 - 5	8	33.33
5 - 10	9	37.50
10 - 50	3	12.50
50 - 100	4	16.67
100+	0	0.00

Difference(lib_units)	Count	%
0.0 - 0.001	0	0.00
0.001- 0.01	12	50.00
0.01 - 0.05	0	0.00
0.05 - 0.1	0	0.00
0.1 - 1.0	12	50.00
1.0+	0	0.00

SEE ALSO

```
read_parasitics(2)
report_delay_calculation(2)
report_timing(2)
```

compare_interface_timing

Compares two **write_interface_timing** reports.

SYNTAX

```
int compare_interface_timing  
ref_timing_file  
cmp_timing_file  
[-output file_name]  
[-absolute_tolerance atol_list]  
[-nosplit]  
[-sort_by_worst]  
[-significant_digit digits]
```

Data Types

<i>ref_timing_file</i>	string
<i>cmp_timing_file</i>	string
<i>file_name</i>	string
<i>atol_list</i>	list
<i>digits</i>	int

ARGUMENTS

ref_timing_file
Specifies the name of the timing file to be used as the reference in the comparison.

cmp_timing_file
Specifies the name of the timing file to be compared to the reference timing file.

-output *file_name*
Specifies the name of the output file to which the results of the comparison are to be written. The information written to *file_name* specifies PASS/FAIL status for every parameter compared by the command. By default, these results are written to the session transcript. All informational, warning, or error messages are still directed to the session transcript even if this option is specified.

-absolute_tolerance *atol_list*
Specifies the absolute error tolerance for time data. See below for a description of the tolerance format. The default is "0.1".

-nosplit
The **-nosplit** option prevents line-splitting. This is most useful for performing diffs on previous scripts or for postprocessing the script.

-sort_by_worst
Specifies that the output is to be sorted from negative to positive, with the worst failure first. After the FAIL section, all PASS lines are sorted alphabetically within each path attribute. By default, the output follows the

order of the reference `write_interface_timing` data, which is sorted alphabetically within each path attribute.

`-significant_digit digits`

Specifies the number of digits to the right of the decimal point that are to be reported following the method used in the **report_timing** command. Allowed values are 0-13. The default value is 2. Use this option if you want to override the default. See below for a more detailed description of significant digits.

DESCRIPTION

The **compare_interface_timing** command compares two interface timing files (called the "reference" and "comparison" files) previously generated by the **write_interface_timing** command. The result is PASS if the timing parameter values in the two files are the same or within a specified tolerance. The result is FAIL if both columns have data and the tolerance is exceeded. If all comparisons in the report are PASS, the return code for the **compare_interface_timing** command is 0. If one or more comparisons result in FAIL, the return code is 1. If either file format does not conform to the **write_interface_timing** standard, the command issues a warning message.

The **compare_interface_timing** command lets you specify the input and output files for the comparison, and the allowed tolerance levels that trigger comparison failures.

If the reference and comparison files were generated using different contexts then the two files may contain different timing arcs along with different slacks. If an arc exists in the comparison file but not in the reference file then **compare_interface_timing** ignores the extra arcs. If an arc exists in the reference file but not in the comparison file, then **compare_interface_timing** reports the slack value for the reference file, but either "N.C." or "----" for the comparison file slack. The slack difference is marked as "----", and the status is FAIL.

The slack for a missing arc is mark as "N.C.", meaning not critical, if it exists in the reference but not in the comparison file. Each of these groups can have several clocks in them, and under certain circumstances **write_interface_timing** will see the path to one clock as critical for the reference view and report that arc and slack, but another clock as critical in the comparison view and report that arc and slack.

The **compare_interface_timing** command deals with significant digits in two distinct ways. First, the internal computations are limited to the minimum precision of the two input files. This means that if the *ref_timing_file* was generated with three digits and the *cmp_timing_file* with four, then **compare_interface_timing** will use three digits for all internal computations.

It is important to make the tolerances larger than a unit of the computational significant digits, or **compare_interface_timing** could generate false FAIL and PASS messages. This can happen because of rounding. For example, if the internal significant digits is two, and the absolute tolerance is set to 0.01, then a difference of 0.014 is rounded to 0.01 and reported as a PASS even though it is outside the tolerance. Similarly, if the absolute tolerance of 0.009 a difference of 0.009 is rounded to 0.01 and reported as FAIL even though it is within tolerance. The user should ensure that the number of digits of accuracy of the input files is greater than the accuracy of the tolerances.

A second use of significant digits is in report generation, which is limited to the minimum of the input precision and the significant digits specified by the user. The user specifies the significant digits for the report through the **-significant_digits** option. For example, if the value for the **-significant_digits** option is five but the *ref_timing_file* was generated with four digits, then the report will be limited to four digits. Note that this use of significant digits does not affect the PASS or FAIL status, but a reported difference could look like it should have a different status if the difference is close to a tolerance and the difference rounded when writing the report.

Absolute tolerance is a list of either one or two floating point numbers and the sign of the values has no effect. If there is only one number it serves to specify both the plus and minus tolerances. The plus tolerance is the absolute value of the number, and the minus tolerance is the inverse of the plus. For example, *atol_list* equal to **0.1** indicates that time differences of less than 0.1 and greater than -0.1 are considered to be passing.

If there are two numbers then the first specifies the minus bound and the second the plus. The minus tolerance is the inverse of the absolute value of the first number, and the plus tolerance is the absolute value of the second number. Having a pair of numbers allows different tolerances to be applied for negative and positive errors, which can be interpreted as optimism and pessimism depending on the arc type. For example, an *atol_list* equal to **{0.2 -0.3}** indicates that data differences of less than 0.3 and greater than -0.2 are considered passing.

EXAMPLES

The following example shows a comparison:

```
prompt> compare_interface_timing demo_net.rpt demo_etm.rpt
? -absolute_tolerance {0.01 0.02}
```

```
*****
```

```
Design      : test
Scenario    : --
Version     : C-2009.06-ICC
Date        : Thu Jan 22 03:01:40 2009
```

```
*****
```

From us	To	Type	Worst Slack		Difference	Stat
			Ref	Model		

-						
c	CLK	max_rise	3.709	3.709	0.000	PASS
c	CLK	max_fall	3.614	3.614	0.000	PASS
c	CLK	min_rise	2.127	2.127	0.000	PASS
c	CLK	min_fall	2.056	2.056	0.000	PASS
CLK	o2	max_rise	4.152	4.131	0.021	FAIL
CLK	o2	max_fall	4.205	4.205	0.000	PASS
CLK	o2	min_rise	1.550	1.550	0.000	PASS
CLK	o2	min_fall	1.612	1.623	-0.011	FAIL

SEE ALSO

`write_interface_timing(2)`

compare_lib

Performs a cross-reference check between a technology library and a symbol library or between a technology library and a physical library.

SYNTAX

```
int compare_lib  
library1  
library2
```

Data Types

```
library1    string  
library2    string
```

ARGUMENTS

library1
Specifies the name of a technology library.

library2
Specifies the name of a symbol library or physical library to be compared with the technology library.

DESCRIPTION

This command compares the given technology and symbol (physical) libraries for consistency. For this command to execute successfully, the libraries to be compared must be in the Synopsys internal database format and must also exist in `dc_shell` or `lc_shell`. To compile a library, use the **read_lib** command. To load a compiled library, use the **read** command.

The **compare_lib** command performs the following checks:

- First, it ensures that each component in the technology library has a corresponding symbol definition in the symbol(physical) library.
- Second, it crosschecks the pin names of each component in the technology library against the pin names defined for its corresponding symbol.
- Third, it ensures that each symbol in the symbol (physical) library has a corresponding cell definition in the technology library.
- Fourth, it crosschecks the pin names of each symbol in the symbol (physical) library against the pin names defined for its corresponding technology cell.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example compares two specific libraries; a technology library and a symbol library.

```
prompt> compare_lib tech_lib.db sym_lib.sdb
```

SEE ALSO

```
read_lib(2)  
report_lib(2)  
write_lib(2)
```

compare_rc

Compares annotated capacitance with estimation capacitance.

SYNTAX

```
int compare_rc  
[-bound max_float_capacitance]  
[-min]  
[-net net_list]  
[-threshold min_float_capacitance]  
[-worst_nets integer_nets]
```

Data Types

<i>max_float_capacitance</i>	float
<i>net_list</i>	list
<i>min_float_capacitance</i>	float
<i>integer_nets</i>	integer

ARGUMENTS

-bound *max_float_capacitance*
Specifies the maximum back-annotated capacitance below which nets are to be considered for the report.

-min
Reports the minimum conditions. By default, the command reports maximum conditions.

-net *net_list*
Compares only those nets in *net_list*. If you also use either the **-threshold** or the **-bound** option, only those nets that meet the criteria in the current instance are used. When you use the **-net** option, the command ignores the **-worst_nets** option.

-threshold *min_float_capacitance*
Specifies the minimum back-annotated capacitance above which nets are to be considered for the report.

-worst_nets *integer_nets*
Specifies the number of worst nets to report based on the difference between the estimated capacitance and the back-annotated capacitance. If a negative number is specified or the number is higher than 500, all nets are reported without ordering. The default is 10.

DESCRIPTION

This command compares the back-annotated capacitance on the nets in the design with the estimation capacitance and plots the percentage deviation of estimation capacitance from the back-annotated capacitance versus the percentage of nets in the deviation range. The worst deviation that is reported is 200 percent. Any deviation

above that is rounded to 200 percent. This command also reports the worst nets in terms of percentage deviation of estimated capacitance from the back-annotated capacitance.

Before running this command, you must source `set_load` files or read in a post route data base with capacitance on the nets in the design.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example runs the **compare_rc** command.

```
prompt> compare_rc -threshold 0.0001 -worst_nets 25
Loading target library 'ssc_core'
Loading design 'address'
Information: The distance unit in Capacitance and Resistance is 1 micron. (GR-7)
Information: Library Derived Horizontal Cap : 0.0001 (GR-10)
Information: Library Derived Vertical Cap : 0.00011 (GR-10)
Information: Library Derived Horizontal Res : 0.00016 (GR-10)
Information: Library Derived Vertical Res : 0.00014 (GR-10)
Information: Using region-based R and C coefficients. (GR-13)
```

Net	Estimated Cap	Back-annotated Cap	%age-diff	fanout
n3331	0.00123256	0.00049	151.543	2
n3325	0.000762279	0.00031	145.896	2
N87AC	0.000726195	0.00031	134.256	2
N37AH	0.000726195	0.00031	134.256	2
n5490	0.000822903	0.00036	128.584	2
n4532	0.000701367	0.00031	126.247	2
n4187	0.00105752	0.00049	115.82	2
n6545	0.000873081	0.00041	112.946	2
N111AI	0.000820671	0.0004	105.168	2
N86AH	0.000812273	0.0004	103.068	2
N63AE	0.000812273	0.0004	103.068	2
n2000	0.000814395	0.0004	103.599	2
N125A	0.000812273	0.0004	103.068	2
N2AD	0.000812273	0.0004	103.068	2
n6933	0.000983985	0.00049	100.813	2
n1849	0.00115294	0.00058	98.7828	2
N49AF	0.000781043	0.0004	95.2607	2
n6051	0.000797972	0.00041	94.6273	2
n6718	0.000957085	0.0005	91.4169	2
n3222	0.000949579	0.0005	89.9158	2
n3119	0.000775716	0.00041	89.1991	2
n4704	0.000777002	0.00041	89.5128	2
n3628	0.000777554	0.00041	89.6473	2
n2597	0.000775716	0.00041	89.1991	2

n3191	0.000777554	0.00041	89.6473	2

100

+

90

+

80

+

70

+

*

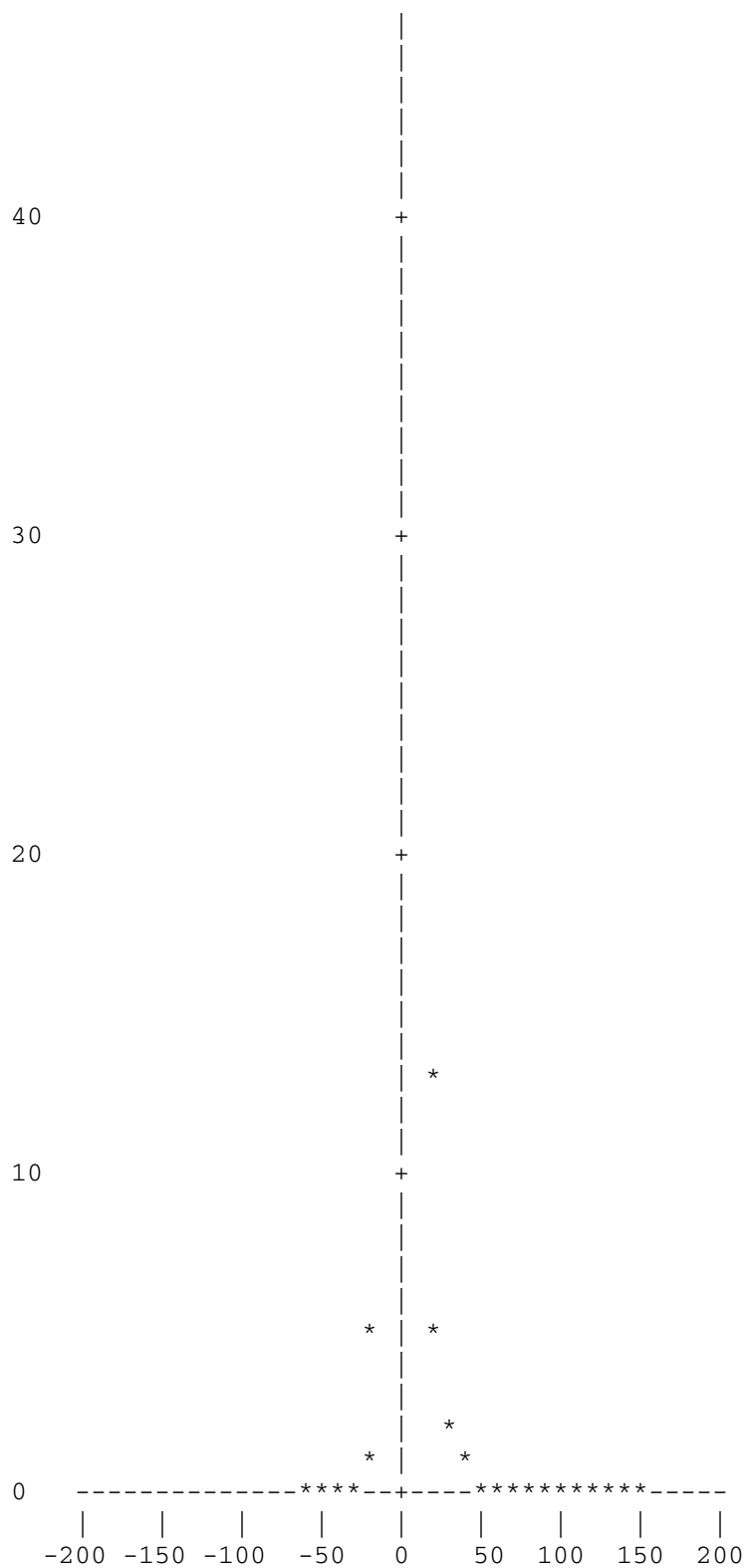
60

+

50

+

compare_rc
300



SEE ALSO

`set_load(2)`

compile_clock_tree

Builds a clock tree based on the clock tree definition.

SYNTAX

```
status compile_clock_tree
[-clock_trees name_or_source_pin_list]
[-config_file_read read_filename]
[-config_file_write write_filename]
[-operating_condition min | max | min_max]
[-high_fanout_net net_or_pin_list]
[-sync_phase rise | fall | both]]
```

Data Types

<i>name_or_source_pin_list</i>	list
<i>net_or_pin_list</i>	list

ARGUMENTS

- clock_trees *name_or_source_pin_list***
Compiles the clock trees whose names or root pins are listed in the *name_or_source_pin_list* argument. Each element of the *name_or_source_pin_list* can be either a user-assigned symbolic name or the name of the source (port or pin) of each clock tree. It must match the full name of the clock tree root pin or port passed to the **set_clock_tree_options** command vy using the **-root** option. By default, the command is applied to all currently-defined clock trees.
- config_file_read *read_filename***
Instructs the clock tree synthesis engine to read the clock tree configuration specified in *read_filename* file. To see the format of the clock tree configuration, build the clock tree and then write out the configuration file. You can edit the file to obtain the desired configuration. If you are already familiar with the format, then clock tree synthesis can begin with a specified configuration file. The configuration can be soft, meaning that only the number of levels is specified, and the clock tree synthesis engine will honor the specification. The number of buffers at each level are determined by clock tree synthesis. The configuration can also be hard, meaning that the number of levels and number of buffers at each level is specified. The buffer type can also be specified. The clock tree synthesis engine will honor all of the specifications.
- config_file_write *write_filename***
Instructs the clock tree synthesis engine to write out the clock tree configuration levels and number of buffers at each level after clock tree synthesis has been completed.
- operating_condition min | max | min_max**
Instructs the clock tree synthesis engine to use the specified operating condition. This option inserts a balanced buffer tree for signal high fanout nets using the same techniques as in clock tree synthesis. You must specify

the nets or nets' driving pin in the *net_or_pin_list*. A common use of this option is for setting or resetting nets. By default, it uses **max**.

-high_fanout_net *net_or_pin_list*
Specifies a list of high fanout nets.

-sync_phase rise | fall | both
Instructs the HFCTS to balance skew at rise, fall, or both edges. Use this option only when you also use the **-high_fanout_net** option. By default, the tool uses **rise** in skew calculation.

DESCRIPTION

This command synthesizes a clock tree and updates the IC Compiler database with the results of the compiled clock trees.

Clock trees to be synthesized must be previously defined by using the **set_clock_tree_options** command before using the **compile_clock_tree** command. Before running the **compile_clock_tree** command, you must also run the **set_clock_tree_references** command to define which clock tree buffers or inverters can be used for those clocks during clock tree synthesis.

By default, design rule checking (DRC) violations beyond exception pins are fixed at the end of clock tree synthesis.

Multicorner-Multimode Support

This command uses information from the clock tree synthesis scenario.

EXAMPLES

The following example shows a clock tree with root *CLK1* implemented with a set of specified library references to be used as buffers. The *CLK1* clock is compiled and updated in the design netlist.

```
prompt> set_clock_tree_options -root [get_ports CLK1]
prompt> set_clock_tree_references -reference {buffer1x buffer2x}
```

The following example shows a command performing clock tree synthesis for the *CLK1* clock.

```
prompt> compile_clock_tree -clock_trees CLK1
```

The following example shows a command reporting the structural and timing characteristics of the compiled clock tree.

```
prompt> report_clock_tree -clock_trees CLK1
```

SEE ALSO

`report_clock_tree(2)`
`set_clock_tree_options(2)`
`set_clock_tree_references(2)`

compile_fp_clock_plan

Compiles clock trees inside a plan group and builds clock trees at the top level according to the plan group's result.

SYNTAX

```
status compile_fp_clock_plan  
[-operation_cond min | max(default) | min_max]  
[-anchor_only]  
[-parallel]
```

ARGUMENTS

-operation_cond [min | max(default) | min_max]
Instructs the tool to use the specified operating condition.

-anchor_only
Specifies that the tool should only insert anchor cells on the plan group input ports, but should not synthesize the clock plan.
By default, the tool inserts anchor cells on the plan group input ports, then synthesizes the clock plan.

-parallel
Specifies that the tool generates clock trees inside plan groups using multiple processes in parallel to speed-up run time of clock planning.

DESCRIPTION

The **compile_fp_clock_plan** command performs the following tasks:

1. Inserts anchor cells on the plan group input ports.
2. Generates the clock trees inside each plan group.
3. Defines the input pin of each anchor cell to be a float pin.
4. Generates the top-level clock tree.
5. Performs detail routing on the clock interface nets.

The log-files and reports are generated in clock-planning output directory.

The plan groups to be synthesized must be defined with the **set_fp_clock_plan_options** command before using the **compile_fp_clock_plan** command. In addition, you must set the following options by using the **set_clock_tree_options** command:

-parallel options is most efficient in speeding up clock planning in large designs meeting the following criterias:

1. Design includes more than 3 plan group
2. Most of the clock sinks are locataed in the plan groups
3. Design has limited clock dependency between plan groups, i.e., only a small number of clock nets come out of a plan group and drive subtrees at top level or in other plan groups.

-parallel options is not compatible with design of fully abutted floorplan. Because parallelized generation of plan group clock tree does not keep the plan group clock trees, **-parallel** options will be ignored when **-keep_block_tree** is set to true by **set_fp_clock_plan_options**, .

Multicorner-Multimode Support

This command is not supported in a multi-scenario design flow.

EXAMPLES

In the following example, the anchor cell CLKBUF12 is inserted in all plan groups, then the clock net *CLK1* is implemented in all plan groups.

```
prompt> set_fp_clock_plan_options -clock_nets CLK1 \  
        -anchor_cell CLKBUF12  
prompt> compile_fp_clock_plan
```

In the following example, the anchor cell CLKBUF12 is inserted in all plan groups, then the clock net *CLK1* is implemented in all plan groups using multiple processes.

```
prompt> set_fp_clock_plan_options -clock_nets CLK1 \  
        -anchor_cell CLKBUF12  
prompt> compile_fp_clock_plan -parallel
```

The following command inserts anchor cells for each plan group, but does not perform clock tree synthesis.

```
prompt> set_fp_clock_plan_options -clock_nets CLK1 \  
        -anchor_cell CLKBUF12  
prompt> compile_fp_clock_plan -anchor_only
```

SEE ALSO

`set_fp_clock_plan_options(2)`

compile_premesh_tree

Invokes clock tree synthesis on a net that is driving a clock mesh.

SYNTAX

```
status compile_premesh_tree  
-clock_tree name_or_root_pin
```

Data Types

name_or_root_pin string

ARGUMENTS

```
-clock_tree name_or_root_pin  
    Specifies the name of the clock tree or the name of the root (a pin, port or  
    net) of the clock mesh. This is a required option.
```

DESCRIPTION

This command invokes the **compile_clock_tree** command to generate a buffer tree with drivers connecting to a clock mesh. Before running **compile_premesh_tree**, you must first create clock mesh wiring grid and add drivers that are connected to the wiring grid. The inputs of the buffer tree that drives the clock mesh wiring grid are typically connected to the clock root. The outputs of the buffer tree that drives the clock mesh wiring grid are connected to the mesh net. These connections must be set up before the command invokes **compile_clock_tree**.

The root net might be associated with one or more clocks, but this command processes the root net as a single electrical signal. Any multiplexing of clocks must be done in the circuitry that drives the driver pin or port of the root net. This command only generates a buffer tree and does not generate any multi-input gates for gated clocks or integrated clock gating (ICG) cells.

See **create_clock_mesh** for more information on clock net, root net, and mesh net.

Multicorner-Multimode Support

This command uses information from the clock tree synthesis scenario.

EXAMPLE

The following command invokes clock tree synthesis on an ideal clock net that was previously connected to the ideal net "clk1".

```
prompt> compile_premesh_tree -clock_tree [get_pins clk1top/Z]  
1
```

SEE ALSO

`create_clock_mesh(2)`
`add_clock_drivers(2)`
`route_mesh_net(2)`
`analyze_subcircuit(2)`
`compile_clock_tree(2)`

compute_polygons

Returns a list of polygons that cover the region computed by performing the specified Boolean operation on the input polygons.

SYNTAX

```
list compute_polygons  
-boolean and | or | not | xor  
poly_list1  
poly_list2
```

Data Types

```
poly_list1    list  
poly_list2    list
```

ARGUMENTS

-boolean and | or | not | xor

Specifies the Boolean operation to perform on the specified polygons.

If the operation type is **and**, the returned polygons cover the areas that are common to the input polygons.

If the operation type is **or**, the returned polygons cover the areas that are covered by any input polygon.

If the operation type is **not**, the returned polygons cover the areas that are covered by the first list of polygons, but not covered by the second list of polygons.

If the operation type is **xor**, the returned polygons cover the areas that are covered by either list of polygons, but not covered by both lists of polygons.

poly_list1

Specifies the first list of polygons. A valid polygon must be rectilinear, so the adjacent points have one same coordinate. You specify a polygon by providing a list of points in the following format: {{x1 y1} {x2 y2} ... {xN yN} {x1 y1}}.

You can specify a list of polygons or a single polygon as the input for this option, see *EXAMPLE* section for more detail.

The coordinate unit is specified in the technology file; typically it is microns.

poly_list2

Specifies the second list of polygons. A valid polygon must be rectilinear, so the adjacent points have one same coordinate. You specify a polygon by providing a list of points in the following format: {{x1 y1} {x2 y2} ... {xN yN} {x1 y1}}.

You can specify a list of polygons or a single polygon as the input for this option, see *EXAMPLE* section for more detail.

The coordinate unit is specified in the technology file; typically it is microns.

DESCRIPTION

This command returns a list of polygons that cover the regions computed by performing the specified Boolean operation on the input polygons. The supported Boolean operations are OR, AND, NOT, and XOR.

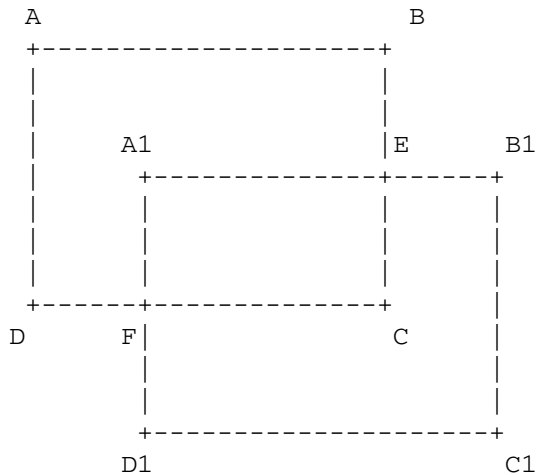
The command returns a list of polygons, where each polygon is represented as a Tcl list.

Note that this command might return a list of polygons that represents a disjoint rectilinear region. You can directly pass the result as a parameter to **compute_polygons** command again, but cannot pass it to other polygon commands: **convert_from_polygon**, **resize_polygon**, **get_polygon_area**. This is because other polygon commands take a single polygon as input. Instead, you must use a Tcl list command, such as the **foreach** or **lindex** command, to extract each polygon from the returned list, and then pass each polygon to the polygon command.

You must open the Milkyway design library before using this command.

EXAMPLES

Assume that you have two polygons, A-B-C-D-A and A1-B1-C1-D1-A1, as shown in the following figure:

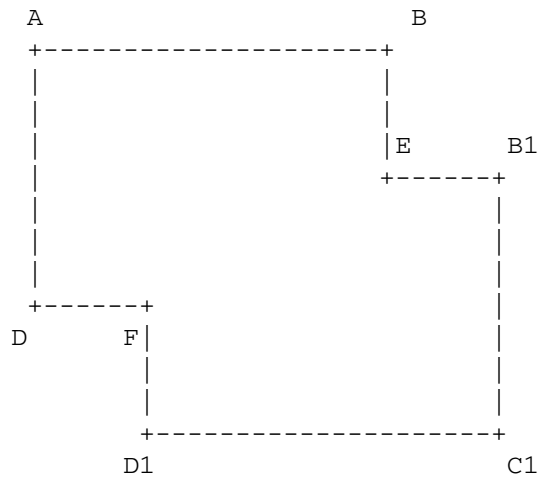


The following example performs the Boolean OR operation on these input polygons. The result is the A-B-E-B1-C1-D1-F-D-A polygon, which covers the region covered by one or both of the input polygons.

```
prompt> compute_polygons -boolean or \  
{0 10} {30 10} {30 30} {0 30} {0 10}} \  
{10 0} {40 0} {40 20} {10 20} {10 0}}  
{10 20} {30 20} {30 10} {10 10} {10 20}}
```

The following figure shows the resulting polygon.

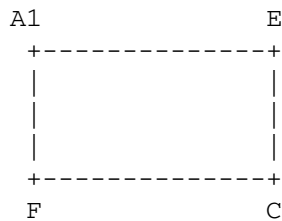
compute_polygons



The following example performs the Boolean AND operation on these input polygons. The result is the A1-E-C-F-A1 polygon, which covers the region covered by both of the input polygons.

```
prompt> compute_polygons -boolean and \
{{0 10} {30 10} {30 30} {0 30} {0 10}} \
{{10 0} {40 0} {40 20} {10 20} {10 0}}
{{10 20} {30 20} {30 10} {10 10} {10 20}}
```

The following figure shows the resulting polygon.

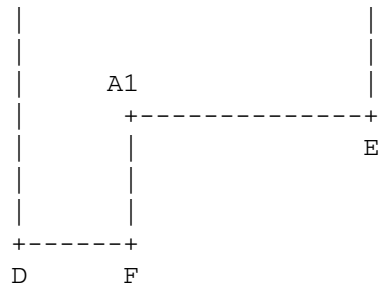


The following example performs the Boolean NOT operation on these input polygons. The result is the A-B-E-A1-F-D-A polygon, which covers the regions covered by the first polygon, A-B-C-D-A, but not by the second polygon, A1-B1-C1-D1-A1.

```
prompt> compute_polygons -boolean not \
{{0 10} {30 10} {30 30} {0 30} {0 10}} \
{{10 0} {40 0} {40 20} {10 20} {10 0}}
{{30 30} {30 20} {10 20} {10 10} {0 10} {0 30} {30 30}}
```

The following figure shows the resulting polygon.

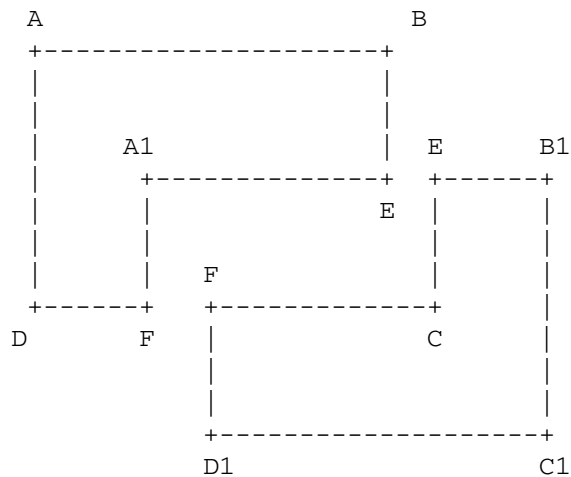




The following example performs the Boolean XOR operation on these input polygons. The result is two polygons, A-B-E-A1-F-D-A and E-B1-C1-D1-F-C-E, which cover the areas covered by either input polygon, but not by both.

```
prompt> compute_polygons -boolean xor \
{{0 10} {30 10} {30 30} {0 30} {0 10}} \
{{10 0} {40 0} {40 20} {10 20} {10 0}}
{{40 20} {40 0} {10 0} {10 10} {30 10} {30 20} {40 20}}
{{30 30} {30 20} {10 20} {10 10} {0 10} {0 30} {30 30}}
```

The following figure shows the resulting polygons (the polygons have been moved slightly to clearly display the disjoint polygons).



The following example shows how to use the output of the **compute_polygons** command as input for other polygon commands.

```
set poly1 [convert_to_polygon [get_net_shapes RECTANGLE#123]]
set poly2 [convert_to_polygon [get_net_shapes RECTANGLE#456]]
set my_polys [compute_polygons -boolean xor $poly1 $poly2]

foreach poly $my_polys {
    set my_xor_rect [convert_from_polygon $poly]
    echo $my_xor_rect
}
```

compute_polygons

```
}
```

The following example shows that **compute_polygons** can accept list of polygons as input.

```
prompt> compute_polygons -boolean or {{{0 0} {10 0} {10 10} \  
        {0 10} {0 0}} {{20 0} {30 0} {30 10} {20 10} {20 0}}} \
```

 {{{10 0} {20 0} {20 10} {10 10} {10 0}}}

```
        {{{0.000 10.000} {30.000 10.000} {30.000 0.000} {0.000 0.000} {0.000 10.000}}}
```

SEE ALSO

```
convert_to_polygon(2)  
convert_from_polygon(2)  
resize_polygon(2)
```

connect_net

Connects the specified net to the specified pins or ports.

SYNTAX

```
status connect_net  
net object_list
```

Data Types

```
net          string  
object_list  list
```

ARGUMENTS

net

Specifies the net to connect. The net must be a scalar (single bit) net, and must exist in the current design.

object_list

Specifies a list of pins and ports to which the net is to be connected. Pins and ports must be at the same hierarchical level as the specified net, and must exist in the current design. If a specified pin or port is already connected, the tool issues an error message.

DESCRIPTION

This command connects a net to the specified pins or ports at the same hierarchical level. The net can be at any level of hierarchy but the pins or ports should be at the same level. A net can be connected to many pins or ports; however, you cannot connect a pin or port to more than one net.

You can use `connect_net` to connect PG and Tie nets to pins or ports. Only PG nets can be connected to leaf level PG pins. PG nets can not be connected to leaf level signal pins. Tie nets can not be connected to leaf level signal output pins. Tie or PG nets can not be connected to a hierarchical pin which is already connected to a Tie or PG net on the other hierarchical side.

To disconnect objects on a net, use the **disconnect_net** command.

To display pins and ports on a net, use the one of **all_connected** or **get_nets -of \$net** commands.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses **connect_net** to connect net NET0 to ports A1 and A2 and pin U1/A. The **all_connected** command returns the objects connected to net NET0.

```
prompt> connect_net NET0 [get_ports {A1 A2}]
prompt> connect_net NET0 [get_pins U1/A]
prompt> all_connected NET0
{A1 A2 U1/A}
```

The following example shows the error message generated when you attempt to connect a pin or port to more than one net:

```
prompt> connect_net MY_NET_1 PORT1
Connecting net 'MY_NET_1' to port 'PORT1'.

prompt> connect_net MY_NET_2 PORT1
Error: Object 'PORT1' is already connected to net 'MY_NET_1'.
```

The following example shows how **connect_net** is used to connect net U1/MY_NET to the U1/MY_PORT port.

```
prompt> connect_net U1/MY_NET U1/MY_PORT
```

SEE ALSO

```
all_connected(2)
create_net(2)
current_design(2)
disconnect_net(2)
remove_net(2)
```

connect_pin

Connects pins or ports at any level of hierarchy.

SYNTAX

```
int connect_pin  
-from from_object  
-to to_list  
-port_name port_name  
-verbose
```

Data Types

<i>from_object</i>	list
<i>to_list</i>	list
<i>port_name</i>	string

ARGUMENTS

-from *from_object*
Specifies the pin or port from which to connect. The direction of the pin or port cannot be inout.

-to *to_list*
Specifies a list of pins and ports to which the connection must be made. Pins and ports can be at any level of hierarchy. The direction of the pins or ports cannot be inout.

-port_name *port_name*
Specifies the base name to use as names of ports when a new port is created on subdesigns to make the connection.

-verbose
Specifies that the tool displays individual netlist operations while making the global connections.

DESCRIPTION

This command performs global connections between the source object specified in the *from_object* argument and the objects specified in the *to_list*. The pins and ports specified in the argument can be at any level of hierarchy. The parent design of the pins and ports must be unique.

Leaf level PG pins or hierarchical pins connected to PG or Tie nets on one side of the hierarchy can not be connected using connect_pin command.

While making the global connections, ports and nets are created in the subdesigns, if needed. Ports in the subdesigns are reused regardless of their names, when the from pin is already connected to a net. Also, a port in a subdesign is reused if it is unconnected and the name is as specified by the **-port_name** option.

Wherever applicable, the name of the net that is created is the name of the connecting port, as long as there is no net with the same name in that design. If there is an existing net with the name, the name of the net is generated.

The **connect_pin** command ensures that multiply-driven nets are not created. The command does not allow a connection from an output pin to an output pin.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses **connect_pin** to connect *U1/Z* and *mid1/bot1/U3/A*.

Use the **report_net -connections** or **report_cell -connections** to view the connections.

```
prompt> connect_pin [get_pins U1/Z] [find pin mid1/bot1/U3/A]
```

```
prompt> report_net -connections [get_net -of [get_pins U1/Z]]
```

SEE ALSO

`all_connected(2)`
`connect_net(2)`
`create_port(2)`
`report_cell(2)`
`report_net(2)`

connect_power_domain

Connects power net information for the specified power domains. This command is supported only in non-UPF mode.

SYNTAX

```
status connect_power_domain  
power_domains  
[-primary_power_net power_net]  
[-primary_ground_net ground_net]  
[-backup_power_net power_net]  
[-backup_ground_net ground_net]  
[-internal_power_net internal_power_net]  
[-internal_ground_net internal_ground_net]
```

Data Types

<i>power_domains</i>	collection
<i>power_net</i>	string
<i>ground_net</i>	string
<i>internal_power_net</i>	string
<i>internal_ground_net</i>	string

ARGUMENTS

power_domains
Specifies the power domains for which to make the power net information connections.

-primary_power_net power_net
Specifies the primary power net information for the power domains.

-primary_ground_net ground_net
Specifies the primary ground net information for the power domains.

-backup_power_net power_net
Specifies the backup power net information for the power domains.

-backup_ground_net ground_net
Specifies the backup ground net information for the power domains.

-internal_power_net internal_power_net
Specifies the internal power net information for the power domains.

-internal_ground_net internal_ground_net
Specifies the internal ground net information for the power domains.

DESCRIPTION

The **connect_power_domain** command creates logical power connections for the specified power domains. All cells in the power domain inherit the power connections. You can

override the inherited power connections for a cell by using the **connect_power_net_info** command.

The primary power and ground nets are used to define the main power connections for the power domain.

The backup power and ground nets are used to define power connections for always-on logic, retention registers, isolation cells, and enable-level shifter cells.

The internal power and ground nets are used to connect power nets to the switching cells present inside the power domain.

EXAMPLES

The following example connects power net information to the TOP_DOMAIN and SUB_DOMAIN power domains:

```
prompt> connect_power_domain TOP_DOMAIN \  
        -primary_power_net T_VDD \  
        -primary_ground_net T_VSS
```

1

```
prompt> connect_power_domain SUB_DOMAIN \  
        -primary_power_net A_VDD \  
        -primary_ground_net A_VSS \  
        -backup_power_net VDD_Backup \  
        -backup_ground_net VSS_Backup
```

1

SEE ALSO

```
connect_power_net_info(2)  
create_power_domain(2)  
disconnect_power_net_info(2)  
get_power_domains(2)  
remove_power_domain(2)  
report_power_domain(2)
```

connect_power_net_info

Connects the specified power net information to the specified power pins. This command is supported only in non-UPF mode.

SYNTAX

```
status connect_power_net_info  
object_list  
-power_pin_name power_pin_name  
-power_net_name power_net_name
```

Data Types

<i>object_list</i>	list
<i>power_pin_name</i>	string
<i>power_net_name</i>	string

ARGUMENTS

```
object_list  
    Specifies the leaf cells for which the power net information connections are  
    made.  
  
-power_pin_name power_pin_name  
    Specifies the name of the power pin.  
  
-power_net_name power_net_name  
    Specifies the name of the power net.
```

DESCRIPTION

The **connect_power_net_info** command makes power net information connections for a specific power pin of a leaf cell. The pin-level connection overrides the domain-level connections made with the **connect_power_domain** command.

See the **report_power_pin_info** command man page for more information.

EXAMPLES

The following example connects the power net information for the PWR pin of the PD0_INST/I0 cell:

```
prompt> connect_power_net_info PD0_INST/I0 \  
        -power_pin_name PWR -power_net_name exp_VDD  
1
```

SEE ALSO

connect_power_domain(2)

connect_power_net_info
320

```
disconnect_power_net_info(2)  
report_power_pin_info(2)
```

connect_power_switch

Connects switch pins of the power switches in the design.

SYNTAX

```
int connect_power_switch  
-source object  
-port_name port_name  
-mode hfn | daisy  
[-ack_out object]  
[-ack_port_name port_name]  
[-direction horizontal | vertical]  
[-verbose]  
[-voltage_area list]  
[-object_list objects]  
[-lib_pin library_pin_names]
```

Data Types

<i>object</i>	collection
<i>port_name</i>	string

ARGUMENTS

-source *object*
Specifies the pin or port from which to connect. The direction of the pin or port cannot be inout.

-port_name *port_name*
Specifies the base name to use as names of switch ports when a new port is created on subdesigns to make the connection. It is required option.

-mode hfn | daisy
Specifies whether the switch connections are to be made in daisy chain of high fanout style. It is a required option.

-ack_out *object*
Specifies the pin or port that is to be used as acknowledge-out signal. It is optional and is only allowed in *-mode daisy*.

-ack_port_name *port_name*
Specifies the base name to use as names of acknowledge ports when a new port is created. This is required with *-ack_out* option in *-mode daisy*.

-direction horizontal | vertical
Specifies the direction for the daisy mode connections. Requires *-mode daisy*. It is used only when tool tries to infer the daisy chained connections based on the instantiated switch cells. The option can only be used with *-mode daisy* and is mutually exclusive to *-object_lists*.

-verbose
Specifies that the tool displays individual netlist operations while making

the global connections.

-voltage_area *list*

Specifies the voltage areas. The patterns can be a collection handle of voltage areas or name patterns. The command will try to collect the switches in the specified voltage area and connect them. If specified with *-object_list*", the command will ignore switches in the *object_list* which are not in specified voltage areas. Switches will be connected in the order the voltage areas are specified using this option.

-object_list *objects*

Specifies the ordered list of cells on which to perform the connections. In *-mode daisy*, the connections will be made in the order in which they are specified, *-direction* cannot be used with this option.

-lib_pin *library_pin_names*

Specifies the list of library input and output pins that should be used in both *-mode daisy* and *-mode hfn*. This is required option for the switch cells that have multiple switch and acknowledge pins. Command will error out if a switch cell with multiple input/output is used but does not have its lib pins specified in this option. This can be used with both *-mode daisy* and *-mode hfn*.

DESCRIPTION

The *connect_power_switch* command performs the connections for the switch cells instantiated and placed in the design based on user's requirements. The command can identify the switch cells in the design based on liberty syntax, and can connect them in either high-fanout style or daisy-chain style.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLE

The following example connects the switches in mode daisy.

```
prompt> connect_power_switch \
    -source enable \
    -port_name enable_va_0 \
    -mode daisy \
    -direction horizontal \
    -voltage_area voltage_area_2
    -verbose
```

1

SEE ALSO

`create_power_domain(2)`
`create_voltage_area(2)`

connect_spare_diode

Uses the spare diode ports found in the design to fix antenna violations.

SYNTAX

```
status connect_spare_diode
[-exclude_nets collection_of_nets]
[-antenna_check_engine internal | hercules]
[-internal_check_option all | top_layer_only]
[-routing skip | route]
[-distance distance_number]
[-
signal_route_options ignore_lower_layers | include_lower_layers | include_all_lower
_layers | advanced]
[-max_ratio max_ratio_number]
```

Data Types

<i>collection_of_nets</i>	collection
<i>distance_number</i>	integer
<i>max_ratio_number</i>	integer

ARGUMENTS

- exclude_nets *collection_of_nets*
Specifies a collection of nets that should not be connected to diodes. If you do not specify this option, all nets will be connected to diodes.
- antenna_check_engine internal | hercules
Specifies either the internal antenna checker or Hercules. The default is **internal**.
- internal_check_option all | top_layer_only
Specifies to report either all antenna rule violations or only the top-layer antenna violations. The default is **all**. This option applies only when -antenna_check_engine internal is specified.
- routing skip | route
Specifies the extent of the routing process. Using **skip** stops routing. Using **route** invokes the detail router in ECO mode to route diode ports. The default is **route**.
- distance *distance_number*
Specifies the maximum distance to search for spare diodes. The default value is 10 times the global route cell size.
- signal_route_options ignore_lower_layers | include_lower_layers | include_all_lower_layers | advanced
Specifies the HPO signal route options for checking a charge-collecting antenna. The default is the value of the doAntennaConx detail routing option.

`-max_ratio max_ratio_number`

Specifies the maximum allowable ratio of wiring area and gate area for checking a charge-collecting antenna.

DESCRIPTION

This command uses spare diodes in the design to fix antenna violations. First, the tool runs the internal antenna checker or uses Hercules output to identify antenna violations. For each antenna violation, this command searches the neighborhood of the antenna for spare diodes. The number of required diode ports is determined by **define_antenna_rule** and **define_antenna_layer_rule**.

When enough diode ports are found to fix an antenna violation, it sets the net ID of all the selected diode ports to be the same as the net ID of the antenna to establish the net connectivity. Finally, you might choose to complete the routing of the connected diode ports.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following is an example of the **connect_spare_diode** command.

```
prompt> connect_spare_diode -antenna_check_engine internal\  
-internal_check_option top_layer_only\  
-signal_route_options include_all_lower_layer\  
-max_ratio 1000
```

The following is an example of the **connect_spare_diode** command.

```
prompt> connect_spare_diode -route skip -distance 10
```

SEE ALSO

```
insert_diode(2)  
define_antenna_rule(2)  
define_antenna_layer_rule(2)
```

connect_supply_net

Connects the supply net to the specified supply ports and pins. This command is supported only in UPF mode.

SYNTAX

```
status connect_supply_net  
supply_net_name  
-ports list
```

Data Types

```
supply_net_name    string  
list               list
```

ARGUMENTS

```
supply_net_name  
    Specify the name of the supply_net to be connected.  
    If a supply net with the specified name, does not exist in the current scope,  
    the supply_net cannot be connected.  
    This is a required option and must be specified.  
  
-ports list  
    Specify which supply ports or pins are to be connected with the supply_net.  
    The name is hierarchical name.
```

DESCRIPTION

The **connect_supply_net** command provides an explicit connect of a supply_net to any supply_ports/pins and overrides (has higher precedence than) the auto-connection semantics that might otherwise apply. If a design element isn't connected explicitly to any supply_net using connect_supply_net, it will share the primary power/ground supply_net with the power_domain it belongs to.

The supply_net must be created on the same power_domain as the supply_port before they could be connected. The instance which contains the pin must also be in the extend of the same power_domain. If the specified supply_net/supply_port/pin are not in the extend of same power_domain, this command fails.

Command returns 1 on success, returns 0 otherwise.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example connects a supply_net VSS with support_port VSS:

```
connect_supply_net  
326
```



```
prompt> create_power_domain PD1 -elements INST_1 PD1 prompt> create_supply_net VSS -  
domain PD1 VSS prompt> create_supply_port VSS -domain PD1 VSS prompt>  
connect_supply_net VSS -ports VSS 1
```

SEE ALSO

```
create_supply_net(2)  
report_supply_net(2)
```

connect_tie_cells

Instantiates tie-high and tie-low cells and connects them to the specified cell ports.

SYNTAX

```
status connect_tie_cells
-objects {object_coll}
-obj_type port_inst | cell_inst | lib_cell
[-tie_high_lib_cell lib_cell]
[-tie_low_lib_cell lib_cell]
[-tie_highlow_lib_cell lib_cell]
[-tie_high_port_name port]
[-tie_low_port_name port]
[-max_fanout number]
[-max_wirelength number]
[-incremental true | false]
```

Data Types

<i>object_coll</i>	collection
<i>lib_cell</i>	string
<i>port</i>	string
<i>number</i>	integer

ARGUMENTS

-objects {*object_coll*}

Specifies the objects from which the tool infers the ports to which the tie-high and tie-low cells are connected. The objects can be cell instances, port instances, or reference cells.

-obj_type port_inst | cell_inst | lib_cell

Identifies the type of object specified in the **-objects** option.

-tie_high_lib_cell *lib_cell*

Specifies the reference cell used to instantiate the tie-high cells. The tool determines the number of tie-high cells to instantiate. Optional

-tie_low_lib_cell *lib_cell*

Specifies the reference cell used to instantiate the tie-low cells. The tool determines the number of tie-low cells to instantiate. Optional

-tie_highlow_lib_cell *lib_cell*

Specifies the reference cell used to instantiate the tie-highlow cells. The tool determines the number of tie-highlow cells to instantiate. Optional

-tie_high_port_name *port*

Specifies the name of the port that is logic high on tie-highlow cell. Required if **-tie_highlow_lib_cell** flag is used.

`-tie_low_port_name port`
 Specifies the name of the port that is logic low on tie-highlow cell. Required if `-tie_highlow_lib_cell` flag is used.

`-max_fanout number`
 Specifies the maximum number of ports that can be driven by single tie-high or tie-low cell.
 If you do not specify this option, the maximum fanout is 8.

`-max_wirelength number`
 Specifies the maximum Manhattan wirelength in microns from a tie-high or tie-low cell to each driven cell port.
 If you do not specify this option, the maximum wirelength is 100.

`-incremental true | false`
 Specifies whether to run in incremental mode.
 By default (false), the command removes all existing tie cells of the specified ports. In incremental mode, existing tie cells that satisfy the wirelength and fanout constraints are not deleted. This provides minimum disruption to the existing placement and routing.

DESCRIPTION

This command specifies the cell ports to which to connect to tie-high and tie-low cells. It instantiates the tie-high and tie-low cells and makes the necessary connections. Before you run this command, the design must be fully placed and connected to power and ground.

A tie-high cell is a special standard cell whose output pin is always at logic high. A tie-low cell is a special standard cell whose output pin is always at logic low. A tie-highlow cell is a combination cell with two output pins, one at logic high; the other at logic low. In some designs, unused input pins cannot be tied directly to the power or ground net and tie-high, tie-low, or tie-highlow cells must be used.

You may choose to specify the library reference cells associated with the tie-high and tie-low cells only. Then the ports that are tied off to power or ground will be connected to tie cells. Alternately, you may provide the library reference cell of the tie-highlow cell instead. If a two-pin tie-highlow cell is desired you must specify the port names of the pins that are logic high and low. Then the ports tied off to power or ground will be connected to the high or low, respectively, port of the tie-highlow cell. You may also specify both one output tie-high and tie-low library reference cells and two output tie-highlow library reference cell. In that case, in regions of the chip where both ports tied off to power and ground are present, the tool will use tie-highlow cells, otherwise tie-high or tie-low cells will be chosen instead. If a tie-highlow option is not used, then both tie-high and tie-low options must be used.

The cell ports to connect to the tie-high and tie-low cells are inferred from the specified list of cell instances, port instances, or reference cells.

You can specify maximum fanout (the maximum number of ports that can be driven by a single tie-high or tie-low cell) and maximum wirelength (the maximum Manhattan wire length between a tie-high or tie-low cell and a driven port) constraints.

The command can also be run in incremental mode.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLE

In the following example, all input ports of cells "BONUS_SET_*/*" that are tied off to power or ground are connected to new tie-high and tie-low cells.

```
prompt> connect_tie_cells -objects [get_cells "BONUS_SET_*/*"] \  
      -obj_type cell_inst \  
      -tie_high_lib_cell TIEH -tie_low_lib_cell TIEL
```

In the following example, all input ports named "A" of cells "BONUS_SET_*/*" that are tied off to power or ground are connected to new or existing tie-high and tie-low cells that are at most 64 microns of Manhattan distance away.

```
prompt> connect_tie_cells -objects {"BONUS_SET_*/*/A"} \  
      -obj_type port_inst \  
      -tie_high_lib_cell TIEH -tie_low_lib_cell TIEL \  
      -max_wirelength 64 -incremental true
```

In the following example, all input ports of cells "BONUS_SET_*/*" that are tied off to power or ground will be connected to new tie-highlow cells. Ports that are tied off to power will be connected to port "OUT1" on cell TIEHL. Ports that are connected to ground will be connected to port "OUT0" on cell TIEHL.

```
prompt> connect_tie_cells -objects [get_cells "BONUS_SET_*/*"] \  
      -obj_type cell_inst \  
      -tie_highlow_lib_cell TIEHL \  
      -tie_high_port_name OUT1 \  
      -tie_low_port_name OUT0
```

convert_from_polygon

Converts a polygon into a list of mutually exclusive rectangles.

SYNTAX

```
list convert_from_polygon  
polygon  
[-format polygon | rectangle]
```

Data Types

polygon list

ARGUMENTS

polygon

Specifies the polygon to be decomposed into rectangles. You must specify the polygon in the following format: {{x1 y1} {x2 y2} ... {xN yN} {x1 y1}}. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point. The coordinate unit is specified in the technology file; typically it is microns.

-format *polygon* | *rectangle*

Specifies the output format of resulting rectangles.

By default, each resulting rectangle is represented in rectangle format: {x1 y1} {x2 y2}.

If you specify **-format polygon**, each resulting rectangle is represented in polygon format: {{x1 y1} {x2 y2} {x3 y3} {x4 y4} {x1 y1}}.

DESCRIPTION

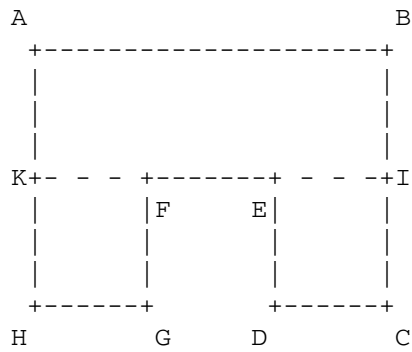
This command converts a polygon into a list of mutually exclusive rectangles. Each returned rectangle is represented as a Tcl list of coordinates.

Note that this command might return a list of polygons that represents a disjoint rectilinear region. Because the polygon commands take a single polygon as input, you cannot directly pass the result as a parameter to another polygon command. Instead, you must use a Tcl list command, such as the **foreach** or **lindex** command, to extract each polygon from the returned list, and then pass each polygon to the polygon command.

You must open the Milkyway design library before using this command.

EXAMPLES

The following example converts the A-B-C-D-E-F-G-H-A polygon to three rectangles, K-F-G-H, E-I-C-D, and A-B-I-K.



```
prompt> convert_from_polygon {{0 20} {30 20} {30 0} {20 0} {20 10}
{10 10} {10 0} {0 0} {0 20}}
{{0 0} {10 10}}
{{20 0} {30 10}}
{{0 10} {30 20}}
```

SEE ALSO

```
convert_to_polygon(2)
compute_polygons(2)
resize_polygon(2)
```

convert_mw_lib

Converts a Milkyway library's cell data.

SYNTAX

```
status convert_mw_lib  
mw_lib  
[-cell_name cell_name]  
[-all]  
[-previous]
```

Data Types

<i>mw_lib</i>	string
<i>cell_name</i>	string

ARGUMENTS

mw_lib
Specifies the library containing the cells to be converted.

-cell_name *cell_name*
Specifies the name of cell to be converted. If the option is specified, all child soft macro cells referred by the current cell will be converted along with the current cell.

-all
If the option is specified, all cells under the design lib will be converted.

-previous
If the option is specified, the cell would be converted to the previous schema version. This option is only supported in IC Compiler.

DESCRIPTION

This command converts cells of the specified library. It does not convert cells in the reference libraries.

If a cell was created by a prior release, it is old and must be updated before it is opened by the current release. Once updated, the cells may be used by any tool from the current release.

It is recommended to convert all cells once using this `convert_mw_lib` command. This minimizes run-time and memory consumption for conversion and would have less impact to the ICC flow. If an old cell is not converted, the `open_mw_cel` command will open the old cell and automatically convert it in memory. If the user does not save the cell, the cell will be automatically converted again during the next `open_mw_cel` command call.

EXAMPLES

```
convert_mw_lib foo_library -cell_name top  
1  
convert_mw_lib foo_library -all -previous  
1
```

SEE ALSO

```
open_mw_cel(2)  
save_mw_cel(2)
```


convert_to_polygon

Returns a polygon for the specified object.

SYNTAX

```
list convert_to_polygon
[-quiet]
object_spec
```

Data Types

object_spec collection of one item

ARGUMENTS

-quiet
 Suppresses reporting of error and warning messages.

object_spec
 Specifies a single object from which to get the polygon. You can specify only a single object, which must be a shape, placement blockage, or route guide.

DESCRIPTION

This command returns a polygon calculated from the input object.

The returned polygon is represented as a Tcl list of coordinates. If the input object is a polygon, this command outputs the points of the polygon. If the input object is a rectangle or path, this command calculates the corresponding polygon.

EXAMPLES

The following example returns a polygon from the RECTANGLE#123 net shape.

```
prompt> convert_to_polygon [get_net_shapes RECTANGLE#123]
{{99.490 111.340} {117.590 111.340} {117.590 111.620} {117.590 99.490}
{99.490 111.340}}
```

SEE ALSO

`convert_from_polygon(2)`
`resize_polygon(2)`

convert_wire_ends

Converts the currently opened cell's signal wire ends.

SYNTAX

status_value **convert_wire_ends**

ARGUMENTS

no arguments

DESCRIPTION

This command searches the currently opened cell for signal wires having zero length extensions. Each of these wires is converted into a wire having half width extensions. If the wire is not convertible (i.e., the wire length is less than the width), the wire is deleted.

This command only modifies wires having a signal route type.

The command returns a status indicating success or failure.

EXAMPLES

The following example converts the "top" cell's zero length extension wires into half width extension wires.

```
prompt > open_mw_lib myLib
{myLib}
prompt > open_mw_cel top
{top}
prompt > convert_wire_ends
Warning: Removed net shape (VWIRE#1796726). (MWUI-140)
Warning: Removed net shape (VWIRE#1687148). (MWUI-140)
Warning: Removed net shape (VWIRE#1687150). (MWUI-140)
1
prompt > save_mw_cel
Information: Saved design named top.CEL;1. (UIG-5)
1
prompt > close_mw_cel
1
prompt > close_mw_lib
1
```

convert_wire_to_pin

Converts wires to pins for either all nets or specified nets in the selected wire set.

SYNTAX

```
status convert_wire_to_pin  
[-net_names net_names]  
objects
```

Data Types

<i>net_names</i>	collection
<i>objects</i>	collection

ARGUMENTS

`-net_names net_names`
Specifies net names to filter selected wires. The *net_names* argument is a collection of net names.

`objects`
Specifies a selected wire set as a collection of wires.

DESCRIPTION

This command searches the specified nets and converts the selected wires of the nets into pins. You must manually select the wires by GUI first.

The successive command **create_macro_fram** can extract those detailed pins converted from wires in the FRAM view. If the *net_names* argument is NULL, all selected wires are converted to their corresponding pins. Otherwise, only the selected wires that belong to the specified *net_names* are converted to pins.

NOTES

Only the wire section connecting to a pin is converted to a pin. If no corresponding pin of selected wires exists in the top cell, nothing is converted. That is, no new pin is created in the top cell (no net list changes).

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example converts user-selected wires of the nets named *OR* and *SCEN*) to pins.

```
prompt> convert_wire_to_pin -net_names {OR SCEN} [get_selection]
```

SEE ALSO

```
create_macro_fram(2)  
get_selection(2)
```

copy_collection

Duplicates the contents of a collection, resulting in a new collection. The base collection remains unchanged.

SYNTAX

```
collection copy_collection  
collection1
```

Data Types

```
collection1      collection
```

ARGUMENTS

```
collection1  
    Specifies the collection to be copied. If the empty string is used for the  
    collection1 argument, the command returns the empty string: A copy of the  
    empty collection is the empty collection.
```

DESCRIPTION

Creates a duplicate of an existing collection by using an efficient mechanism. It is more efficient, and almost always sufficient, to simply have more than one variable reference the same collection, as described below. For example, if you create a collection and save a reference to it in a variable named *c1*, assigning the value of *c1* to another variable named *c2* simply creates a second reference to the same collection:

```
prompt> set c1 [get_cells "U1*"]  
{"U1", "U10", "U11", "U12"}  
prompt> set c2 $c1  
{"U1", "U10", "U11", "U12"}
```

This procedure creates two references to the same collection, but this has not copied the collection. If you change the *c1* variable, *c2* continues to reference the same collection:

```
prompt> set c1 [get_cells "block1"]  
{"block1"}  
prompt> query_objects $c2  
{"U1", "U10", "U11", "U12"}
```

However, there might be instances when you really do need a copy; and, in those cases, you can use the **copy_collection** command to create a new collection that is a duplicate of the original.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows the result of copying a collection. Functionally, it is not much different than having multiple references to the same collection.

```
prompt> set c1 [get_cells "U1*"]
{"U1", "U10", "U11", "U12"}
prompt> set c2 [copy_collection $c1]
{"U1", "U10", "U11", "U12"}
prompt> unset c1
prompt> query_objects $c2
{"U1", "U10", "U11", "U12"}
```

SEE ALSO

`collections(2)`

copy_floorplan

Copies floorplan data from a specified design to the current design in a Milkyway design library.

SYNTAX

```
status copy_floorplan
[-library design_library_name]
-from design_name
[-macro]
[-filler]
[-pad]
[-power_plan]
[-verbose]
[-incremental]
```

Data Types

<i>design_library_name</i>	string
<i>design_name</i>	string

ARGUMENTS

-library *design_library_name*
Specifies the name of the Milkyway design library for the source and destination designs. If this option is not specified, the current open library is used. If no library is currently open, this option is required.

-from *design_name*
Specifies the name of the source design. The source design must be specified, otherwise, the **copy_floorplan** command will not execute.

-macro
Copy placement of macros. Placement here includes position, orientation, fixed or placed status and so on.

-filler
Copy placement of filler cells. Placement here includes position, orientation, fixed or placed status and so on.

-pad
Copy placement of I/O pads, corner pads and flip chip pads. Placement here includes position, orientation, fixed or placed status and so on.

-power_plan
Copy power net connections and routing.

-verbose
Print detailed message and statistics on data copied and not copied.

-incremental
Do not update the existing named objects in the destination design. Can be

combined with all other options.

DESCRIPTION

The **copy_floorplan** command copies floorplan data from a specified design to the current design in a Milkyway design library, and the current design must be opened before the command is executed. Floorplan data includes die area, core area, rows, wire tracks, placement and routing obstructions, bounds, groups and regions, power domains and voltage areas, the XY coordinates of cell instances and top level pins, routes, and power net connections.

Use this command after floorplanning is done, but before standard cell placement and routing is done, which is an early stage in an implementation flow.

The source design and the current design are Milkyway cells and must exist in the same design library. And you must specify the **-from** option to indicate the source design. If you do not specify the type of data to copy (**-macro**, **-filler**, **-pad**, or **-power**), the command copies all existing floorplan data from the source design to the current design. Otherwise, only the data specified by the options is copied.

The command does not do a complete netlist comparison. It goes through the netlist of the source design and looks for the objects by name in the destination design. By default, if a cell instance or net does not exist in the destination design, it is skipped.

The command returns 1 on success, 0 on failure.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example copies the macro and pad placement information from design A to the current design in design library mwlib.

```
prompt> copy_floorplan -library mwlib -from A -macro -pad
```

The following example copies all floorplan information from design A to the current open design, printing detailed information about the copy process.

```
prompt> copy_floorplan -from A -verbose
```

SEE ALSO

`write_floorplan(2)`

copy_mim

Copies cell placement, blockages, or shape from one multiple instantiated module (MIM) plan group to others in same group.

SYNTAX

```
status copy_mim
[-type placement | blockage | boundary]
[-restore_placement]
collection
```

Data Types

collection list containing one plan group

ARGUMENTS

`[-type placement | blockage | boundary]`
Identifies the type of copy desired. The **placement** argument copies the cell placement of the given MIM to all other MIMs in the same group. The **blockage** argument copies all types of placement blockages overlapping the given MIM to all other MIMs in the same group. The **boundary** argument copies the MIM shape from the given MIM to all other MIMs in the same group. The default is **placement**.

`[-restore_placement]`
Restores the placement of cells in the MIM group of the MIM specified by **placement**. This option undoes the result of the previously-executed **copy_mim -type placement** command. This works even if you exit and restart the tool as the original placement (that is, before the `copy_mim -type placement`) is saved to the database.

collection
Specifies a collection containing one MIM plan group. This is the source MIM to copy to all other MIMs in the same group.

DESCRIPTION

This command performs various manipulations on MIMs. MIMs are modules that have the same reference. A set of these modules is called a MIM group. A design can have more than 1 MIM group. For example, if modules A1 and A2 have reference A and modules B1 and B2 have reference B, they form 2 MIM groups.

The command performs manipulations during the virtual flat stage. This is the period when MIMs are physically represented in the design as plan groups. You can use this command to create MIMs with the same cell placement, blockages, and shapes.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example copies the placement of a MIM plan group, M1, to MIM plan groups M2 and M3.

```
prompt> copy_mim -type placement M1
```

The following example assumes you have created placement blockages in M1 that you want copy to M2 and M3.

```
prompt> copy_mim -type blockage M1
```

The following example shows how to undo the placement of M1, M2, and M3; that is, to restore placement to the state before the execution of a **copy_mim -type placement M1** command.

```
prompt> copy_mim -restore_placement M1
```

Note that this command undoes only the placement, not the blockage copy.

SEE ALSO

`flip_mim(2)`
`report_mim(2)`

copy_mw_cel

Copies Milkyway designs from a source design library to a target design library.

SYNTAX

```
status copy_mw_cel  
-from source_mw_cel_name  
[-to target_mw_cel_name]  
[-from_library source_library_name]  
[-to_library target_library_name]  
[-hierarchy]  
[-check_only]  
[-overwrite]
```

Data Types

<i>source_mw_cel_name</i>	collection
<i>target_mw_cel_name</i>	string
<i>source_library_name</i>	string
<i>target_library_name</i>	string

ARGUMENTS

-from *source_mw_cel_name*

Specifies the Milkyway designs to be copied. If you use the **-hierarchy** option, you can specify only a single design. If you do not use the **-hierarchy** option, you can specify a single design or multiple designs.

You can specify a Milkyway design by name, pattern, or collection. For example, *top* matches a Milkyway design named *top* in the current design library; *top** matches all Milkyway designs whose names start with *top*; and *** specifies all Milkyway designs in the current design library.

If you specify a version in the source file name, only that version of the Milkyway design is copied. If you do not specify a version in the source file name, the latest version of the Milkyway design is copied.

-to *target_mw_cel_name*

Specifies the name of the target Milkyway design.

If you specify multiple source Milkyway designs in the **-from** option, the tool assumes that you want to copy them without changing the names, so it ignores this option.

If you use both this option and the **-hierarchy** option, specify only the top-level design name as *target_mw_cel_name*, because it is the only Milkyway design name that can be changed.

If you do not specify this option, the source Milkyway designs are copied to the target library without changing the name.

-from_library *source_library_name*

Specifies the design library that contains the source Milkyway designs.

If you use patterns, such as *"top"* or *"*"*, to specify the source Milkyway designs in the **-from** option, the tool ignores this option. All patterns are matched in the current design library.

If you do not specify this option, the tool uses the current design library

as the source library.

-to_library *target_library_name*

Specifies the target library to which the Milkyway designs are copied. Generally, you do not need to open the target library before running this command.

If you use both this option and the **-hierarchy** option, and the specified target design library does not exist, the tool creates a new library and copies the hierarchical design into it.

If you do not specify this option, the tool uses the current design library as the target library.

-hierarchy

Copies the source Milkyway design and its subdesigns to the target design library.

-check_only

Prints the list of Milkyway designs, views, and versions but does not copy the designs. This option is valid only when used with the **-hierarchy** option.

-overwrite

Overwrites the target Milkyway design when copying. This option cannot be used with the **-hierarchy** option.

DESCRIPTION

This command copies the specified Milkyway designs from the source design library to the target design library. You must have write permission for the target design library. Only read permission is required for the source design library.

You can use the **-hierarchy** option to copy both the top-level design and its subdesigns. Subdesigns that are instantiated from the reference library stay in the reference library and are not copied to the target design library. The target library sets the reference path to those reference libraries. The latest version of any open and unsaved Milkyway design in the hierarchical tree is saved from memory into the target design library. If the open Milkyway design is not the latest version, the Milkyway design is copied from disk instead of memory. If the Milkyway design is not open, the Milkyway design on disk is copied.

The copied Milkyway designs include all information from the original Milkyway design. The copied Milkyway designs have their versions set to 1 if the target library is a new library. If the target library is not a new library, the version of the copied Milkyway design is incremented by 1 from the latest version before the copy, unless you use the **-overwrite** option.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example copies version 2 of a design named test to a design named

bill. Both test and bill are in the current design library.

```
prompt> copy_mw_cel -from "test.CEL;2" -to bill  
1
```

The following example copies all Milkyway designs from the current library to another library named new_lib.

```
prompt> copy_mw_cel -from * -to_library new_lib  
1
```

The following example copies the design named ORCA and all of its subdesigns from the current library to a library named lib1.

```
prompt> copy_mw_cel -hierarchy -to_library lib1 -from ORCA  
Copied Library Files  
From: /orca/orca_lib  
To: lib1  
  
Copied Cells  
From: /orca/orca_lib  
To: lib1  
  
Copying cell: ORCA.CEL;1  
Copying cell: BLENDER_2;  
1
```

SEE ALSO

```
close_mw_cel(2)  
create_mw_cel(2)  
current_mw_cel(2)  
get_mw_cels(2)  
open_mw_cel(2)  
remove_mw_cel(2)  
rename_mw_cel(2)
```

copy_mw_lib

Copies a Milkyway library to another location.

SYNTAX

```
status copy_mw_lib
[-from mw_lib]
-to lib_name
```

Data Types

<i>mw_lib</i>	string
<i>lib_name</i>	string

ARGUMENTS

-from *mw_lib*
Specifies the name of the source Milkyway library to be copied. The *mw_lib* value can be a library name or a collection of libraries. The **-from** and **-from_lib_id** options are mutually exclusive. By default, the command uses the current Milkyway library.

-to *lib_name*
Specifies the destination Milkyway library name.

DESCRIPTION

This command copies a Milkyway library to another location. It returns a status indicating success or failure.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example copies a Milkyway library design to another location.

```
prompt> copy_mw_lib -from design -to design.bak
1
```

SEE ALSO

rename_mw_lib(2)

copy_objects

Copies one or more objects.

SYNTAX

```
new_objects copy_objects
[-delta vector]
[-to point]
[-use_same_net]
objects
```

Data Types

objects collection

ARGUMENTS

-delta vector
Specifies the displacement to move the copied object from the original object's position.

-to point
Specifies the absolute position of the copied object.

-use_same_net
Specifies whether the copied object should be connected to the same net as the original object. If not specified the object is unconnected.
Note: Only applicable to objects that can have net connections.

objects
Specifies the objects to be copied.

RETURNS

new_objects
Contains a collection of the copied objects.

DESCRIPTION

This command copies one or more objects and can optionally place the new objects at a new location or a delta from the original location.

In general all attributes of an object are copied. There are a few exceptions to this rule:

- The new object will have a different database id than the original (this is required as all database ids must be unique).

- The new object will be, by default, not be connected to the same net as the original object unless the user specifies the **-use_same_net** option.

- The new object will be at a different position if either the **-delta** or **-to** options are specified.

See `get_edit_property(2)` man page for more details on which objects can be copied.

Note: Snapping of the position of the new object is done automatically using global snap settings.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example copies the currently selected object placing the copied object 100 units to the right and 200 units above the original object.

```
prompt> copy_objects -delta {100 200} [get_selection]
```

SEE ALSO

```
move_objects(2)
remove_objects(2)
resize_objects(2)
rotate_objects(2)
align_objects(2)
distribute_objects(2)
expand_objects(2)
flip_objects(2)
set_object_snap_type(2)
get_object_snap_type(2)
```


count_drc_violations

Returns the number of design rule constraint (DRC) violations.

SYNTAX

```
int count_drc_violations  
[-intersect bbox]  
[-ignoring bbox]  
[-include_types types]  
[-ignore_types types]  
[-include_layers layers]  
[-ignore_layers layers]  
[-use_new_drc]  
[-drc_cell_name drc_cell_name]  
[-return_error_type]  
[-verbose]
```

Data Types

<i>bbox</i>	bounding box
<i>types</i>	string or list of strings
<i>layers</i>	string or list of strings
<i>drc_cell_name</i>	string

ARGUMENTS

-intersect *bbox*
Include DRC violations within or touching this bounding box.

-ignoring *bbox*
Ignore DRC violations completely within this bounding box.

-include_types *types*
Include DRC violations of these types. Use -verbose to find type names.

-ignore_types *types*
Ignore DRC violations of these types. Use -verbose to find type names.

-include_layers *layers*
Include DRC violations on these layers. Use -verbose to find layer names.

-ignore_layers *layers*
Ignore DRC violations on these layers. Use -verbose to find layer names.

-use_new_drc
Use the geNewdrc DRC error cell. This is already the default.

-drc_cell_name *drc_cell_name*
Specify a particular DRC error cell name. Use topCellName for the verify_route results. Use topCellName_aDRC.err for the verify_drc results.

`-return_error_type`
Returns the name of the error type found (if there are multiple errors , it returns the first one).

`-verbose`
Report all the DRC violations in full form.

DESCRIPTION

This command reports the number of DRC errors of the specified types, on the specified layers, within the given domain. It also lists the DRC error if `-verbose` is selected.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

```
prompt> count_drc_violation -verbose
```

SEE ALSO

`verify_route(2)`
`verify_drc(2)`

cputime

Reports the CPU time in seconds.

SYNTAX

```
int cputime  
[-all]  
[-verbose]
```

ARGUMENTS

-all

Report the total CPU time of the main process and its child processes.

-verbose

Report in detail the CPU time and elapsed (wall-clock) time of the main process and each child process, including placement, extraction, and routing.

DESCRIPTION

This command reports the CPU time in seconds. By default it reports the CPU time for the main process. If you specify **-all**, it reports the total CPU time of the main process and all its child processes. If you specify both **-all** and **-verbose**, it reports the CPU time and elapsed (wall-clock) time for the main process and each child process. The runtime numbers are collected from the operating system.

When you use the **-num_cpus** option, the tool runs multiple threads. Currently the operating system measures the CPU time by adding the usage for all threads. It does not take into account parallelization of the threads. This means that the report might show a larger CPU time than elapsed time when you specify the **-num_cpus** option. The elapsed time (wall-clock time) is more appropriate for measuring multicore runtime.

The elapsed time depends on many external factors and can vary for every session. It depends on factors such as the I/O traffic, the network traffic, RAM and swap usage, and the other processes running on the same machine. When the elapsed time is much longer than the CPU time, it might point out an inefficiency of the computing environment, such as insufficient memory, too many processes running on the same machine, or a slow network. The only way to make the elapsed time close to the CPU time is to run only one session on a machine with enough RAM and using the local disk.

EXAMPLES

The following example shows the default command output.

```
prompt> cputime  
1202
```

The following example shows the output when using the **-all** and **-verbose** options.

```
prompt> cputime -all -verbose
Main process  CPU: 1202 s (0.33 hr) Elapse: 1800 s (0.50 hr)
Child "placement" called 3 times, total CPU: 200 s (0.06 hr) Elapse: 250 s (0.07
hr)
Child "extraction" called 2 times, total CPU: 100 s (0.03 hr) Elapse: 150 s (0.0
4 hr)
Total CPU:  1502 s ( 0.42 hr) Elapse: 1800 s ( 0.50 hr)

1523
```

SEE ALSO

```
mem(2)
monitor_cpu_memory(3)
```

create_auto_shield

Performs automatic shield routing.

SYNTAX

```
status create_auto_shield
[-with_ground net_name]
[-nets collection_of_nets]
[-ignore_shielding_net_pins]
[-ignore_shielding_net_rails]
[-coaxial_below]
[-coaxial_above]
```

Data Types

<i>net_name</i>	string
<i>collection_of_nets</i>	collection

ARGUMENTS

- with_ground *net_name***
Specifies which power or ground net to tie the shielding wires to.
By default, the shielding wires are tied to the ground net. If the design contains multiple ground nets, you must use this option to specify the ground net.
- nets *collection_of_nets***
Specifies the nets to be shielded. If the specified nets do not have shielding rules defined, the tool uses the default width and minimum spacing for the layer as defined in the technology file.
By default, this command performs shielding on all nets with defined shielding rules. To define the shielding rules, use the **define_routing_rule** and **set_net_routing_rule** commands.
- ignore_shielding_net_pins**
Prevents connection of the shielding wires to the standard cell power or ground (PG) pins.
By default, the shielding wires are connected to the standard cell PG pins.
- ignore_shielding_net_rails**
Prevents connection of the shielding wires to the standard cell rails.
By default, the shielding wires are connected to the standard cell rails.
- coaxial_below**
Creates coaxial shielding below the shielded net segment layer.
By default, this command does not perform coaxial shielding below the shielded net segment layer.
- coaxial_above**
Creates coaxial shielding above the shielded net segment layer.
By default, this command does not perform coaxial shielding above the shielded net segment layer.

DESCRIPTION

The **create_auto_shield** command shields nets based on the associated shielding rules.

By default, the command performs same-layer shielding on all nets with predefined shielding rules. The router routes shielding wires based on the shielding widths and spacing defined in the shielding rules. The shielding wires are tied to the ground net, standard cell ground pins, and standard cell rails.

To explicitly specify the nets on which to perform shielding, use the **-nets** option. To tie the shielding wires to a named power or ground net, use the **-with_ground** option. To prevent connections to the standard cell PG pins, use the **-ignore_shielding_net_pins** option. To prevent connections to the standard cell rails, use the **-ignore_shielding_net_rails** option.

By default, the command does not perform coaxial shielding. To perform coaxial shielding above the shielded net segment layer, use the **-coaxial_above** option. To perform coaxial shielding below the shielded net segment layer, use the **-coaxial_below** option. When the tool performs coaxial shielding, the coaxial shielding segments

- * Are routed in the preferred layer direction
- * Use the defaultWidth layer attribute from the technology file
- * Are spaced such that they leave at least one signal routing resource on each layer

PREREQUISITES

Before you run this command, you must first define the shielding rules with the **define_routing_rule** command and assign these rules to the nets to be shielded with the **set_net_routing_rule** command.

This requirement applies when you do not specify the **-nets** option, you want to use a nondefault width or minimum spacing, or you want to snap the shielding segments to the grid.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example defines the shielding rules; assigns these shielding rules to all clock nets; and then shields the CLK_B1 clock net with the VSS ground net, the CLK_B2 clock net with the GND ground net, and the CLK_B3 and CLK_B4 clock nets with the VDD power net. The shielding wires are not connected to the standard cell PG pins.

```
prompt> define_routing_rule clock_net_shielding \  
-default_reference_rule -taper_level 0 -snap_to_track \  
-widths {METAL1 0.16 METAL2 0.2 METAL3 0.2 METAL4 0.2} \  
-spacing {METAL1 0.18 METAL2 0.21 METAL3 0.21 METAL4 0.21} \  

```

create_auto_shield

```
-shield_width {METAL1 0.16 METAL2 0.2 METAL3 0.2 METAL4 0.2} \  
-shield_spacing {METAL1 0.18 METAL2 0.21 METAL3 0.21 METAL4 0.21} \  
-via_cuts {via1 1X1 via2 1X1 via3 1X1}
```

```
prompt> set_clock_nets [get_nets -filter "net_type == Clock" -hier]
```

```
prompt> set_net_routing_rule -rule clock_net_shielding $clock_nets
```

```
prompt> create_auto_shield -with_ground VSS -nets {CLK_B1} \  
-ignore_shielding_net_pins
```

```
prompt> create_auto_shield -with_ground GND -nets {CLK_B2} \  
-ignore_shielding_net_pins
```

```
prompt> create_auto_shield -with_ground VDD -nets {CLK_B3 CLK_B4} \  
-ignore_shielding_net_pins
```

SEE ALSO

```
define_routing_rule(2)  
set_net_routing_rule(2)
```

create_base_array

Creates a base array record in the design.

SYNTAX

```
status create_base_array
[-tile_name tile_name]
-coordinate rectangle
[-direction direction]
```

Data Types

<i>tile_name</i>	string
<i>rectangle</i>	list
<i>direction</i>	string

ARGUMENTS

-tile_name *tile_name*
Specifies the name of unit tile used in the base array record.
If **-tile_name** is not specifies, by default it creates a base array using "unit" as its unit tile.

-coordinate *rectangle*
Specifies the lower left corner and the upper right corner of the base array.
The values are specified in microns relative to the chip origin.

-direction *direction*
Specifies the direction of the base array record.
The validated values of direction can be: horizontal and vertical.
By default, the direction is set to horizontal.

DESCRIPTION

The **create_base_array** command allows you to create a base array record into database.

This command returns 1 if succeeds.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates a base array record.

```
prompt> create_base_array -coordinate {{200 200} {1800 1800}} -
```



```
direction horizontal
-tile_name unit
1
```

SEE ALSO

```
remove_base_arrays(2)
```

create_boundary

Creates a boundary for a design or library cell.

SYNTAX

```
status create_boundary
[-coordinate rectangle
 | -poly {point point ...}
 | -by_terminal]
[-core]
[-left_offset l_offset]
[-right_offset r_offset]
[-top_offset t_offset]
[-bottom_offset b_offset]
[-lib_cell_type type
```

ARGUMENTS

-coordinate *rectangle*

Creates a boundary using the specified bounding box. This option is mutually exclusive with **-poly** and **-by_terminal**; you can only use one of them. If you do not specify any of these options, the boundary is created according to current boundary. You cannot specify this argument with the **-lib_cell_type** option.

-poly {*point point ...*}

Creates a rectilinear boundary with *n* points. This option is mutually exclusive with **-coordinate** and **-by_terminal**; you can only use one of them. If you do not specify any of these options, the boundary is created according to current boundary. You cannot specify this argument with the **-lib_cell_type** option.

-by_terminal

Creates a boundary based on the bounding box of all the terminals in the design (if you specify *design*) or the library cells (if you specify **-lib_cell_type**).

-core

Indicates that **-coordinate** also defines the core area for design planning.

-left_offset *l_offset*

Specifies the distance to adjust the left side of the boundary.

-right_offset *r_offset*

Specifies the distance to adjust the right side of the boundary.

-top_offset *t_offset*

Specifies the distance to adjust the top of the boundary.

-bottom_offset *b_offset*

Specifies the distance to adjust the bottom of the boundary.

-lib_cell_type *type*

Specifies the type of library cell for which to create boundaries. If you also specify **-by_terminal**, the boundary is created based on the terminal locations; otherwise the boundary is created based on the current boundary. This option and the *design* argument are mutually exclusive. You cannot use this option when you specify the **-coordinate** or **-poly** options.

DESCRIPTION

The **create_boundary** command allows you to create a boundary for a design or library cells. If you specify **-lib_cell_type**, a boundary is created for all library cells of the specified type. If you specify **design**, a boundary is created for the specified design. If you do not specify either **-lib_cell_type** or *design*, a boundary is created for the current design.

For a design, you can specify the boundary by using the **-coordinate**, **-poly**, or **-by_terminal** option. If you do not specify how to create the boundary, the boundary is created based on the current boundary.

For library cells, you can specify the boundary only by using the **-by_terminal** option.

You can also adjust the boundary by using the **-left_offset**, **-right_offset**, **-top_offset**, and **-bottom_offset** options. However, these four options cannot be used with **-poly**.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command creates a boundary for the current design by specifying a bounding box.

```
prompt> create_boundary -coordinate {{0 0} {100 100}}
1
```

The following command creates a boundary for the current design by specifying a point array.

```
prompt> create_boundary -poly {{0 0} {100 0} \
{100 100} {50 100} {50 50} {0 50}}
1
```

The following command creates a boundary for the current design based on its terminal locations and adjusts the left side of the boundary.

```
prompt> create_boundary -by_terminal -left_offset 10
```

1

The following command creates a boundary for all io_pad cells in the reference library.

```
prompt> create_boundary -lib_cell_type io_pad  
1
```

SEE ALSO

create_bounds

Creates a fixed move bound or floating group bound in the design.

SYNTAX

```
int create_bounds
[-name bound_name]
[-coordinate {llx1 lly1 urx1 ury1 ...}]
[-dimension {width height}]
[-effort low | medium | high | ultra]
[-type soft | hard]
[-exclusive]
[-color range_0_to_63]
[-cycle_color]
object_list
```

ARGUMENTS

- name *bound_name***
Specifies the name of the bound.
- coordinate {*llx1 lly1 urx1 ury1 ...*}**
Specifies a list of lower left and upper right coordinates of a move bound. Each combination of {*llx lly urx ury*} defines a target placement area for the objects. The areas can overlap or be disjoint. Rectilinear move bounds are allowed, but must be divided into a number of individual rectangular bounds. These rectangles bounds should be specified in this coordinate list. The coordinates are in microns relative to the chip origin.
- dimension {*width height*}**
Specifies the dimension of a group bound. The numbers are in microns.
- effort low | medium | high | ultra**
Specifies the effort to bring cells closer inside a group bound. The default effort is medium.
- type soft | hard**
Specifies the type of the bound to be either hard or soft. Either **-coordinate** or **-dimension** must be specified to use this switch. It is not allowed to set the bound type on auto group bounds. The default is to create soft bounds.
- exclusive**
Assigns a property type of exclusive to the move bound being created. Move bounds that are exclusive require all of their cells to be placed inside them and prohibit the placement of other cells in the same area. Exclusive move bounds will be respected by both coarse placement and legalization.
- color *range_0_to_63***
Specifies the move bound color. Substitute a value for *range_0_to_63* that is in the range of **0** to **63**, inclusive. The default is **no_color** which is the special value for no user specified color.

`-cycle_color`

Allows the tool to automatically assign a color. By default, this option is off.

`object_list`

Specifies a list of cells/ports/pins to be included in the bound. If a cell is a hierarchical cell, the bound also be applied to all cells in the subdesign. It is an error to include a cell for a move bound which has already been assigned to another move bound. The cell list is not mandatory for move bound. User can create an empty move bound without specifying any cell lists. If a port is a hierarchical port, the corresponding port will be attached to the bound instead. If a pin given, the lower left point of the first geometry of the pin will be marked on the cell of the pin, and the cell will be added to the bound. If a pin is a hierarchical pin, it will be ignored.

DESCRIPTION

The `create_bounds` command allows the user to define region-based placement constraints for coarse placement. There are two different types of bounds available: move bounds and group bounds. Move bounds restrict the placement of cells to a specific region of the core area. Move bounds require absolute coordinates to be specified, using the `-coordinate` option. Group bounds, on the other hand, are floating region constraints. Cells in the same group bound will be placed within a specified bound but the absolute coordinates are not fixed. Instead, they are optimized by the placer. The `-dimension` option needs to be specified for a group bound.

Generally, there is no guarantee that cells will be placed completely within bounds. For instance, coarse placement will violate bounds if the quality of its primary placement objectives would otherwise be destroyed.

In these situations, the user should revisit their bounds and floorplan to ensure that the design is not overconstrained. Alternatively, the user can use the `"-type hard"` option to specify the bound type to be hard (the default is soft). The coarse placer will try to honor the hard bound as hard constraints while sacrificing other objectives, such as timing and routability. It is not recommended to use many hard bounds because this will lead to inferior placement solutions.

If `-coordinate` is given, a move bound will be created. If `-dimension` is given, a group bound with the given bounding box will be created. If neither of these two switches is used, a group bound will be created with a bounding box computed internally by the tool. In this case, the user can use `-effort` to specify the effort level to bring cells closer. It is not allowed to use `-effort` when either `-coordinate` or `-dimension` is used. All automatically generated bounds are soft bounds. It is not allowed to use `-type` to change the bound type of these auto group bounds.

If `-coordinate` is given and no cell list is provided as an argument, then an empty move bound will be created. User can later on associate the cells with the particular move bound by using `attach_bounds` command.

If more than one placeable area is defined in the `-coordinate` option, the coarse placement engine can place cells in any of the rectangles. These placeable areas can overlap, or be disjoint. One application of this is to define a rectilinear move

`create_bounds`

bound by decomposing the original rectilinear move bound into a number of rectangle bounds, and list each rectangle bound in the -coordinate list. The number of parameters in the list must be a multiple of 4.

Placement bounds can alternatively be defined in an input PDEF file. These are hard bounds and will be honored by coarse placement. However, you will not be able to view/remove them by using report_bounds and remove_bounds commands. These two commands work only with the bounds created by the create_bounds command. Rectangular hard move bounds (where -coordinate has only 4 parameters) created by create_bounds can be written out to a PDEF file using the write_pdef command. There is currently no IEEE PDEF support for other types of bounds (soft or rectilinear). However, the bounds created by create_bounds will be persistent in the db. Therefore, if the db is saved and re-loaded, there is no need to re-create them.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example constrains the placement of the instance "INST_1" to lie within the rectangle with lower left corner (100 100) and upper right corner (200 200).

```
prompt> create_bounds -name "foo" -coordinate {100 100 200 200} INST_1
```

The following is an example to create a hard rectilinear move bound on a hierarchical instance "INST2". The rectilinear bound is created as a set of two rectangles [(10 10)(30 20)] & [(20 20)(30 30)] with a common edge.

```
prompt> create_bounds -name "foo2" -coordinate {10 10 30 20 20 20 30 30} -  
type hard INST_2
```

SEE ALSO

```
remove_bounds(2)  
report_bounds(2)  
update_bounds(2)  
create_placement(2)  
set_cell_location(2)  
set_dont_touch_placement(2)
```

create_buffer_tree

Creates a buffer tree for the specified driver pins and nets.

SYNTAX

```
status create_buffer_tree  
[-from pin_net_list]  
[-net_scope]  
[-no_legalize]
```

Data Types

pin_net_list collection

ARGUMENTS

-from *pin_net_list*

Specifies the driver pins and nets for which the tool should create buffer trees. When the **-from** argument is not specified, the **create_buffer_tree** command works on all drivers with a transitive fanout of 40 or more.

-net_scope

Sets the scope of buffer tree creation to the specified net, rather than the entire buffer tree driven by the specified net. This option is ignored if **set_ahfs_options -optimize_buffer_trees** is set to false.

When this options is specified **create_buffer_tree** constructs a buffer tree, as needed, on each net specified in the **-from** option to reduce the high-fanout of each net, but does not remove any pre-existing buffers or inverters. Thus the "scope" of the construction is a single specified net.

When this option is not specified, **create_buffer_tree** can remove pre-existing buffers and inverters in the fanout of each specified net to improve QoR.

-no_legalize

Disables placement legalization at the end of buffering.

DESCRIPTION

The **create_buffer_tree** command creates a buffer tree for each specified driver pin and for the driver pin of each specified net.

The **create_buffer_tree** command generates a hierarchical buffer tree (buffers are inserted across hierarchical boundaries). The command is location-based. Therefore, the specified driver pin should not be a hierarchical pin, because a hierarchical pin does not have a location.

This command works *only* on a placed design, and it requires the presence of the following libraries and information:

- The physical libraries that correspond to the specified logical libraries.

- The capacitance and resistance per unit length, derived from the physical library or specified by you.

Multicorner-Multimode Support

This command uses information from active scenarios only.

EXAMPLES

The following example creates a buffer tree that is implemented at driver a/0.

```
prompt> create_buffer_tree -from [get_pins a/0]
```

The following example creates buffer trees for driver a/0 and net n1:

```
prompt> create_buffer_tree -from {[get_pins a/0] [get_nets n1]}
```

SEE ALSO

```
remove_buffer_tree(2)  
report_buffer_tree(2)  
report_ahfs_options(2)  
report_cbt_options(2)  
report_net_fanout(2)  
set_ahfs_options(2)  
set_cbt_options(2)
```

create_cell

Creates cells in the current design or its subdesigns.

SYNTAX

```
int create_cell  
cell_list  
reference_name  
[-view view_name]  
[-freeze_silicon]
```

Data Types

<i>cell_list</i>	list
<i>reference_name</i>	string
<i>view_name</i>	string

ARGUMENTS

cell_list
Specifies the names of cells created in the current design. Each cell name must be unique within the current design.

reference_name
Specifies the design or library cell that new cells reference. Ports on the reference determine the name, number, and direction of pins on the new cell.

-view *view_name*
Specifies the Milkyway view of the reference design used for creating the new cell. Using this option can increase runtime, because it triggers a link of the design.
When creating the cell, the tool verifies that the specified view exists for the reference cell.
Do not use this option for creating standard cells, which always use the FRAM view of the reference cell.
Note that all cells in the design must have a FRAM view in the reference library, regardless of the view used when creating a new cell. The **create_cell** command currently verifies only the existence of the specified view. If you specify **-view CEL** and a FRAM view does not exist for the reference design, you might get errors later in the design flow. To create a FRAM view for a reference design, use the **create_macro_fram** command.

-freeze_silicon
Supports the ECO freeze silicon flow. The cell instances created with this option are in transitional state. The **place_freeze_silicon** or **map_freeze_silicon** command will delete them and replace them with existing spare cells of the same reference library cell.

DESCRIPTION

This command creates new cells in the current design or its subdesigns based on the

cell_list argument. New cells are the instantiation of an existing design or library cell.

The created cells are unplaced. To be viewed properly in the GUI, the cells must be placed either manually by using the **set_cell_location** command or automatically by using placement commands.

To remove cells from the current design, use the **remove_cell** command.

Although the *reference_name* argument accepts names in the format *library/library_cell*, it does not imply that the actual library cell used for the new cell will be from the specified library. The actual library cell used is determined by the current link library settings.

EXAMPLES

The following example creates a cell named *cell1* under the subdesign corresponding to the *mid1* cell.

```
prompt> create_cell {mid1/cell1} my_lib/AND2
Creating cell 'cell1' in design 'mid'.
```

The following example creates cells using library cells as references.

```
prompt> create_cell {U3 U4} my_lib/NAND2
prompt> create_cell "U5" my_lib/NOR2
```

In IC Compiler, reference cells in the physical library can be used without the need for a logical library reference if the physical cell reference has only power and ground pins. This allows you to create some cells (such as filler, power pad, tap, and cap cells) without having logical library references for them. The following example creates a power pad using a physical library reference. No slash is used to designate a library, since there is only one physical library.

```
prompt> create_cell my_power_pad pvdd
```

The following example creates a FRAM view before using the **-view** option.

```
prompt> create_macro_fram -library_name logo -cell_name cell_logo \
-identify_macro_pin_by_pin_text
prompt> close_mw_library logo
prompt> open_mw_library design
prompt> open_mw_cell routed
prompt> create_cell -view FRAM cell_logo my_logo
```

SEE ALSO

```
remove_cell(2)
report_cell(2)
set_cell_location(2)
```

```
create_macro_fram(2)
place_freeze_silicon(2)
map_freeze_silicon(2)
```

create_clock

Creates a clock object and defines its waveform in the current design.

SYNTAX

```
status create_clock  
[-name clock_name]  
[-add]  
[source_objects]  
[-period period_value]  
[-waveform edge_list]
```

Data Types

<i>clock_name</i>	string
<i>source_objects</i>	list
<i>period_value</i>	float
<i>edge_list</i>	list

ARGUMENTS

-name *clock_name*

Specifies the name of the clock being created. If you do not use this option, the clock is given the same name as the first clock source specified in *source_objects*. If you do not use *source_objects*, you must use this option, which creates a virtual clock not associated with a port or pin. Use this option along with *source_objects* to give the clock a more descriptive name than that of the pin or port where it is applied.

If you specify the **-add** option, you must use the **-name** option and the clocks with the same source must have different names.

-add

Specifies whether to add this clock to the existing clock or to overwrite the existing clock. Use this option to capture the case where multiple clocks must be specified on the same source for simultaneous analysis with different clock waveforms. When you specify this option, you must also use the **-name** option. Defining multiple clocks on the same source pin or port causes longer runtime and higher memory usage than a single clock, because the synthesis timing engine must explore all possible combinations of launch and capture clocks. Use the **set_false_path** command to disable unwanted clock combinations. This option is ignored (the default), unless multiple clocks analysis is enabled by setting the **timing_enable_multiple_clocks_per_reg** variable to **true**.

source_objects

Specifies a list of pins or ports on which to apply this clock. If you do not use this option, you must use **-name** *clock_name*, which creates a virtual clock not associated with a port or pin. If you specify a clock on a pin that already has a clock, the new clock replaces the old clock unless you use the **-add** option.

-period *period_value*
Specifies the period of the clock waveform in library time units.

-waveform *edge_list*
Specifies the rise and fall edge times, in library time units, of the clock over an entire clock period. The first time in the list is a rising transition, typically the first rising transition after time zero. There must be an even number of increasing times, and they are assumed to be alternating rise and fall times. The numbers must represent one full clock period. If **-waveform** *edge_list* is not specified, but **-period** *period_value* is, a default waveform with a rise edge of 0.0 and a fall edge of *period_value*/2 is assumed.

DESCRIPTION

The **create_clock** command creates a clock object in the current design. The command defines the specified *source_objects* as clock sources in the current design. A pin or port can be a source for a single clock. If *source_objects* is not specified, but a *clock_name* is given, a virtual clock is created. A virtual clock can be created to represent an off-chip clock for input or output delay specification. For more information about input and output delay, refer to the **set_input_delay** and **set_output_delay** command man pages.

Clock objects hold attributes that affect the clock network, such as **dont_touch_network**, **fix_hold**, and **propagated_clock**. Using **create_clock** on an existing clock object overwrites the attributes previously set on the clock object. The **create_clock** command also defines the waveform for the clock. The clock can have multiple pulses per period. Setup and hold path delays are automatically derived from the clock waveforms of the path startpoint and endpoint. The **fix_hold** attribute (set by the **set_fix_hold** command) directs **compile** to fix hold violations for a clock.

By default, a new path group is created for the clock. This groups together the endpoints related to this clock for cost function calculation. To remove the clock from its assigned group, use the **group_path** command to reassign the clock to another group or to the default path group. For more information, refer to the **group_path** command man page.

The new clock has ideal timing, so no propagation delay through the clock network is assumed. To enable propagation delay through the clock network, use the **set_propagated_clock** command. To add skew or uncertainty to the ideal waveform, use the **set_clock_latency** or **set_clock_uncertainty** command.

To show information about all clock sources in a design, use the **report_clock** command. To get a list of clock sources, use the **get_clocks** command. To return sequential cells related to a given clock, use the **all_registers** command. To undo **create_clock**, use the **remove_clock** command.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example creates a clock on port PHI1 with a period of 10.0, rise at 5.0, and fall at 9.5:

```
prompt> create_clock "PHI1" -period 10 -waveform {5.0 9.5}
```

In the following example, the clock has a falling edge at 5 and a rising edge at 10, with a period of 10. Because the **-waveform** option expects the edges to be ordered first rise then fall, and to increase in value, the fall edge can be given as 15; that is, the next falling edge after the first rise edge at 10.

```
prompt> create_clock "PHI2" -period 10 -waveform {10 15}
```

The following example creates a clock named CLK on pin u13/Z, with a period of 25, fall at 0.0, rise at 5.0, fall at 10.0, rise at 15.0, and so forth:

```
prompt> create_clock "u13/Z" -name "CLK" -period 25 -waveform {5 10 15 25}
```

The following example creates a virtual clock PHI2 with a period of 10.0, rise at 0.0, and fall at 5.0:

```
prompt> create_clock -name "PHI2" -period 10 -waveform {0.0 5.0}
```

The following example creates a clock with multiple sources and a complex waveform:

```
prompt> create_clock -name "clk2" -period 10 -waveform {0.0 2.0 4.0 6.0} \  
          {clkgen1/Z clkgen2/Z clkgen3/Z}
```

SEE ALSO

- all_clocks(2)
- all_registers(2)
- check_timing(2)
- current_design(2)
- get_clocks(2)
- group_path(2)
- remove_clock(2)
- report_clock(2)
- reset_design(2)
- set_clock_latency(2)
- set_clock_uncertainty(2)
- set_dont_touch_network(2)
- set_fix_hold(2)
- set_max_delay(2)
- set_min_delay(2)
- set_output_delay(2)
- set_propagated_clock(2)

create_clock_mesh

Creates a grid of horizontal and vertical straps that are joined by vias for clock mesh implementation.

SYNTAX

```
status create_clock_mesh
[-net net]
[-layers hor_ver_list]
[-num_straps hor_ver_list | pitches hor_ver_list
  | -max_pitches hor_ver_list
  | -relative_pitches hor_ver_list]
[-widths hor_ver_list
  | -relative_widths hor_ver_list]
[-keepouts list_of_points]
[-ring]
[-bounding_box rectangle]
[-lower_left hor_ver_list]
[-upper_right hor_ver_list]
[-max_displacement hor_ver_list]
[-load net_or_pins]
[-avoid instances]
[-no_snap]
[-check_only]
[-remove_display]
[-verbose]
[-offset hor_ver_list]
[-full_pitch_perimeter]
[-pitches ]
```

Data Types

<i>net</i>	a net
<i>hor_ver_list</i>	a Tcl list where the first element specifies the horizontal value and the second element specifies the vertical value. The unit for the coordinates and distance values is the library distance unit.
<i>list_of_points</i>	a list of lower left and upper right points
<i>net_or_pins</i>	a collection of one net or multiple pins

ARGUMENTS

-net *net*
Connects new straps to the existing net that you specify. Use **create_net** if necessary.

-layers *hor_ver_list*
Specifies the horizontal and vertical layers to use for the mesh.

-num_straps *hor_ver_list* | pitches *hor_ver_list*
Specifies the number of horizontal and vertical straps to be created. Both numbers should be positive integers. A mesh created with **-num_straps {2 2}** has an outer ring if only **-ring** is specified or a shape of a pound sign (#)

create_clock_mesh

if **-ring** is not specified. High strap numbers result in a dense mesh. If you do not specify the **-pitches** or a **-num_straps** option, the command will synthesize a mesh based on internal experimental values.

-max_pitches *hor_ver_list*

Specifies the maximum horizontal and vertical pitches of the mesh. The first distance is the maximum vertical distance between horizontal straps, and the second is the horizontal distance between vertical straps. This option is mutually exclusive with **-num_straps**. This is a soft constraint; that is, if the pitch does not divide the effective bounding box evenly, additional straps will be added to allow the straps to be evenly distributed. You might want to use the electrical or physical limits as a base for the settings of this option.

-relative_pitches *hor_ver_list*

Specifies the relative horizontal and vertical pitches of the mesh. The pitches are represented as multiples of the default minimum wire spacing for the chosen layers. The first distance controls the minimum vertical distance between horizontal straps, and the second controls the horizontal distance between vertical straps. This option is mutually exclusive with **-num_straps**. This is a soft constraint; that is, if the pitch does not divide the effective bounding box evenly, additional straps will be added to allow the straps to be evenly distributed. You might want to use the electrical or physical limits as a base for the settings of this option.

-widths *hor_ver_list*

Specifies the widths of horizontal and vertical straps. If you do not specify this option, the command will select widths based on internal experimental values.

-relative_widths *hor_ver_list*

Specifies the widths of horizontal and vertical straps that are expressed as multiples of the default minimum wire width for the specified layers.

-keepouts *list_of_points*

Specifies a list of explicit locations of placement blockages or cells rather than design objects. The tool does not generate straps and vias that intersect these blocks.

-ring

Encloses the mesh with a rectangular ring that connects the ends of horizontal and vertical straps. This adds routing overhead, capacitance, and power, but the ring might be more effective for reducing impedance among drivers and loads near the perimeter of the mesh. Note that the ring cannot be completed if the mesh is blocked.

-bounding_box *rectangle*

Specifies the bounding box for the mesh. If both **-load** and **-bounding_box** are specified, the **-bounding_box** option takes precedence.

-lower_left *hor_ver_list*

Specifies the location of straps precisely. The specified list consists the X location of the leftmost vertical strap and the Y location of the lowest horizontal strap. Use this option to specify the point where the lowest horizontal strap cross the leftmost vertical strap. The horizontal straps

will be limited on the left by the left edge of the (implied or specified) bounding box. Similarly, the vertical straps is limited on the bottom edge of the bounding_box.

-upper_right hor_ver_list

Specifies the point where the highest horizontal strap cross the rightmost vertical strap. This option can only be used with **-lower_left**. No vertical strap is allowed to the right of *ur_xloc*, and no horizontal strap is allowed above *ur_yloc*.

-max_displacement hor_ver_list

Sets the maximum allowed horizontal displacement of vertical straps and the maximum allowed vertical displacement of horizontal straps. Normally, the command places straps at the exact geometric location according to the specified options, but it will displace straps within the maximum allowable values to avoid obstructions. The default value of displacement is determined by internal experimental values, which can be overridden by **-max_displacement**. A large *max_hor* allows vertical straps to move great distances horizontally; a zero value eliminates any vertical straps that encounter a obstruction. See also the **-no_snap** option.

-load net_or_pins

If specified, use the bounding box of the specified pins as the bounding box for the mesh. If a net is present, then all pins on that net are considered.

-avoid instances

Specifies a collection of instances, typically large blocks. straps and vias that intersect these blocks will not be generated.

-no_snap

Specifies to turn the snap off. Normally, the command will adjust the position and spacing of the straps to coincide with the available routing tracks. If you do not want to do this, use the **-no_snap** option. The **-no_snap** option is useful early in the design process to help verify the **-bounding_box**, **-offset**, or other options. The option allows you to see where the straps are placed based on these options directly.

-check_only

Specifies not to actually insert the straps. Use the option to report the number of straps that would be created and their properties. If GUI is awake, it will display the tentative horizontal and vertical straps in the GUI after removing any previously annotated tentative straps.

-remove_display

Removes the tentative mesh straps previously annotated by the **-check_only** option in the GUI. It operates only if the GUI is awake.

-verbose

Specifies to generate messages during the process of creating straps. This option is off by default. Use **-verbose** if you want to see these messages.

-offset hor_ver_list

Moves the newly created vertical straps to the right by the specified horizontal value, and the vertical straps up by the specified vertical value. Use this option to line up with power and ground straps, but avoid

intersecting.

-full_pitch_perimeter

Adjusts spacing so that the four straps on the right, left, top, and bottom sides are away from the perimeter by a full pitch of a grid. The default is half a pitch.

-pitches

Specifies the horizontal and vertical pitches of the mesh. The first distance is the vertical distance between horizontal straps, and the second is the horizontal distance between vertical straps. This option is mutually exclusive with **-num_straps**. This is a hard constraint.

DESCRIPTION

This command builds a grid of horizontal and vertical straps suitable for a clock mesh. It is one of the first steps in clock mesh tree creation. Alternatively, you could create such a grid with a series of **create_net_shape** and **create_via** commands, but this **create_clock_mesh** command is specifically designed for the task.

The mesh will cover an rectangular area which may be specified directly with the **-bounding_box** option, or indirectly with the **-load** option. If **-load** is used, the mesh will be just big enough to cover all of the pins specified in the load. Typically, it is easiest just to specify the ideal clock net as the **-load** option. If neither **-bounding_box** nor **-load** is specified, the command will chose an area based on the site arrays in the current design.

The layers of the straps are specified as a list with the **-layers** option, with the first element in the list being the horizontal layer and the second being the layer to used for vertical straps. The widths may be omitted, in which case heuristic width straps are used. If **-widths** are used, the first element is the width for horizontal straps and the second for vertical straps. Alternatively, widths may be specified relative to the minimum width for the layer with the **-relative_widths** options.

Note that for the options that accept lists, the order of the elements in the list is important. E.g. when the first element in the **-layers** list is the horizontal layer (e.g. "M7"), it corresponds to the first element in the **-num_straps** options (e.g. 7): thus generating 7 horizontal straps on layer M7. If the second element in the **-layers** list is "M8" and the second element in the **-widths** option is 2.0, then straps on layer M8 will have a width of 2.0 microns.

The mesh will cover the entire rectangle defined by the bounding box unless **-avoid** or **-keepouts** is used to specify one or more blocks where the mesh should not be created. Use this option with care: it is possible to construct meshes with poor electrical connectivity, or even disconnected straps, using this option.

If you want a ring around the perimeter of your mesh, use the **-ring** option. A mesh created with the **-ring** option,

```
-ring -num_straps {2 2}
```

will look like a square. A mesh created without the **-ring** option,

```
-num_straps {2 2}
```

will look like a pound side ("#").

The number of straps can be any non-negative number, so

```
-num_straps {1 1}
```

will produce two straps shaped like a plus sign "+". And

```
-num_straps {8 0}
```

will add 8 horizontal straps and no vertical straps. Note that when 0 straps are specified in one dimension, the calculations are done internally as if two or more straps were specified in the other dimension, and then those straps are suppressed. This is important because the length of the horizontal straps, for example, is dependent on the vertical straps. For example, a command with

```
-num_straps {2 0}
```

will place two horizontal straps directly on top of the two horizontal straps created by

```
-num_straps {2 7}
```

This was done to facilitate producing complex meshes by running **create_clock_mesh** multiple times.

By default, the spacings between straps are equal in the interior of the mesh, but only half as wide around the perimeter. This is done so that each horizontal/vertical crossing is surrounded by the same area.

If you want evenly spaced straps, use the **-full_pitch_perimeter** option. For example, if a bounding box were {{0 0} {120.0 240.0}} and the options

```
-num_straps {2 4} -full_pitch_perimeter
```

were used to generate 2 horizontal straps and 4 vertical straps, the vertical straps would be 48 microns apart and 48 microns from the right and left sides. (240 divided by 5). Without the **-full_pitch_perimeter** option, the interior pitch would be adjusted to 60 microns, and the leftmost mesh strap is placed 30 microns (one half pitch) away from the bounding box, and the rightmost mesh strap would be 30 microns (one half pitch) away from the right side of the bounding box. Similar calculations would apply for the top and bottom straps. This style is preferred if you intend to place a driver at each interior intersection (in this case, 8 drivers), because then each driver is in the center of a rectangle with the same dimensions, and all drivers would serve similar loads.

If you know exactly where you want to place the wires, you may want to use the **-lower_left** and **-pitch** options. The argument to the **-lower_left** option is the exact position of the center of the crossing of the leftmost vertical wire with the bottommost horizontal wire. Using the **-pitch** option lets you specify the precise distance between wires. If you do this, you cannot specify the **-num_straps** option: the number of straps will be determined for you. But if you want fewer straps, you can limit the grid by specifying the **-upper_right** option. No vertical wires will be

added to the right of this point, nor any horizontal wires above this point.

Thus

```
create_clock_mesh -layer {M5 M6} -lower_left {15.125 76.015} -pitches {100 200}
```

would create vertical wires with X coordinates 15.125, 215.125, 415.125, etc and horizontal wires with Y coordinates of 76.015, 176.015, 276.015, etc.

A comment on the terminology used to define nets (relevant to all the clock mesh commands):

The commands that deal with clock meshes distinguish among three types of important nets. The term "clock net" refers to a net normally associated with the original ideal clock. Typically it is connected to an input port, or it may be marked in SDC as a derived clock. Generally, a clock starts out as an ideal net connected to a port or a clock driver and many loads.

The "root net" is the net at the apex of the tree of buffers that drive the mesh. The "root net" is often the same net as the "clock net", but need not be. If several clocks are distributed by the same mesh, the root net will be a common point for their distribution, but the root net need not be uniquely associated with one clock.

The "mesh net" is the net of the wiring grid. In the methodology illustrated here, the mesh net is created using **create_net** before creating the mesh itself.

Because each of these nets is important in understanding and analyzing the circuit, it is advisable to assign them evocative names. If there is only one clock involved, perhaps called "clk" connected to a port called "clk", then the net associated with the mesh might be called "clk_mesh." and the root net is also net "clk." The **create_clock_mesh** command could use **-load** clk and **-net** clk_mesh. Later, the **add_clock_drivers** command would take the pins that are on the net called clk, disconnect them from net "clk" and connect them to net clk_mesh, and add drivers that start from the net clk and drive clk_mesh. If you chose to use the **analyze_subcircuit** command, it would model the circuit starting from net "clk" through all the pins connected to net "clk_mesh".

If there were two distinct clocks, "clk1" and "clk2" that are gated together into the root of the tree, the root might be called "clk1_clk2_root", and the mesh that is driven might be "clk1_clk2_mesh". The **create_clock_mesh** command could use **-load** clk1_clk2_root and **-net** clk1_clk2_mesh. Later, the **add_clock_drivers** command would take the pins that are on the net called clk1_clk2_root, disconnect them from clk1_clk2_root and connect them to net clk1_clk2_mesh, and add drivers that start from the net clk1_clk2_root and drive clk1_clk2_mesh. The **add_clock_drivers** command would not deal with the nets "clk1" or "clk2". The **analyze_subcircuit** command would model the circuit starting from net "clk1_clk2_root" through all the pins connected to net "clk1_clk2_mesh". The **analyze_subcircuit** command would not begin its analysis at net clk1 or clk2, because **analyze_subcircuit** requires a single net at its root.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

This example creates a grid with horizontal straps on M7 and vertical straps on M8. It will include the top and bottom straps that form a ring around the mesh, and it connects the load pins to the net of the mesh, which is called "clk1_mesh". The command will enter just enough horizontal straps so that the distance between them will not be more than 100 times the minimum pitch for layer M7. It will add just enough vertical straps so that the pitch between them will not be more than 150 times the minimum pitch for layer M8. Since the widths of the straps are not specified, the command will choose the widths according to its internal experimental values.

```
prompt> create_net clk1_mesh
1
prompt> set_attribute [get_net clk1_mesh] net_type "Clock"
1
prompt> create_clock_mesh -net [get_net clk1_mesh] \
    -load [get_pins -of clk] -ring \
    -relative_pitches {100 150} -layers {M7 M8}
1
```

SEE ALSO

```
add_clock_drivers(2)
route_mesh_net(2)
compile_premesh_tree(2)
analyze_subcircuit(2)
compile_clock_tree(2)
```

create_command_group

Creates a new command group.

SYNTAX

```
string create_command_group  
group_name
```

Data Types

```
group_name      string
```

ARGUMENTS

```
group_name  
    Specifies the name of the new group.
```

DESCRIPTION

The **create_command_group** command is used to create a new command group, which you can use to separate related user-defined procedures into functional units for the online help facility. When a procedure is created, it is placed in the "Procedures" command group. With the **define_proc_attributes** command, you can move the procedure into the group you created.

group_name can contain any characters, including spaces, as long as it is appropriately quoted. If *group_name* already exists, **create_command_group** quietly ignores the command. The result of **create_command_group** is always an empty string.

EXAMPLES

The following example demonstrates the use of the **create_command_group** command.

```
prompt> create_command_group {My Procedures}  
prompt> proc plus {a b} {return [expr $a + $b]}  
prompt> define_proc_attributes plus -command_group "My Procedures"  
prompt> help  
My Procedures:  
    plus  
  
...
```

SEE ALSO

```
define_proc_attributes(2)  
help(2)
```

create_connview

Automatically invoke Hercules connectivity engine to generate connectivity (CONN) view and current source files (CSF) for nonstandard cells. The generated CONN view and CSF files can be used during IR drop analysis.

SYNTAX

```
status create_connview
-library library_name
-design design_name
[-power_nets net_names]
[-ground_nets net_names]
[-generate_csf]
[-skip_via mask_names]
[-connview_skip_cell cell_names]
[-csf_skip_cell cell_names]
[-layer_text mask_names_text]
```

Data Types

<i>library_name</i>	string
<i>design_name</i>	string
<i>net_names</i>	string
<i>mask_names</i>	string
<i>cell_names</i>	string
<i>mask_names_text</i>	string

ARGUMENTS

-library *library_name*
Specify the name of the library. The library is the one in MilkyWay database. This is a required option. There is no default value for this option.

-design *design_name*
Specify the single cell name in the specified Milkyway library for which you want to generate CONN view. This is a required option. There is no default value for this option.

-power_nets *net_names*
Specify names of the top-level power nets to be extracted. At least one of the following options: -power_nets, or -ground_nets is needed.

-ground_nets *net_names*
Specify names of the top-level ground nets to be extracted. At least one of the following options: -power_nets, or -ground_nets is needed.

-generate_csf
Specify to generate current source files (CSF) and then attach to CONN view during CONN view generation. The files will be used during rail analysis functions such as ICC-DP PNA or PrimeRail/AstroRail to improve accuracy by placing current sinks of hard macros into the locations described in the CSF file. By default, the option is off, meaning not generating CSF files.

`-skip_via mask_names`
Specify the mask names of vias for extracting the connectivity not through the specified vias. If the option is not specified, the command will extract connectivity through via1 to via11 (metal1 to metal12).

`-connview_skip_cell cell_names`
Specify the child cells to be excluded from the CONN view generation. By default, all the child cells will be included.

`-csf_skip_cell cell_names`
Specify the child cells to be excluded from the current source file generation. By default, all the child cells will be included.

`-layer_text mask_names_text`
Specify metal mask names with name or number of the layers used for text identifying the corresponding metal layer geometries. If all text layers match the geometry layers, then the option is not needed. The format of the layer text is as follows.
mask_name:layer_name_or_number
For example,
`-layer_text {metal2:32 metal3:31}`
Use layer 32 to identify metal2 geometry, and layer 31 for metal3.

DESCRIPTION

Non-standard cells, such as hard macros, can have significant effects on voltage drop and electromigration at the full-chip level. The power consumption of these cells or the power and ground routing that exists inside these cells could allow voltage to flow through the cells to other regions of the chip. It is therefore necessary to be able to consider the internal power and ground network and power consumption of these types of cells during IR drop analysis. The command combines the values of the routing and the power usage into a single construct that rail analysis tools such as ICC-DP PNA or PrimeRail can access. The construct consists of the following two parts: Connectivity (CONN) view: The CONN view represents the power and ground networks inside the cell in question. Current source file (CSF): The CSF file is an ASCII file that records where current (and hence power) is drawn inside the cell. This command automatically invoke Hercules connectivity engine to generate connectivity (CONN) view and current source files (CSF) for nonstandard cells.

Note: You should close the library and cell of interest before running this command. The number of cells printed in each Hercules command is limited to 300 to prevent the Hercules parser from breaking.

Multicorner-Multimode Support

This command is not supported in a multi-scenario design flow.

EXAMPLES

The following example shows how to generate a CONN view and CSF files with library `ts1ge256x16m4_lib`, cell view named `TS1GE256X16M4`, power net `VDD` and ground net `VSS`.

```
prompt> create_connview -library ts1ge256x16m4_lib -design TS1GE256X16M4 -  
power_nets VDD -ground_nets VSS -generate_csf
```

The following example shows how to generate a CONN view and CSF files with library ts1ge256x16m4_lib, cell view named TS1GE256X16M4, power net VDD, skip vias via2 and via3, skip cells S1B100W10_V4 and S1B100W10_V1_5 during CONN view generation, and use layer 5 as metal1 and layer 12 as metal2.

```
prompt> create_connview -library ts1ge256x16m4_lib -design TS1GE256X16M4 -  
power_nets VDD -skip_via {via2 via3} -  
connview_skip_cell {S1B100W10_V4 S1B100W10_V1_5} -layer_text {metal1:5 metal2:12}
```

SEE ALSO

```
analyze_fp_rail(2)  
synthesize_fp_rail(2)
```

create_custom_wire

Creates custom wires for a given collection of nets

SYNTAX

status

create_custom_wire

```
[-start_command | -add_point point | -delete_point | -end_points | -end_command | -undo ]
[-nets collection_of_nets]
[-horizontal_layer layer]
[-horizontal_width distance]
[-horizontal_pitch distance]
[-vertical_layer layer]
[-vertical_width distance]
[-vertical_pitch distance]
[-set_of_layers list_of_layers_with_options]
[-override filename]
[-mark_as mark_type]
[-start_at start_type]
[-switch_end_at_turning bool]
[-skip_count int]
[-multi_via_size int]
[-extend_first bool]
[-extend_last bool]
[-snap_to_track bool]
[-look_inside_std_cell bool]
[-reverse_net_order bool]
[-draw_bbox_vias bool]
[-show_detailed_info bool]
[-honor_route_guides bool]
[-extend_first_to_boundaries bool]
[-extend_last_to_boundaries bool]
```

Data Types

<i>collection_of_nets</i>	collection
<i>mark_type</i>	string
<i>start_type</i>	string
<i>list_of_layers_with_options</i>	list
<i>layer</i>	string
<i>file_name</i>	string
<i>distance</i>	distance

ARGUMENTS

-start_command
Start the mouse interaction mode.

-add_point *point*
Adds mouse point to process

-delete_point
Delete the last point

-end_points
Commits point input. Undo operation becomes possible after it.

-end_command
Ends the command

-undo
Removes last committed wires and vias.

-nets *collection_of_nets*
collection of nets. Maximum nets allowed is 2

-horizontal_layer *layer*
Metal layer number or layer name to be given. Default is the first listed horizontal metal layer from tech file.

-horizontal_width *distance*
Width of the given metal layer from the tech file

-horizontal_pitch *distance*
Pitch of the given metal layer from the tech file

-vertical_layer *layer*
Metal layer number or layer name to be given. Default is the first listed vertical metal layer from tech file.

-vertical_width *distance*
Width of the given metal layer from the tech file

-vertical_pitch *distance*
Pitch of the given metal layer from the tech file

-mark_as *mark_type*
Mark created wires and vias as one of: strap ring (default)

-set_of_layers *list_of_layers_with_options*
The list of layers with options in the following format: {<layer> <horizontal|vertical> <width> <bool> list} It specifies the set of layers to create wires on. The parameter is a list of layer, horizontal or vertical flags, boolean flags, which tell to search targets on specified layer. The <layer> can be specified either with the layerNumber or maskName from the technology file.

-override *file_name*
It is used to over-ride the *width* and/or *pitch* specs specified by other options. An over-ride should be specified on one single line and has either one of the following formats: net_number width pitch (net_number1 netnumber2 ...) width pitch.

-start_at *start_type*
When creating wire start at the low or high end of the net. One of: low_end (default) high_end

-switch_end_at_turning *bool*
When turning switch ends. It's false by default.

-skip_count *int*
Specifies number of nets are skipped during the change of the direction

-multi_via_count *int*
Specifies the minimum number of via cuts. It's 1 by default.

-extend_first *bool*
Extend the wire from the first point entered. It's false by default.

-extend_last *bool*
Extent the last bit of bus. It's false by default.

-snap_to_track *bool*
Create wire on wire track. It's false by default.

-look_inside_std_cell *bool*
Create vias to connect wires to pins inside std cells. It's false by default.

-reverse_net_order *bool*
Reverse the order of the bit of bus. It's false by default.

-draw_bbox_vias *bool*
Draws only the bounding box of the via. It's true by default.

-show_detailed_info *bool*
Shows widths and increasing length number, dynamically for each nets. It's false by default.

-honor_route_guides *bool*
Consider variable route rule defined for the nets. It's false by default.

-extend_first_to_boundaries *bool*
Extend the first created wire and generate a pin at boundary. It's false by default.

-extend_last_to_boundaries *bool*
Extend the last created wire and generate a pin at boundary. It's false by default.

DESCRIPTION

Creates wires between specified points. Because it's interactive command, it requires GUI to be started with *gui_start* command. You can change the internal application of DRC rules by using the **set_preroute_drc_strategy** command.

NOTES

You can use the undo operation to remove segments with *create_custom_wire -undo*.

EXAMPLES

The following example creates a custom wire shape and then destroys it with undo.

```
prompt> gui_start
prompt> create_custom_wire -start_command
prompt> set_preroute_drc_strategy -drc_off -min_layer METAL -max_layer METAL5
prompt> create_custom_wire -add_point {100.00 100.100}
prompt> create_custom_wire -add_point {200.00 100.100}
prompt> create_custom_wire -end_points
prompt> create_custom_wire -undo
prompt> create_custom_wire -end_command
```

SEE ALSO

```
set_preroute_drc_strategy(2)
gui_start(2)
```

create_differential_group

Defines a differential group for nets.

SYNTAX

```
status create_differential_group
-group group_name
-nets {collection_of_nets}
```

Data Types

<i>group_name</i>	string
<i>collection_of_nets</i>	list

ARGUMENTS

```
-group group_name
    Enter the name of the differential group of cells that are being defined.

-nets {collection_of_nets}
    Enter the name(s) of the net(s) you want to define associated to a specified
    differential group
```

DESCRIPTION

Defines your differential groups, one group at a time. Each group can have two or more nets. After you define the differential groups, you can route them by using the `route_differential` command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example is to setup a differential group, "dgroup", associated to the net(s) with prefix of "net_diff".

```
prompt> create_differential_group -group dgroup -nets \
[get_nets net_diff*]
```

SEE ALSO

`route_differential(2)`

create_drc_error

Creates an error record for an error type in the default error type class.

SYNTAX

```
collection create_drc_error
-error_view mw_error_view
-type error_type
[-status error | fixed | ignored | waived]
[-info description]
[-details details]
```

Data Types

<i>mw_error_view</i>	list or collection of one item
<i>error_type</i>	string
<i>description</i>	string
<i>details</i>	list

ARGUMENTS

-error_view *mw_error_view*
Specifies the error view in which to create the error record. You may specify only a single error view; specifying more than one error view causes an error. This is a required option.

-type *error_type*
Specifies the error type for the new error object. You can specify the error type by using the id or name of the error type. Although the command accepts error types from any error type class, you should use this command only for error types in the default error type class. For error types in nondefault error type classes, you should use the class-specific error object creation commands. For more information, see the DESCRIPTION section. This is a required option.

-status error | fixed | ignored | waived
Sets the current status of this error.
If you do not specify this option, the status is set to **error**.

-info *description*
Provides a full description of this error.
If you do not specify this option, the description is left empty.

-details *details*
Specifies the error details, including shapes, nets, and layers.
To specify the error details, use the following syntax for the *details* argument:
details: {*numDetails detail*+}
detail: { [{*net net*}]
 num_shapes {*shape*+}}
shape: { [{*layer layer*}]
 num_rectangles {*rectangle*+}}

rectangle: {llx lly urx ury}

You can specify one or more details for an error. For each detail, you must specify one or more shapes that are associated with the error. Each shape is composed of one more rectangles and is optionally associated with a layer. You can also associate the shapes with a net.

This argument is optional; however, an error with no details is virtually useless. You can also add details by using the **add_drc_error_detail** command.

DESCRIPTION

This command creates a new error object of the specified error type. Every error object must be associated with one and only one error type. The error type association is made when the error object is created and cannot be changed. Therefore, you must create an error type before you can create error objects of that type.

When an error type is created with the **create_drc_error_type** command, it is associated with one of the following error type classes: default, open, openlocator, short, and spacing. An error object is interpreted based on its error type class.

In addition to the **create_drc_error** command, IC Compiler also provides the following error object creation commands, which are specific to the nondefault error type classes: **create_open_drc_error**, **create_open_locator_drc_error**, **create_short_drc_error**, and **create_spacing_drc_error**.

To ensure that the error objects are correct-by-construction, you should use the **create_drc_error** command only to create error objects with the default error type class. To create error objects with a nondefault error type class, use the appropriate class-specific error object creation command.

If successful, the **create_drc_error** command returns a collection containing the new error object.

EXAMPLES

The following example opens an error view named mydesign_mydrc.err, creates an error type named Overlap, and then creates an error record of the Overlap type, which is associated with one shape on the RV layer:

```
prompt> set cellId [open_mw_cel -not_as_current mydesign_mydrc.err]
{mydesign_mydrc}
prompt> set typeId [create_drc_error_type -name Overlap \
-info "Metal and blockage overlap" -error_view $cellId]
1024
prompt> create_drc_error -type $typeId -error_view $cellId \
-details {1 {1 {{layer RV} 1 {574.0700 430.3600 578.9700 433.4400}}}}
{1280}
```

SEE ALSO

add_drc_error_detail(2)
create_drc_error_type(2)
create_open_drc_error(2)

```
create_open_locator_drc_error(2)
create_short_drc_error(2)
create_spacing_drc_error(2)
get_drc_errors(2)
list_drc_error_types(2)
remove_drc_error(2)
report_drc_error_type(2)
```

create_drc_error_type

Creates an error type record. Each error object must be associated with one and only one error type record.

SYNTAX

```
integer create_drc_error_type  
-error_view mw_error_view  
-name type_name  
[-class default | open | openlocator | short | spacing]  
[-info description]  
[-status error | fixed | ignored | waived]  
[-level err_type_level]
```

Data Types

<i>mw_error_view</i>	list or collection
<i>type_name</i>	string
<i>description</i>	string
<i>err_type_level</i>	string

ARGUMENTS

-error_view *mw_error_view*

Specifies the error view in which to create the error type record. You may specify only one error view; specifying more than one error view causes an error.

This is a required option.

-name *type_name*

Specifies the name of the error type. The specified name must be unique in a case-insensitive comparison with existing names. The command fails if an error type with the specified name already exists.

This is a required option.

-class default | open | openlocator | short | spacing

Specifies the error class for the created error type. The class name keywords are case insensitive.

The open error type class is for violations where routed shapes for a net are not fully connected and is described as two or more disjoint nodes of connected net shapes.

The openlocator error type class is for violations that describe opens in nets, not as disjoint nodes of connected net shapes, but as the potential connection flyline for an open.

The short error type class is for violations where routed shapes overlap in the same area in a layer.

The spacing error type class is for violations where two routed shapes in the same layer violate a spacing rule.

If you specify this option, error objects of this type receive data interpretation specific to the error class.

If you do not specify this option or if you specify **default**, error objects of this type receive the default data interpretation.

-info *description*
 Provides a full description of the created error type. If you do not specify this option, the description is left empty.

-status error | fixed | ignored | waived
 Sets the default status for errors of the created error type.
 If you do not specify this option, the default status is **error**.

-level *err_type_level*
 Specifies the severity level of the created error type. The value must be one of **Error**, **"Critical Error"**, **"Major Error"**, **"Moderate Error"**, **"Minor Error"**, or **Recommendation**.
 If you do not specify this option, the severity level is **Error**.

DESCRIPTION

This command creates a new error type with the specified type name. Every error object must be associated with one and only one error type. The error type association is made when the error object is created and cannot be changed. Therefore, you must create an error type before you create error objects of that type.

If successful, the command returns the data object ID of the new error type record. Otherwise, the command returns 0. You can use the returned ID for the error type record in the **-type** option that identifies the type when creating error objects of that type. You can also identify the error type by using the type name specified as the argument to the **-name** option.

The IC Compiler error browser allows you to display error objects grouped by type. Error objects can also be sorted or filtered by type. Error types can be declared to be in a type class. Error objects of a type in a type class receive data interpretation that is specific to that type class. Nondefault type classes are: open, openlocator, short, and spacing. To create an error type in a nondefault type class, use the **-class** option. The error type association with a type class is made when the error type is created and cannot be changed.

EXAMPLES

The following example opens an error view named mydesign_mydrc.err and creates an error type named Overlap in the default type class:

```
prompt> set cellId [open_mw_cel -not_as_current mydesign_mydrc.err]
{mydesign_mydrc}
prompt> create_drc_error_type -name Overlap \
  -info "Metal and blockage overlap" -error_view $cellId
1024
```

The following example adds a type named Via Spacing that is in the spacing type class to an error view named mydesign_err.err:

```
prompt> create_drc_error_type -name "Via Spacing" -class "spacing" \
  -error_view "mydesign_err.err" \
  -info "Via corner spacing" -error_view $cellId
```

SEE ALSO

```
list_drc_error_types(2)
report_drc_error_type(2)
create_drc_error(2)
create_open_drc_error(2)
create_open_locator_drc_error(2)
create_short_drc_error(2)
create_spacing_drc_error(2)
get_drc_errors(2)
```

create_edit_group

Creates a new edit group from the specified objects.

SYNTAX

```
collection create_edit_group  
[-name string]  
objects
```

ARGUMENTS

-name string

Specifies the optional name of the edit group.

The name must be unique if specified. If you do not specify a name, the tool generates a unique name for the edit group.

objects

Specifies the collection of objects to be placed in the edit group.

Objects are limited to those objects that are movable and do not already belong to another edit group.

If any specified object is invalid, no edit group is created.

DESCRIPTION

Creates a new edit group from the specified objects.

If the command completes successfully, a collection containing the created edit group is returned.

EXAMPLES

The following example creates an edit group from the current selection.

```
prompt> set gr [create_edit_group [get_selection]]
```

SEE ALSO

```
get_edit_groups(2)  
remove_edit_groups(2)  
report_edit_groups(2)
```

create_fp_block_shielding

Creates signal shielding for plan groups and soft macros.

SYNTAX

```
status create_fp_block_shielding
[-inside_boundary]
[-outside_boundary]
[-side_list {left | right | top | bottom}]
[-metal_layers layer_list]
[-shielding_width factor_or_distance]
[-width_in_microns]
[-tie_to_net net]
[-block_level]
[objects]
```

Data Types

<i>layer_list</i>	collection
<i>factor_or_distance</i>	float
<i>net</i>	collection of one
<i>objects</i>	collection

ARGUMENTS

-inside_boundary

Creates shielding on the inside of plan group(s) and/or soft macro(s). If you do not specify the **-inside_boundary** and **-outside_boundary** options, the command creates shielding on both the interior and exterior sides of the plan group(s) and/or soft macro(s). By default, this option is off (false).

-outside_boundary

Creates shielding on the outside of the plan group(s) and/or soft macro(s). If you do not specify the **-inside_boundary** and **-outside_boundary** options, the command creates shielding on both the interior and exterior sides of the plan group(s) and/or soft macro(s). By default, this option is off (false).

-side_list {left | right | top | bottom}

Indicates the side(s) on which to create shielding rectangles. By default, the command creates shielding on all sides of the plan group(s) and/or soft macro(s). In the case of a rectilinear boundary, all boundary edges which are along the top of the boundary (face up) will be shielded with "top" rectangles, and those edges which are along the left side (and face left) will be shielded with "left" shielding rectangles, etc. etc..

-metal_layers *layer_list*

Specifies a layer object collection indicating the metal layer(s) on which to create shielding. By default, the command determines the appropriate layers for shielding on each side. See the explanation in DESCRIPTION for detailed information on automatic layer selection by the command.

-shielding_width *factor_or_distance*

Specifies the factor or distance by which the shielding width is multiplied. The shielding width is determined by a metal layer's pitch, unless you use the **-width_in_microns** option, in which case the command interprets the value of *factor_or_distance* as an absolute distance in microns. By default, the *factor_or_distance* value is **3.0**.

-width_in_microns

Interprets the value of the **-shielding_width** option as an absolute distance value in micron units. By default, this option is off (false).

-tie_to_net *net*

All shielding rectangles are tied logically to the single net represented by the net collection. This must be a single net (VSS, VSS1, GND, VDD, etc...) in the collection (i.e. what would be returned by [get_nets -all {VSS}]). By default, all shielding rectangles are logically floating.

-block_level

This option is for creating block shielding from the inside of a soft macro cell opened for edit. This will only create shielding around the inside of the cell boundary outline. No shielding will be created outside of the cell boundary outline. The options **-inside_boundary** and **-outside_boundary** are meaningless and disregarded in the context of the use of this option. Also, the *object_list* object collection is meaningless and disregarded when this option is specified. All other options may be used in combination with this option. By default, this option is off (false).

objects

Specifies a collection of plan group(s) and/or soft macro(s) on which to apply route blockages for signal integrity shielding. By default, the command processes all plan groups and soft macros.

DESCRIPTION

This command creates signal shielding for plan groups and soft macros. The shielding is used to maintain signal integrity when routing a design. Shielding takes the form of metal rectangles around the boundary of a block. These rectangles are regular route metal rectangles, subject to all metal layer rules. When shielding would block access to pins along a side edge on the same layer as the shielding, the shielding is segmented to allow access to the pins. You can specify a list of plan groups/soft macros for the command to process. By default, it processes all plan groups/soft macros.

The command can create shielding on the inside and/or outside of a plan group/soft macro. When the command creates inside shielding for soft macros, it is created at the child level. The command accepts a list of sides indicating on where to create shielding. By default, the command creates shielding on all sides, based on the rules provided below.

You can specify the metal layers on which to create shielding. By default, the command creates shielding on each metal layer and uses each layer's preferred direction to determine on which sides to create the shielding. For example, if metal 2's preferred direction is horizontal, the command creates metal 2 blockage rectangle blockages on the top and bottom sides of the plan group/soft macro. The

argument to this option is a collection of layer objects returned by the **get_layer** command (for example, **[get_layer {metal1 metal2}]**).

You can specify the width of the shielding to the command. By default, the command uses a width of 3 times the route pitch for each respective metal layer. The pitch is multiplied by the multiplier that you specify by using the **-shielding_width** option or by the default multiplier of **3**. You can flag the command to use an absolute width of microns instead of the multiplier by specifying the **-width_in_microns** option.

The rectangles the command creates are marked in the database, so that they can be recognized for deletion by the **remove_fp_block_shielding** command.

If it is desired for the shielding rectangles to be tied to a particular power or ground net, then the **-tie_to_net** option may be used to specify which power or ground net to use. This option may be used by passing a single net name or a net collection of 1, such as may be returned by the **[get_nets -all {VSS}]** command. This functionality only logically ties the rectangles to a particular power or ground net. It is beyond the scope of this option to physically connect to power or ground.

If it is desired to create shielding from inside of a soft macro opened in IC Compiler for edit, then the **-block_level** option may be called to create block shielding around the inside of the opened-for-edit soft macro's inside cell outline boundary.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates metal route blockages (shielding) on the top and left interior sides on metal layers 2, 6, and 7.

```
prompt> create_fp_block_shielding inside_boundary \  
-side_list {top left} \  
-metal_layers [get_layer {metal2 metal6 metal7}]
```

The following example creates exterior metal route blockages (shielding) of width 5 microns on the specified list of soft macro and plan groups. This example allows the command to determine on which metal layers and sides to create shielding.

```
prompt> create_fp_block_shielding -outside_boundary \  
-shielding_width 5.0 -width_in_microns [get_cells \  
{macro1 macro2 plan_group2 plan_group_8}]
```

This example creates shielding rectangles on inside and outside boundaries of all plan groups and soft macros and assigns them to the ground net VSS. This example allows the command to determine on which metal layers and sides to create shielding.

```
prompt> create_fp_block_shielding outside_boundary \  
-tie_to_net [get_nets -all {VSS}]
```

This example creates shielding rectangles on the inside of the cell boundary outline of the cell that is opened for edit. This example allows the command to determine on which metal layers and sides to create shielding.

```
prompt> create_fp_block_shielding-block_level
```

SEE ALSO

```
remove_fp_block_shielding(2)
```

create_fp_blockages_for_child_hardmacro

Creates routing blockages for hard macros within soft macros.

SYNTAX

```
status create_fp_blockages_for_child_hardmacro  
list_of_softmacros
```

Data Types

```
list_of_softmacros      collection
```

ARGUMENTS

```
list_of_softmacros  
    This is a collection of the soft macros which will be processed. If not  
    supplied all soft macros will be processed.
```

DESCRIPTION

For each soft macro processed each hard macro within that soft macro will have a placement blockage created in the top level. This placement blockage will have the same boundary as the hard macro.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

Create placement blockages for hard macros in all soft macros.

```
create_fp_blockages_for_child_hardmacro
```

create_fp_group_block_ring

Creates a block ring for a group of hard macro blocks.

SYNTAX

```
status create_fp_group_block_ring
-nets nets
[-horizontal_ring_layer layer]
[-horizontal_ring_offset offset]
[-horizontal_ring_spacing spacing]
[-horizontal_ring_width width]
[-vertical_ring_layer layer]
[-vertical_ring_offset offset]
[-vertical_ring_spacing spacing]
[-vertical_ring_width width]
[-horizontal_strap_width width]
[-horizontal_strap_layer layer]
[-vertical_strap_width width]
[-vertical_strap_layer layer]
[-output_directory directory]
[-skip_strap]
```

Data Types

<i>nets</i>	collection of list
<i>layer</i>	string
<i>offset</i>	float
<i>spacing</i>	float
<i>width</i>	float
<i>directory</i>	string

ARGUMENTS

-nets *nets*
Specify the power or ground nets to perform group block ring creation. This is a required option.

-horizontal_ring_layer *layer*
Specify the horizontal ring layer, where the default is the top-most horizontal layer.

-horizontal_ring_offset *offset*
Specify the horizontal ring offset in microns, where the offset is measured from the block boundary but not from the PNS region. The default is the minimum spacing between hard macro blocks and the ring defined in the technology file.

-horizontal_ring_spacing *spacing*
Specify the horizontal ring spacing in microns between power and ground nets. The default is minimum spacing defined in the technology file.

create_fp_group_block_ring

`-horizontal_ring_width width`
Specify the horizontal ring width in microns. The default is the minimum width defined in the technology file.

`-vertical_ring_layer layer`
Specify the vertical ring layer, where the default is the top-most vertical layer.

`-vertical_ring_offset offset`
Specify the vertical ring offset in microns, where the offset is measured from the block boundary but not from the PNS region. The default is the minimum spacing between hard macro blocks and the ring defined in the technology file.

`-vertical_ring_spacing spacing`
Specify the vertical ring spacing in microns between power and ground nets. The default is minimum spacing defined in the technology file.

`-vertical_ring_width width`
Specify the vertical ring width in microns. The default is the minimum width defined in the technology file.

`-horizontal_strap_width width`
Specify the horizontal strap width, where the default is the minimum width defined in the technology file. The straps are inserted inside the narrow regions between neighboring macro blocks.

`-horizontal_strap_layer layer`
Specify the horizontal strap layer, where the default is the top-most horizontal layer.

`-vertical_strap_width width`
Specify the vertical strap width, where the default is the minimum width defined in the technology file. The straps are inserted inside the narrow regions between neighboring macro blocks.

`-vertical_strap_layer layer`
Specify the vertical strap layer, where the default is the top-most vertical layer.

`-output_directory directory`
Specify the directory in which to store the group block ring creation results. The default is `./pna_output`.

`-skip_strap`
Specify not to insert straps between hard macro blocks; only rings are created around the group of macro blocks.

DESCRIPTION

This command creates power and ground rings around a set of hard macro blocks with either rectangular or rectilinear boundaries. Straps are also inserted between neighboring blocks unless the **-skip_strap** option is specified.

A PNS region must first be created by using the **set_fp_rail_region_constraints** command. This PNS region provides a guideline for the group block ring contour. For example, a rectilinear region is needed as a guideline if a rectilinear ring is desired for a group of hard macro blocks. The specified PNS region is first shrunk based on all hard macro blocks inside this region to obtain the group block boundary while maintaining the region topology. Next, a ring is created around the group block boundary.

Channels between neighboring blocks are identified for strap insertion. The straps are inserted in the middle of each channel unless it is too narrow.

Sandwich-type rings (multiple rings in one layer) are supported by specifying multiple nets, for example **-net {VDD VDD VSS}**. Straps in channels are still created as a pair of power and ground straps in this case.

You can preview the group block ring result before actually committing the group block ring by using the **commit_fp_group_block_ring** command.

Remove the PNS region constraint after group block ring creation and before power network synthesis.

EXAMPLES

The following example shows how to create power (VDD) ground (VSS) group block ring for those hard macro blocks inside the predefined PNS region. The rings are created on horizontal layer METAL5 with a ring width of 1.0 micron and an offset of 0.6 micron and on vertical layer METAL6 with a ring width of 1.0 micron and an offset of 0.6 microns. The spacing between horizontal (or vertical) power and ground rings is 0.5 micron. The straps between neighboring blocks are created on horizontal layer METAL3 with a strap width of 0.5 micron and on vertical layer METAL4 with a strap width of 0.5 micron.

```
prompt> create_fp_group_block_ring
-nets {VDD VSS}
-horizontal_ring_layer METAL5
-horizontal_ring_offset 0.6
-horizontal_ring_spacing 0.5
-horizontal_ring_width 1.0
-vertical_ring_layer METAL6
-vertical_ring_offset 0.6
-vertical_ring_spacing 0.5
-vertical_ring_width 1.0
-horizontal_strap_width 0.5
-horizontal_strap_layer METAL3
-vertical_strap_width 0.5
-vertical_strap_layer METAL4
```

SEE ALSO

```
set_fp_rail_region_constraints(2)
commit_fp_group_block_ring(2)
```

create_fp_pins

Creates pins for the specified list of child ports.

SYNTAX

```
pins create_fp_pins
[-side side_num_or_name]
-layer layer
[-width width]
[-step step_num]
{-at location | -offset offset}
child_ports
```

Data Types

<i>side_num_or_name</i>	string
<i>layer</i>	layer
<i>width</i>	real
<i>step_num</i>	integer
<i>location</i>	point
<i>child_ports</i>	collection

ARGUMENTS

-side *side_num_or_name*
Specifies a side number or name. If **-side** is not provided, the argument **-at** must be used and off-edge pins will be created at/or around the location specified by **-at**.

-layer *layer*
Specifies the layer name for a pin to be created.

-width *width*
Specifies the pin width of a pin to be created.

-step *step_num*
Specifies a wire track step for multiple pins to be created. The default is **1**.

-at *location*
Specifies location of the first pin to be created. For multiple off-edge pin creation, the first pin is located at the wiretrack intersection closest to the **-at** location, other pins are clockwise distributed around the first pin. The **-at** and **-offset** options are mutually exclusive; use only one.

-offset *offset*
Specifies the offset from the start of a side. The **-at** and **-offset** options are mutually exclusive; use only one.

child_ports
Specifies child ports for which to create pins. It is a collection of pins.

DESCRIPTION

This command creates pins for the specified list of child ports. Snapping is done automatically using global snap settings.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example create pins for all child ports of a soft macro named *u2_large* on a layer named *M1*.

```
prompt> set pins [get_pins -of [get_cell u2_large]]
prompt> set layer [get_layer M1]
prompt> create_fp_pins -side 1 -layer $layer -offset 10 $pins
```

SEE ALSO

`get_nets(2)`
`get_pins(2)`

create_fp_placement

Places hard macros and leaf cells.

SYNTAX

```
int create_fp_placement
[-effort low | high]
[-max_fanout positive_integer]
[-no_hierarchy_gravity]
[-no_legalize]
[-incremental placement_string]
[-congestion_driven]
[-timing_driven]
[-num_cpus number_of_cpus]
[-plan_groups collection_of_plan_groups]
[-voltage_areas collection_of_voltage_areas]
[-optimize_pins]
[-ignore_scan]
[-write_placement_blockages]
```

Data Types

<i>positive_integer</i>	integer
<i>placement_string</i>	string
<i>number_of_cpus</i>	integer
<i>collection_of_plan_groups</i>	collection
<i>collection_of_voltage_areas</i>	collection

ARGUMENTS

-effort low | high

Provides a trade-off between quality of results (QoR) and the amount of CPU runtime spent performing a flat placement. The valid values are **low** and **high**. For fast results that provide good, nonoverlapping hard macro locations, set the effort to **low**. For higher quality placement, but with an increase in CPU runtime, set the effort to High. The default value is **low**.

-max_fanout *positive_integer*

Specifies that placement ignore nets having a fanout higher than the specified number specified by *positive_integer*. In other words, the wire length of these nets are not considered in placement. The default value is **512**.

-no_hierarchy_gravity

Prevents the grouping together of cells of the same hierarchy block. The default is to group cells of the same hierarchy block together. In general, this type of grouping results in better placement because designs tend to partition well along the hierarchy boundaries. However, this option should be used if the designer knows that the hierarchy is not a good partitioning of the design.

-no_legalize
Creates a placement that is not legalized. The default is to legalize it.

-incremental *placement_string*
Performs incremental placement from current placement. The valid values are **all**, **top_level_cells**, **plan_groups**, and **voltage_areas**. Use **all** to perform an incremental placement on the entire design. Use **top_level_cells** to run an incremental placement on top level cells that do not belong to any plan group or voltage area. Use **plan_groups** together with the **-plan_groups** option to perform incremental placement to the cells within the specified plan groups. Use **voltage_areas** together with the **-voltage_areas** option to perform incremental placement to cells within the specified voltage areas. The default is for the command to run placement from scratch.

-congestion_driven
Enables congestion-driven placement mode. By default, the command runs with congestion-driven placement mode off.

-timing_driven
Enables timing-driven placement mode. By default, the command runs with timing-driven placement mode off.

-num_cpus *number_of_cpus*
Specifies the number of CPUs used in parallel during coarse placement. The *number_of_cpus* is an integer value that is less than or equal to the number of free CPUs on your machine and greater than or equal to 1. By default, the command uses 1 CPU. Only one Galaxy-PSYN License is checked out.

-plan_groups *collection_of_plan_groups*
Performs refine placement for the specified plan groups. This option is valid only for plan groups placed inside the core area and when you specify the **-incremental plan_groups** option.

-voltage_areas *collection_of_voltage_areas*
Performs refine placement for the specified voltage areas. This option is valid only for voltage areas placed inside the core area and when you specify the **-incremental voltage_areas** option.

-optimize_pins
Does simultaneous placement and pin assignment for block level designs. The pin assignment done here only supports pin physical constraints such as side, location and order.

-ignore_scan
Ignores scan chain connections during placement.

-write_placement_blockages
All blockages computed internally during placement will be written to a file called *design_planning_blockages.tcl* in the working directory. This includes blockages due to channels that are narrower than the sliver size and macro padding blockages. If the *design_planning_blockages.tcl* file already exists in the working directory, it will be overwritten.

DESCRIPTION

This command performs a virtual flat placement to obtain a placement of standard cells and hard macros. It provides you with an initial placement for creating a floorplan to determine the relative locations and shapes of the top-level physical blocks of a flat design with no placed plan groups or voltage areas. For designs with plan groups and/or voltage areas placed, it provides you with a refined placement that honors the exclusiveness of plan groups and/or voltage areas.

When you use the **-optimize_pins** option, this command results in an virtual flat placement result as described above and also give a pin assignment respecting the pin constraints. In this mode, the placer understands pin constraints such as side constraints; and thus, the overall quality of placement and pin assignment is better than separately running placement and pin assignment. Use this option at the block level, not at the chip level. It does not work with I/O pads.

Multicorner-Multimode Support

This command uses information from active scenarios only.

Support Relative Placement (RP) Groups

This command places the RP groups (refer to: `create_rp_group`). Both macros and standard cells are supported in the RP group.

EXAMPLES

The following example shows running a congestion driven placement.

```
prompt> create_fp_placement -congestion_driven
```

SEE ALSO

```
set_fp_placement_strategy(2)
report_fp_placement(2)
set_fp_macro_options(2)
set_fp_macro_array(2)
create_rp_group(2)
add_to_rp_group(2)
```

create_fp_plan_group_padding

Creates padding for the specified plan groups.

SYNTAX

```
status create_fp_plan_group_padding  
[-internal_widths {left right top bottom}]  
[-external_widths {left right top bottom}]  
[plan_groups]
```

Data Types

<i>left</i>	float
<i>right</i>	float
<i>top</i>	float
<i>bottom</i>	float
<i>plan_groups</i>	collection or list

ARGUMENTS

-internal_widths {*left right top bottom*}

Specifies the explicit per-side widths of the internal paddings in microns. You can use this option if you want to specify non-uniform internal padding widths. Specify all 4 sides in the indicated sequence. By default, this option is {1 1 1 1}.

-external_widths {*left right top bottom*}

Specifies the explicit per-side widths of the external paddings in microns. You can use this option if you want to specify non-uniform external padding widths. Specify all 4 sides in the indicated sequence. By default, this option is {0 0 0 0}.

plan_groups

Applies padding to the specified plan groups. By default, the tool applies padding to all plan groups in the current design.

DESCRIPTION

This command performs plan group padding, which sets placement blockages on the internal and external edges of the boundary. The internal padding is equivalent to the core-to-boundary spacing of the committed block. External padding is equivalent to macro padding. Internal padding should be set to at least 1 on all blocks to allow room for pins and pin routing. You can use the **-internal_widths** and **-external_widths** options to create non-uniform padding widths on a per-side basis. All values are in microns.

To remove plan group padding, use the **remove_fp_plan_group_padding** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates internal plan group padding on a per-side basis:

```
prompt> create_fp_plan_group_padding\  
-internal_widths {2 3 4 5}\  
-external_widths {6 7 8 9}
```

SEE ALSO

```
remove_fp_plan_group_padding(2)
```

create_fp_virtual_pad

Creates virtual power or ground pads for power network analysis and power network synthesis.

SYNTAX

```
status create_fp_virtual_pad  
[-nets string]  
[-layer string]  
[-point {x y}]  
[-load_file string]  
[-save_file string]
```

ARGUMENTS

-nets *string*
Specify the name(s) of the power or ground nets that you want power network analysis performed on. You can specify only one net at a time.

-layer *string*
Specify the layer name of a pad that you want to create.

-point {*x y*}
Specify a virtual pad point {*x y*} that you want to place. The value, *x* or *y* is float number. Unit in {*x y*} point is micron.

-load_file *string*
Specify the file name with virtual power pads pads.

-save_file *string*
Specify the file name to save virtual power pads.

DESCRIPTION

Creates virtual power or ground pads based on net nad, or load the virtual pads from the specified file, or save the virtual pads into the specified file

create_generated_clock

Creates a generated clock object.

SYNTAX

```
string create_generated_clock
[-name clock_name]
[-add]
source_objects
-source master_pin
[-master_clock clock]
[-divide_by divide_factor
 | -multiply_by multiply_factor]
[-duty_cycle percent]
[-invert]
[-preinvert]
[-edges edge_list]
[-edge_shift edge_shift_list]
[-combinational]
```

Data Types

<i>clock_name</i>	string
<i>source_objects</i>	list
<i>master_pin</i>	list
<i>clock</i>	string
<i>divide_factor</i>	integer
<i>multiply_factor</i>	integer
<i>percent</i>	float
<i>edge_list</i>	list
<i>edge_shift_list</i>	list

ARGUMENTS

-name *clock_name*

Specifies the name of the generated clock. If you do not use this option, the clock receives the same name as the first clock source specified in the **-source** option. If you specify the **-add** option, you must use the **-name** option and the clocks with the same source must have different names.

-add

Specifies whether to add this clock to the existing clock or to overwrite. Use this option to capture the case where multiple generated clocks must be specified on the same source, because multiple clocks fan into the master pin. Ideally, one generated clock must be specified for each clock that fans into the master pin. If you specify this option, you must also use the **-name** option.

Defining multiple clocks on the same source pin or port causes longer runtime and higher memory usage than a single clock, because the synthesis timing engine explores all possible combinations of launch and capture clocks. Use the **set_false_path** command to disable unwanted clock combinations. This option is ignored by default, unless multiple clocks analysis is enabled by

setting the **timing_enable_multiple_clocks_per_reg** variable to **true**.

source_objects

Specifies a list of ports or pins defined as generated clock source objects.

-source master_pin

Specifies the master clock pin, which is either a master clock source pin or a pin in the fanout of the master clock and driving the generated clock definition pin. The clock waveform at the master pin is used for deriving the generated clock waveform.

-master_clock clock

Specifies the master clock to be used for this generated clock if multiple clocks fan into the master pin.

-divide_by divide_factor

Specifies the frequency division factor. If the *divide_factor* is 2, the generated clock period is twice as long as the master clock period.

-multiply_by multiply_factor

Specifies the frequency multiplication factor. If the *multiply_factor* is 3, the period is one third as long as the master clock period.

-duty_cycle percent

Specifies the duty cycle (in percent), if frequency multiplication is used. This is a number between 0 and 100. The duty cycle is the high pulse width.

-invert

Inverts the generated clock signal no matter the sense of the source clock on master pin is unate or non-unate (in case of frequency multiplication and division).

-preinvert

Creates a generated clock based on the inverted clock signal only when the source clock on master pin has a non-unate sense, or the generated clock will not be inverted just as this option has not been specified. The difference between the **-invert** option and the **-preinvert** option is that the **-invert** option first creates the generated clock, then inverts the signal, and the **-preinvert** option first inverts the signal, and then creates the generated clock signal.

-edges edge_list

Specifies a list of positive integers that represents the edges from the source clock that are to form the edges of the generated clock. The edges are interpreted as alternating rising and falling edges and each edge must be not less than its previous edge. The number of edges must be an odd number and not less than 3 to make one full clock cycle of the generated clock waveform. The first edge must be greater than or equal to 1. For example, 1 represents the first source edge, 2 represents the second source edge, and so on.

-edge_shift edge_shift_list

Specifies a list of floating point numbers that represents the amount of shift, in library time units, that the specified edges are to undergo to yield the final generated clock waveform. The number of edge shifts specified must be equal to the number of edges specified. The values can be positive or

create_generated_clock

negative, positive indicating a shift later in time, negative a shift earlier in time. For example, 1 indicates that the corresponding edge is to be shifted by 1 library time unit.

-combinational

Specifies that the source latency paths for this type of generated clock only includes the logic where the master clock propagates along combinational paths. The source latency paths will not flow through sequential element clock pins, transparent latch data pins, or the source pins of other generated clocks.

DESCRIPTION

The **create_generated_clock** command creates a generated clock object in the current design. This command defines a list of objects as generated clock sources in the current design. You can specify a pin or a port as a generated clock object. The command also specifies the clock source from which it is generated. The advantage of using this command is that whenever the master clock changes, the generated clock changes automatically.

The generated clock can be created as a frequency divided clock with the **-divide_by** option, a frequency multiplied clock with **-multiply_by**, or an edge derived clock with **-edges**. In addition, the frequency divided or multiplied clock can be inverted with the **-invert** option. The shifting of edges of the edge-derived clock is specified with the **-edge_shift** option. The **-edge_shift** option is used for intentional edge shifts and not for clock latency. If a generated clock is specified with a *divide_factor* that is a power of 2 (1, 2, 4, ...), the rising edges of the master clock are used to determine the edges of the generated clock. If the *divide_factor* is not a power of 2, the edges are scaled from the master clock edges.

Using **create_generated_clock** on an existing **generated_clock** object overwrites the attributes of the **generated_clock** object.

The **generated_clock** objects are expanded to real clocks at the time of analysis.

The following commands can reference the **generated_clock**:

```
set_clock_latency
set_clock_uncertainty
set_propagated_clock
set_clock_transition
```

To display information about generated clocks, use the **report_clock** command.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example creates a frequency **-divide_by** 2 generated clock:

```
prompt> create_generated_clock -divide_by 2  
-source CLK [get_pins foo]
```

The following example creates a frequency **-divide_by 3** generated clock. If the master clock period is 30, and the master waveform is {24 36}, the generated clock period is 90 with waveform {72 108}.

```
prompt> create_generated_clock -divide_by 3  
-source CLK [get_pins div3/Q]
```

The following example creates a frequency **-multiply_by 2** generated clock with a duty cycle of 60%:

```
prompt> create_generated_clock -multiply_by 2  
-duty_cycle 60 -source CLK [get_pins foo1]
```

The following example creates a frequency **-multiply_by 3** generated clock with a duty cycle equal to the master clock duty cycle. If the master clock period is 30, and the master waveform is {24 36}, the generated clock period is 10 with waveform {8 12}.

```
prompt> create_generated_clock -multiply_by 3  
-source CLK [get_pins div3/Q]
```

The following example creates a generated clock whose edges are edges 1, 3, and 5 of the master clock source. If the master clock period is 30, and the master waveform is {24 36}, the generated clock period is 60 with waveform {24, 54}.

```
prompt> create_generated_clock -edges {1 3 5}  
-source CLK [get_pins foo2]
```

The following example shows the generated clock in the previous example with each derived edge shifted by 1 time unit. If the master clock period is 30, and the master waveform is {24 36}, the generated clock period is 60 with waveform {25, 55}.

```
prompt> create_generated_clock -edges {1 3 5}  
-edge_shift {1 1 1} -source CLK [get_pins foo2]
```

The following example creates an inverted clock:

```
prompt> create_generated_clock -divide_by 2 -invert
```

SEE ALSO

```
check_timing(2)  
create_clock(2)  
get_generated_clocks(2)  
remove_generated_clock(2)  
report_clock(2)  
set_clock_latency(2)  
set_clock_transition(2)  
set_clock_uncertainty(2)  
set_propagated_clock(2)  
timing_enable_multiple_clocks_per_reg(3)
```

create_generated_clock

create_ilm

Creates an interface logic model (ILM) for the current design.

SYNTAX

```
status create_ilm
[-identify_only]
[-extract_only]
[-ignore_ports port_list]
[-no_auto_ignore]
[-latch_level levels]
[-keep_macros]
[-keep_boundary_cells]
[-keep_full_clock_tree]
[-include_side_load boundary | all | none]
[-traverse_disabled_arcs]
[-compact none | output | all]
[-case_controlled_ports port_list]
[-must_connect_ports port_list]
[-keep_parasitics]
[-include_xtalk]
[-include_all_logic]
[-scenarios scenario_list]
[-verbose]
```

Data Types

<i>port_list</i>	collection
<i>levels</i>	int
<i>scenario_list</i>	list

ARGUMENTS

- identify_only**
Specifies that the interface logic of the design should be identified but not extracted. If you use this option, the extraction step is skipped. The default is to identify and extract interface logic.
- extract_only**
Specifies that the interface logic of the design should be extracted but not identified. If you use this option, the command assumes that the interface logic has already been identified. The default is to identify and extract interface logic.
- ignore_ports *port_list***
Specifies the input and inout ports whose fanout is to be ignored when identifying the interface logic. Clock ports are automatically ignored; you do not have to specify them in this list.
You can use this option to exclude the fanout of ports connected to chip-level nets, such as scan enable and reset, from impacting the contents of interface logic. If these nets are not explicitly ignored, they cause unnecessary logic, such as internal registers, to be part of the interface

logic on the current design.

You can also use the **-ignore_ports** option to selectively generate interface logic for a subset of the ports on a block. For example, you can use it on an ILM that models only the timing behavior on a subset of the ports on a block.

The default is to use all nonclock ports in identifying the contents of interface logic.

For functional correctness of the design, all the ignored ports will be connected to already identified logic.

-no_auto_ignore

Disables the automatic determination of input and inout ports whose fanout should be ignored when identifying interface logic.

By default, **create_ilm** ignores a port if the percentage of the total registers in the design in the transitive fanout of the input or inout port is greater than or equal to the threshold percentage specified in the

ilm_ignore_percentage variable (default is 25). The ports thus ignored while doing the ILM traversal are then connected to already identified logic.

This default helps to identify the test enable and reset ports of your design. Examine the current value of the **ilm_ignore_percentage** variable and change it, if needed. The default might potentially ignore ports you do not want to ignore or fail to ignore ports you do want to ignore. Carefully read the messages that the **create_ilm** command issues when you use default to see which ports have been ignored and to what percentage of registers they fanned out. Input and inout ports are never auto-ignored when creating a compact ILM, so the **-no_auto_ignore** option is not applicable when you are creating a compact ILM.

-latch_level levels

Specifies the number of logic levels over which time borrowing can occur for latch chains that are part of the interface logic.

When you use this option, note that the value of *levels* must account for the borrowing behavior of latches only on interface timing paths. If n represents a latch level, $n + 1$ represents the number of latches maintained in latch chains that originate at input ports. The $(n + 1)$ th latch functions as an edge-triggered register and decouples the I/O paths from the internal paths. Use this option only when there are latches in the interface logic for a block. Specifying this option might potentially result in less accurate models, because not all latches are allowed to borrow.

The default is to assume all latches found in interface logic are potential borrowers; thus, all logic from I/O ports to flip-flops or output ports are identified as belonging to interface logic.

-keep_macros

Specifies that all macro cells of the original design should be retained in the ILM. If you use this option, even the macro cells that are not part the interface logic of the original design are included in the ILM.

You can use the **-keep_macros** option to include all macros of the original design in the ILM. A cell is considered to be a macro cell if it is specified as macro in the physical library or if it is a large black-box cell.

The default is not to retain macro cells of the original design that are not part of the interface logic.

-keep_boundary_cells

Includes boundary cells for all ignored ports. You might have specified the

ignored ports by using the **-ignore_ports** option, or they might have been automatically selected by default.

Including the boundary cells ensures that design rule checking (DRC) from the top level considers the cells attached to ignored ports.

This option is no longer required because, by default, **create_ilm** either retains the connections from the ignored ports to the identified interface logic or it retains the timing critical boundary paths from the ignored ports. The side load inclusion on the boundary nets include any remaining cells not covered by these steps.

-keep_full_clock_tree

Retains the entire clock tree of all clocks created with the **create_clock** and **create_generated_clock** commands as part of the interface logic.

Because the entire clock tree of the block is retained in the ILM, the generated ILM is quite large.

The default is to retain the clock trees connected to the interface logic and the logic in the four critical clock paths (minimum rise, minimum fall, maximum rise, and maximum fall) from all clock sources for each scenario.

-include_side_load boundary | all | none

Specifies the side-load cells to include in the ILM. Side-load cells are not in the interface logic format, but they might affect the timing of an interface path.

Note that regardless of the value you specify, **create_ilm** always includes the side-load cells of all routed and extracted nets. This is done to ensure the most accurate ILM model for the postroute flow.

Use **boundary** for boundary side-load cells, which are those side-load cells on nets that are connected to the ports of the ILM. Including only boundary side-load cells in the ILM usually provides the best trade-off between model size and accuracy. This is the default option.

Use **all** to extract the most accurate model.

Use **none** to obtain the smallest possible (and less accurate) ILM.

-traverse_disabled_arcs

Specifies that the interface logic is not affected by disabled arcs and pins on the design. This option ignores the case analysis constraints to force the traversal of disabled arcs while determining interface logic. If you specify this option, the size of the model increases.

The default is to honor the constraints of the **set_case_analysis** command and not to traverse the timing arcs that are disabled due to case analysis. Thus the created ILM will not have such timing paths and logic.

The **-traverse_disabled_arcs** option is incompatible with the creation of compact ILMs (**-compact all | output**), because it disables case analysis, which affects the critical path timing. Because only critical timing paths are retained from each port when creating compact ILMs, this option is not applicable when creating compact ILMs.

-compact none | output | all

Specifies the technique used to reduce the size of the ILM.

By default (**-compact all**), the ILM retains the logic in the four critical timing paths (minimum rise, minimum fall, maximum rise, and maximum fall) from each input port to the first flip-flop (or to an output port if it is a synchronous path), as well as the paths from flip-flops to output ports. Along with the logic in these critical paths, all logic from I/O ports to flip-flop or output ports that contain latches are retained. You can control the number

of latch levels retained by using the **-latch_level** option. When you use this compaction technique, no ports are auto-ignored, because all the ports are analyzed for their associated timing-critical boundary paths.

If you limit compaction to output ports only by using the **-compact output** option, the size of the ILM increases. This option keeps the logic on the critical paths to the output ports, but it prevents compaction on paths from the input ports.

If you disable compaction altogether (**-compact none**), the ILM contains all logic from the boundary ports to the first flip-flop. If your design does not have complete and correct SDC files, you must use this option because compaction is slack-based for each port.

-case_controlled_ports *port_list*

Specifies a list of input and inout ports on which the case analysis value can be set or modified at the top level.

You can use this option to specify a set of ports whose case analysis value can change when linked with the top-level design. When this option is used, the compact ILM identifies the ports whose timing paths can be affected by the specified ports and treats them like non-compact ports. Non-compact interface logic is retained for these ports instead of only the four critical timing paths (minimum rise, minimum fall, maximum rise, and maximum fall). This option can also affect what logic gets included in a noncompact ILM (**create_ilm -compact none**), because the fanout from the port is treated as if case analysis was not applied on the port.

-must_connect_ports *port_list*

Specifies a list of ports that must be connected to already identified logic for functional correctness of design.

You can use this option to specify a set of ports, such as scan enable or reset, whose connectivity is necessary with the already identified logic. This option has no effect when used with the **-compact none** option, because **create_ilm** retains the entire fanout from all ports.

-keep_parasitics

Includes detailed parasitics in the ILM view.

If you do not invoke the **extract_rc** command before invoking this command, **extract_rc** is automatically run to extract the parasitics when you use this option. The design must be routed (global routing, track assignment, or detail routing) for **extract_rc** to succeed and generate the required parasitics. These parasitics in the ILM view are automatically used when the **extract_rc** command is run at the top level.

If you plan to use the generated ILM in the top-level flow with coupling capacitances (for example, use **extract_rc -coupling_cap** at the top level), the generated ILM must have coupling capacitances.

By default, the ILM is generated with coupling capacitances if you enabled the signal integrity options before generating the ILM or you specify the **-include_xtalk** option when you run **create_ilm**. To generate an ILM with coupling capacitances, the design must be either track assigned or detail routed.

If you plan to use the generated ILM in the top-level signal integrity flow use the **-include_xtalk** option when you run **create_ilm** to generate an ILM with both coupling capacitance data and signal integrity information.

The ILM generated by the **-keep_parasitics** option can be used as the timing and logic model for top-level timing-driven routing, postroute timing analysis and optimization, and the **route_opt** core command.

Note that top-level routing uses the FRAM view for routing, while top-level timing-driven routing uses the FRAM view for routing and the ILM for timing. This option is on by default if design is routed (global routing, track assignment, or detail routing) and is ignored for unrouted designs.

-include_xtalk

Stores signal integrity (SI) and coupling capacitance information with the ILM view.

This option is honored only when the design is track assigned or detail routed. It is ignored for unrouted or global routed designs.

-include_all_logic

Includes all cells, nets, and pins in the block in the ILM.

In general, you should not use this option, because the ILM can be huge if the block is huge. You should use this option only in special situations, such as for a clock generator block or for a very, very small block.

-scenarios *scenario_list*

Specifies a list of scenarios that must be used by tool to generate the ILM model. Interface logic is identified for each scenario and the union of the per-scenario interface logic is retained in the ILM.

By default, all scenarios are used for a multicorner-multimode design. To avoid high memory usage, you should activate only one scenario or a minimal set of scenarios before running this command. For more details, see the EXAMPLES section.

This option is not supported for non-multicorner-multimode designs.

-verbose

Prints comparative statistics about the number of design objects in the original design and in the ILM. By default, the command prints the comparative statistics.

DESCRIPTION

This command creates an interface logic model (ILM) for the current design by discarding all the logic that is not part of the interface logic. After command execution, the ILM model is saved to disk and the design in memory is the unmodified original design (the state before running **create_ilm**).

ILMs provide an accurate and robust model generation solution for a hierarchical design flow. ILMs embody a structural approach to model generation; the original gate-level netlist for a block is modeled by another gate-level netlist that contains the interface logic of the block.

By preserving interface logic without any modification, an ILM is a highly accurate representation of the original design. An ILM does not abstract logic; it discards the logic that is not required for modeling boundary timing. For typical designs and technologies in postlayout flows, an ILM preserves the original block timing to within 10 ps.

The **create_ilm** command first determines the cells, nets, and pins of the current design that are part of the interface logic. The interface logic contains all logic whose timing is impacted by or impacts the external environment of a block (details given below).

The following cells and clock trees are part of interface logic:

- All cells in timing paths that lead from input ports to registers or output ports that terminate the paths.
- All cells in timing paths that lead to output ports from registers or input ports that originate the paths.
- The clock trees that drive interface registers, including any registers in the clock tree. Clock-gating circuitry is part of interface logic if it is driven by external ports, but not if it is driven by registered outputs on a block.

Notice that interface logic does not include internal register-to-register paths and logic on a block associated only with these paths.

The **create_ilm** command implicitly performs an **update_timing** on the design, if required.

The following physical design information is included in the ILM:

- Locations for all leaf cells in the ILM
- Locations for all ports of the ILM
- Nets that traverse the ILM boundary are specially marked so that wire length and capacitance for these nets are estimated again when you run the **place_opt** or **clock_opt** commands at the top level. When the wire-length estimation is performed for these nets, it is ensured that they go through the ILM port for greater accuracy.

Use of ILM models is expected to significantly improve both memory and run time for performing top-level optimization of large designs.

IEEE 1801 (UPF) constraints associated with the interface logic are retained in the ILM to enable the hierarchical UPF flow.

Multimode Notes

- **create_ilm** automatically detects the presence of multiple scenarios (multiple modes or corners). In this case, the interface logic is traversed for each scenario -- if an interface timing path is disabled in one scenario but enabled in another, the path is included in the interface logic. You must use the **-scenarios** option.
- The scenarios defined at the top and block levels must be identical in name and number. You must use the **-scenarios** option.
- To have the appropriate timing constraints from the ILMs available at the top

level, you must either use the **propagate_constraints** command to propagate them up from the ILMs or use the **read_sdc** command to apply them from the top level.

Multicorner Notes

- Detailed parasitics are extracted and stored with the ILM for each specified TLUPlus file.
- The TLUPlus files used at the top and block levels must be identical across all scenarios.

Multicorner-Multimode Support

This command uses information from active scenarios only.

EXAMPLES

The following example shows how to create an ILM for a multicorner-multimode design by specifying the list of scenarios with the **all_scenarios** command.

```
prompt> open_mw_cel top
prompt> extract_rc
prompt> create_ilm -scenarios [all_scenarios]
```

The following example shows how to create an ILM for a multicorner-multimode design by specifying the list of scenarios by name. You should have one or a minimal number of scenarios active prior to running this command.

```
prompt> open_mw_cel top
prompt> extract_rc
prompt> all_active_scenarios
prompt> create_ilm -scenarios {scenario1 scenario2 scenario3}
```

The following example shows how to save an ILM in a Milkyway design library. The ILM is saved in the ILM view of the Milkyway design library. The example creates an ILM for the current design specifying that the fanin and fanout of the reset and scan_enable ports should be ignored.

```
prompt> create_ilm -ignore_ports [get_ports {reset scan_enable}]
```

The following example sets the **ilm_ignore_percentage** variable to 50, so that a port is ignored if the percentage of the total design registers in its transitive fanout

is greater than 50.

```
prompt> set ilm_ignore_percentage 50
ilm_ignore_percentage = "50"
prompt> create_ilm
```

The following example uses the **-keep_macros** and **-keep_boundary_cells** options. All macros and boundary cells for ignored ports are to be included in the ILM. The **-latch_level** option with a value of 1 states that the first latch encountered in I/O paths could potentially borrow, but the second latch can be treated as an edge-triggered device. Thus, two levels of latches are maintained in the interface logic.

```
prompt> create_ilm -keep_macros \
-keep_boundary_cells -latch_level 1
```

The following example shows the use of the **-keep_parasitics** option.

```
prompt> open_mw_cel f11
prompt> create_ilm -keep_parasitics
```

The following example includes the signal integrity (SI) information with the ILMs.

```
prompt> open_mw_cel f11
prompt> create_ilm -include_xtalk
```

SEE ALSO

```
all_scenarios(2)
create_clock(2)
create_generated_clock(2)
extract_rc(2)
open_mw_cel(2)
propagate_constraints(2)
propagate_ilm(2)
read_sdc(2)
save_mw_cel(2)
set_case_analysis(2)
write(2)
ilm_ignore_percentage(3)
```

create_ilm_models

Creates interface logic models (ILMs) for the specified macros of the design.

SYNTAX

```
status create_ilm_models
[-no_auto_ignore]
[-latch_level levels]
[-keep_full_clock_tree]
[-keep_parasitics]
[-verbose]
[-include_xtalk]
[-include_all_logic]
[-compact none | output | all [-in_context]]
[-scenarios scenario_list]
macro_reference_list
```

Data Types

<i>levels</i>	int
<i>scenario_list</i>	list
<i>macro_reference_list</i>	list

ARGUMENTS

-no_auto_ignore

Disables the automatic determination of input and inout ports whose fanout should be ignored when identifying interface logic.

By default, **create_ilm** ignores a port if the percentage of the total registers in the design in the transitive fanout of the input or inout port is greater than or equal the threshold percentage specified in the **ilm_ignore_percentage** variable (default 25). The ports thus ignored while doing the ILM traversal are then connected to already identified logic.

This default helps to identify the test enable and reset ports of your design. You should examine the current value of the **ilm_ignore_percentage** variable and change it, if needed. The default might potentially ignore ports you do not want to ignore or fail to ignore ports you do want to ignore. Carefully read the messages that the **create_ilm** command issues when you use default to see which ports have been ignored and to what percentage of registers they fanned out.

When you create a compact ILM (**-compact all**), **create_ilm** retains only the logic in the four critical timing paths (minimum rise, minimum fall, maximum rise, and maximum fall) from all input ports and to all output ports. Input and inout ports are never auto-ignored when creating a compact ILM, so it is not necessary to specify the **-no_auto_ignore** option.

-latch_level *levels*

Specifies the number of logic levels over which time borrowing can occur for latch chains that are part of the interface logic.

When you use this option, note that the value of *levels* must account for the borrowing behavior of latches only on interface timing paths. If *n* represents a latch level, *n* + 1 represents the number of latches maintained in latch

chains that originate at input ports. The $(n + 1)$ th latch functions as an edge-triggered register and decouples the I/O paths from the internal paths. Use this option only when there are latches in the interface logic for a block. Specifying this option might potentially result in less accurate models, because not all latches are allowed to borrow.

The default is to assume all latches found in interface logic are potential borrowers; thus, all logic from I/O ports to flip-flops or output ports are identified as belonging to interface logic.

When you create a compact ILM (**-compact all**), **create_ilm** retains only the logic in the four critical timing paths (minimum rise, minimum fall, maximum rise, and maximum fall) from all input ports and to all output ports. The **-latch_level** option is ignored for compact ILMs.

-keep_full_clock_tree

Retains the entire clock tree of all clocks created with the **create_clock** and **create_generated_clock** commands as part of the interface logic.

Because the entire clock tree of the block is retained in the ILM, the generated ILM is quite large.

The default is to retain the clock trees connected to the interface logic and the logic in the four critical clock paths (minimum rise, minimum fall, maximum rise, and maximum fall) from all clock sources for each scenario.

-keep_parasitics

Includes detailed parasitics in the ILM view.

If you do not invoke the **extract_rc** command before invoking this command, **extract_rc** is automatically run to extract the parasitics when you use this option. The design must be routed (global, track assign, or detail) for **extract_rc** to succeed and generate required parasitics. These parasitics in the ILM view are automatically used when the **extract_rc** command is run at the top-level.

If you plan to use the generated ILM in the top-level flow with coupling capacitances (for example, use **extract_rc -coupling_cap** at the top-level), the generated ILM must have coupling capacitances.

By default, the ILM is generated with coupling capacitances if you enabled the signal integrity options before generating the ILM or you specify the **-include_xtalk** option when you run **create_ilm**. To generate an ILM with coupling capacitances, the design must be either track assigned or detail routed.

If you plan to use the generated ILM in the top-level signal integrity flow use the **-include_xtalk** option when you run **create_ilm** to generate an ILM with both coupling capacitance data and signal integrity information.

The ILM generated by the **-keep_parasitics** option can be used as the timing and logic model for top-level timing-driven routing, postroute timing analysis and optimization, and the **route_opt** core command.

Note that top-level routing uses the FRAM view for routing, while top-level timing-driven routing uses the FRAM view for routing and the ILM for timing. This option is on by default if design is routed (global routing, track assignment, or detail routing) and is ignored for unrouted designs.

-verbose

Prints comparative statistics about the number of design objects in the original design and in the ILM. By default, the command prints the comparative statistics.

`-include_xtalk`

Stores signal integrity (SI) and coupling capacitance information with the ILM view.

This option is honored only when the design is track assigned or detail routed. It is ignored for unrouted or global routed designs.

`-include_all_logic`

Includes all cells, nets, and pins in the block in the ILM.

In general, you should not use this option, because the ILM can be huge if the block is huge. You should use this option only in special situations, such as for a clock generator block or for a very, very small block.

`-compact none | output | all`

Specifies the technique used to reduce the size of the ILM.

By default (**-compact all**), the ILM contains the logic in the four critical timing paths (minimum rise, minimum fall, maximum rise, and maximum fall) from each input port to the first flip-flop (or to an output port if it is a synchronous path), as well as the paths from flip-flops to output ports. When you use this compaction technique, no ports are auto-ignored, because all the ports are analyzed for their associated timing-critical boundary paths.

If you limit compaction to output ports only by using the **-compact output** option, the size of the ILM increases. This option keeps the logic on the critical paths to the output ports, but it prevents compaction on paths from the input ports.

If you disable compaction altogether (**-compact none**), the ILM contains all logic from the boundary ports to the first flip-flop. If your design does not have complete and correct SDC files, you must use this option because compaction is slack-based for each port.

`-in_context`

Generates compact ILMs using the transition time at the inputs of the blocks in the top-level design. The generated ILMs are more accurate because they are in the context of the top-level design.

`-scenarios scenario_list`

Specifies a list of scenarios that must be used by the tool to generate the ILM model. Interface logic is identified for each scenario and the union of the per-scenario interface logic is retained in the ILM.

By default, all scenarios are used for a multicorner-multimode design. To avoid high memory usage, you should activate only one scenario or a minimal set of scenarios before running this command. For more details, see the **EXAMPLES** section.

This option is not supported for non-multicorner-multimode designs.

`macro_reference_list`

Specifies the soft macros for which ILM models need to be created. This is a required option.

DESCRIPTION

This command creates interface logic models (ILMs) for the specified macros in the current design.

This command calls the **create_ilm** command to create the ILM for each soft macro.

Multicorner-Multimode Support

This command uses information from active scenarios only.

EXAMPLES

In the following example, ILM models are created for a multicorner-multimode design by specifying the list of scenarios with the **all_scenarios** command. The scenario list is used for both blk1 and blk2.

```
prompt> open_mw_cel top
prompt> create_ilm_models -scenarios [all_scenarios] {blk1 blk2}
```

In the following example, ILM models are created for a multicorner-multimode design by specifying a list of scenarios by name. The scenario list is used for both blk1 and blk2.

```
prompt> open_mw_cel top
prompt> create_ilm_models -scenarios {scenario1 scenario2} {blk1 blk2}
```

SEE ALSO

```
all_scenarios(2)
create_ilm(2)
ilm_ignore_percentage(3)
```

create_lib_track

Creates wire tracks for a library.

SYNTAX

```
status create_lib_track  
[-lib lib_name]  
[-tile tile_name]  
[-dir dir_list]  
[-offset offset_list]
```

Data Types

<i>lib_name</i>	string
<i>tile_name</i>	string
<i>dir_list</i>	list
<i>offset_list</i>	list

ARGUMENTS

-lib *lib_name*
Specify library where wire track is created. If not specified, the wire tracks will be created in current open library.

-tile *tile_name*
Specifies the tile where the wire track information is stored. Default value is "unit" and therefore the information will be stored in unitTile.CEL.

-dir *dir_list*
Specifies preferred routing direction for each routing layer one by one. The list is enclosed in parenthesis {}. The format is -dir {poly V/H metall V/H metall V/H ...} Where 'V' for "vertical" and 'H' for "horizontal". Both mask name and user-defined layer name of routing layers can be used in the list.

-offset *offset_list*
Specifies offset for each routing layer one by one. The list is enclosed in parenthesis {}. The format is -offset {poly value metall value metall value ...} The unit of value is micron. Both mask name and user-defined layer name of routing layers can be used in the list.

DESCRIPTION

This command creates wire tracks for a library. If some routing directions or offsets in routing layers and poly layer are not set, Default value will be used. Incremental setting mode is supported. The recommended value of offset is one half of the pitch. A status indicating success or failure is returned.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets routing directions and offset for opened library "design", which has 9 routing layers. unitTile will store the wire track information.

```
prompt> open_mw_lib design
prompt> create_lib_track -dir {poly V metal1 H metal2 V \
metal3 H metal4 V metal5 H metal6 V metal7 H metal8 V \
metal9 H} -offset {poly 0.00 metal1 0.14 metal2 0.16 \
metal3 0.14 metal4 0.14 metal5 0.14 metal6 0.14 metal7 0.14 \
metal8 0.14 metal9 0.14}
1
```

SEE ALSO

```
create_mw_lib(2)
open_mw_lib(2)
```


create_macro_fram

Creates a FRAM view of a macro cell by extracting blockage, pin, and via information from the CEL view.

SYNTAX

```
status create_macro_fram
[-library_name library_name]
[-cell_name cell_name]
[-preserve_all_metal_blockage]
[-routing_blockage_output_layer metBlk | rGuide | zeroG]
[-treat_all_blockage_as_thin_wire]
[-treat_metal_blockage_as_thin collection_of_layers]
[-extract_blockage_by_block_core_with_margin collection_of_layer_value]
[-extract_blockage_by_merge_with_threshold collection_of_layer_value]
[-identify_macro_pin_by_pin_text]
[-extract_pin_connectivity_through collection_of_layers]
[-poly_pin_text_layers {list_of_layers}]
[-m1_pin_text_layers {list_of_layers}]
[-m2_pin_text_layers {list_of_layers}]
[-m3_pin_text_layers {list_of_layers}]
[-m4_pin_text_layers {list_of_layers}]
[-m5_pin_text_layers {list_of_layers}]
[-m6_pin_text_layers {list_of_layers}]
[-m7_pin_text_layers {list_of_layers}]
[-m8_pin_text_layers {list_of_layers}]
[-m9_pin_text_layers {list_of_layers}]
[-m10_pin_text_layers {list_of_layers}]
[-m11_pin_text_layers {list_of_layers}]
[-m12_pin_text_layers {list_of_layers}]
[-m13_pin_text_layers {list_of_layers}]
[-m14_pin_text_layers {list_of_layers}]
[-m15_pin_text_layers {list_of_layers}]
[-nwell_pin_text_layers list_of_layers]
[-pwell_pin_text_layers list_of_layers]
[-deep_nwell_pin_text_layers list_of_layers]
[-deep_pwell_pin_text_layers list_of_layers]
[-pin_must_connect_area_layer_number layer_number_name...]
[-auto_pin_must_connect_area_threshold collection_of_layer_value]
[-extract_via_within_pin_area_only]
[-extract_via_on_layer collection_of_layers]
```

Data Types

<i>library_name</i>	string
<i>cell_name</i>	string
<i>collection_of_layers</i>	collection
<i>collection_of_layer_value</i>	collection
<i>list_of_layers</i>	list

ARGUMENTS

`-library_name library_name`

Specifies the name of the library containing the macro cell.

`-cell_name cell_name`

Specifies the name of the macro cell for which to create a FRAM view. The CEL view must already exist in the library.

`-preserve_all_metal_blockage`

Preserves all system metal blockage layers as they exist in the CEL view and converts all system metal blockage layers to real metal layers. By default, the extraction process trims system metal blockages around pins as needed to prevent design rule violations. Using this option, the system metal blockages are retained exactly, so be sure that the preserved system metal blockages do not violate the design rules.

`-routing_blockage_output_layer metBlk | rGuide | zeroG`

Specifies the type of output layer produced in the FRAM view to represent routing blockages: **metBlk** (system metal blockage), **rGuide** (route guide), or **zeroG** (zero spacing route guide). The router recognizes all three types of layers as blockage and will not route over them. The default output layer is **metBlk** (system metal blockage). The router treats system metal blockage as thin metal for all design rule checking.

`-treat_all_blockage_as_thin_wire`

This option works only for non-standard cells. It causes real metal blockages to be treated as thin wire for design rule checking purposes. By default, real metal blockages are treated as fat wire if fat or thin wire if thin. The spacing rules can be different for thin versus fat wire, which affects the spacing required between routes and real metal blockages. This option setting also affects the trimming of real metal blockages during extraction. The extraction process trims real metal blockages around pin areas to allow connections to be routed to the pins without violating the spacing to the real metal blockage areas.

`-treat_metal_blockage_as_thin collection_of_layers`

Use this option when system metal blockage covers all of the cell in a specific layer and you want the router to keep a spacing of at least minSpacing between the route wire and the macro cell boundary in that layer. With this option, the command shrinks the system metal blockage by (max value in fatTblSpacing) - (minSpacing) and converts it to real metal blockage. This option still treats the real metal blockage as fat metal but reshapes the real metal blockage to cause the router to keep a spacing of at least minSpacing between the route wire and the macro cell boundary in that layer.

These are the mask layers that can be specified in this option:

poly m1 m2 m3 ... m15

`-extract_blockage_by_block_core_with_margin collection_of_layer_value`

Causes each specified layer to be extracted with a core blockage that covers the whole macro area for that layer, with a specified margin between the blockage area and the macro cell boundary. The extraction process trims back the blockage away from pins in the same layer, leaving space between the pins and blockage according to the spacing rule for that layer, treated as either

thin or fat wire, depending on the related option settings.
Specify each layer name with a single value as in the following example:

```
{poly 0 m1 -1 m2 0 m3 0.45 ... }
```

The value for each layer can be -1, 0, or a positive number. The value -1 means do not create a core blockage; this is the same as omitting the layer name from the list. The number 0 means use the default margin, which is calculated as twice the spacing value defined for the layer in the technology file. (This spacing value is the largest of the following values in the technology file: MinSpacing, fatThinMinSpacing, and the largest value in the fatTblSpacing table.)

A positive number sets the margin explicitly to that value. The margin is the distance between the blockage and the macro boundary.

These are the mask layers that can be specified in this option:

```
poly m1 m2 m3 ... m15
```

If neither **-extract_blockage_by_block_core_with_margin** nor **-extract_blockage_by_merge_with_threshold** is used to specify the type of blockage to generate for a layer, the command creates a core blockage that covers the whole macro area for that layer, except for the trimming of blockage away from pins in the same layer according to the spacing rules for the layer. This type of blockage extraction is called "Block All".

-extract_blockage_by_merge_with_threshold *collection_of_layer_value*

Causes each specified layer to be extracted as it exists in the CEL view, but with merging of geometries that are close together. If two nearby geometries are within the specified spacing threshold, the extraction process fills in the space between the two shapes, thereby creating a single larger shape for the blockage in the FRAM view.

Specify each layer name with two values. The first value is the X (horizontal) spacing threshold and the second value is the Y (vertical) spacing threshold. For example,

```
{m1 3 3 m2 4 4 m3 4 4 ... }
```

One possible value to use for the spacing thresholds is (2*MinSpacing + MinWidth), which is the narrowest space that could allow a single route to pass between two shapes in that layer.

If both thresholds are set to zero for a layer, the geometries are extracted as they exist in the CEL view without merging.

These are the mask layers that can be specified in this option:

```
poly m1 m2 m3 ... m15
```

-identify_macro_pin_by_pin_text

Identifies macro pins by considering the text layers associated with the pin layers. By default, when this option is not used, the extraction process assumes that the macro cell was created by Synopsys tools and already contains pin data.

When this option is used, the following **create_macro_fram** command options are also used to further specify the pin extraction process:

```
-extract_pin_connectivity_through  
-poly_pin_text_layers  
-m1_pin_text_layers  
-m2_pin_text_layers  
-m3_pin_text_layers  
...  
-m15_pin_text_layers  
-nwell_pin_text_layers
```

-pwell_pin_text_layers
-deep_nwell_pin_text_layers
-deep_pwell_pin_text_layers
Otherwise, these options have no effect.

-extract_pin_connectivity_through *collection_of_layers*
Specifies the cut layers through which to extract pin connectivity. Using this option causes the command to traverse connectivity through the specified cut and via layers to identify connections between pins on different layers. By default, the tool does not traverse connectivity through cut layers. These are the mask layers that can be specified in this option:
polyCont v1 v2 v3 ... v14 bv1 bv2

-poly_pin_text_layers {*list_of_layers*}
Specifies the text layers associated with poly pins. If the pin text is on the same layer as the pin geometry, you do not need to use this option. However, if the pin text is on one or more layers different from the pin geometry layer, use this option to identify the associated text layers. The layers can be identified by the layer name or layer number.

-m1_pin_text_layers {*list_of_layers*}
Specifies the text layers associated with m1 pins. If the pin text is on the same layer as the pin geometry, you do not need to use this option. However, if the pin text is on one or more layers different from the pin geometry layer, use this option to identify the associated text layers. The layers can be identified by the layer name or layer number.

-m2_pin_text_layers {*list_of_layers*}
Specifies the text layers associated with m2 pins.

-m3_pin_text_layers {*list_of_layers*}
Specifies the text layers associated with m3 pins.

-m4_pin_text_layers {*list_of_layers*}
Specifies the text layers associated with m4 pins.

-m5_pin_text_layers {*list_of_layers*}
Specifies the text layers associated with m5 pins.

-m6_pin_text_layers {*list_of_layers*}
Specifies the text layers associated with m6 pins.

-m7_pin_text_layers {*list_of_layers*}
Specifies the text layers associated with m7 pins.

-m8_pin_text_layers {*list_of_layers*}
Specifies the text layers associated with m8 pins.

-m9_pin_text_layers {*list_of_layers*}
Specifies the text layers associated with m9 pins.

-m10_pin_text_layers {*list_of_layers*}
Specifies the text layers associated with m10 pins.

-m11_pin_text_layers {*list_of_layers*}
 Specifies the text layers associated with m11 pins.

-m12_pin_text_layers {*list_of_layers*}
 Specifies the text layers associated with m12 pins.

-m13_pin_text_layers {*list_of_layers*}
 Specifies the text layers associated with m13 pins.

-m14_pin_text_layers {*list_of_layers*}
 Specifies the text layers associated with m14 pins.

-m15_pin_text_layers {*list_of_layers*}
 Specifies the text layers associated with m15 pins.

-nwell_pin_text_layers *list_of_layers*
 Specifies the text layers associated with N-well bias pg pins. If the pin text is on the same layer as the pin geometry, you do not need to use this option. However, if the pin text is on one or more layers different from the pin geometry layer, use this option to identify the associated text layers. The layers can be identified by the layer name or layer number.

-pwell_pin_text_layers *list_of_layers*
 Specifies the text layers associated with P-well bias pg pins.

-deep_nwell_pin_text_layers *list_of_layers*
 Specifies the text layers associated with deep N-well bias pg pins.

-deep_pwell_pin_text_layers *list_of_layers*
 Specifies the text layers associated with deep P-well bias pg pins.

-pin_must_connect_area_layer_number *layer_number_name...*
 Specifies for each connection layer the associated spare layer being used to designate the required via connection area within the pin geometry. By default, the router can connect a wire to any part of a large pin. To restrict the connection to a subarea of a large pin, specify the subarea polygon on a spare layer and specify the spare area associated with each connection layer using this option. List the connection layers and associated spare layers as in the following example:
 {m1 31 m2 32 m3 33 ...}
 You can specify each layer by its layer number or layer name. These are the mask layers that can be specified in this option:
 poly m1 m2 m3 ... m15

-auto_pin_must_connect_area_threshold *collection_of_layer_value*
 Specifies the width threshold for automatic determination of the required via connection area within the pin geometry. By default, the router can connect a wire to any part of a large pin. To automatically restrict the connection to a subarea of a pin with an irregular shape, use this option and specify each layer and its associated width threshold. For example,
 {m1 0.56 m2 0.63 m3 0.63 ... }
 Areas within a pin having a width smaller than the threshold are excluded from allowing a via connection to be made. For a fat pin, the first-dimension fat contact threshold is used to calculate the allowed connection area, so that the router does not drop a fat contact on the thin part of a fat pin.

These are the mask layers that can specified in this option:
poly m1 m2 m3 ... m15

-extract_via_within_pin_area_only

Causes the extraction process to extract vias existing in pin areas in the CEL view. By default, no vias are extracted from the CEL view. Use this option when you want the router to be aware of any vias existing in a pin when it makes a connection to the pin.

-extract_via_on_layer *collection_of_layers*

Specifies the mask layer or layers from which to extract vias. By default, no vias are extracted from the CEL view. Use this option when you want the router to be aware of any vias on the corresponding mask layer when it makes a connection to the pin. These are the mask layers that can specified in this option:

polyCont v1 v2 Ö v14 bv1 bv2

DESCRIPTION

This command creates a FRAM view of a macro cell by extracting blockage, pin, and via information from the CEL view. The FRAM view of a macro cell contains all the information needed for placement and routing of the macro and excludes cell-internal information that is not needed for placement and routing. The exclusion of unnecessary data can reduce the database size, increase the routing accuracy, and reduce the routing time.

Both the **extract_blockage_pin_via** command and the **create_macro_fram** command create FRAM views of cells by extracting blockage, pin, and via information from CEL views. Use the **extract_blockage_pin_via** command for extraction of standard cells and pads. Use the **create_macro_fram** command for extraction of macros. The extraction process is the same for both commands. However, they offer slightly different options with respect to extraction of blockages and via regions.

System metal blockages defined in the CEL view can be converted into system metal blockages, route guides, or zero-spacing route guides in the FRAM view. Proper extraction of system metal blockages can be used to prevent DRC violations between routes and cell-internal metal, while allowing the greatest possible flexibility to the router. You can choose to preserve all metal blockages as defined in the CEL view or have them trimmed away from pins to allow route connections while preventing DRC violations. For a non-standard cell, you can also choose to have some or all routing layers treated as thin wire rather than fat wire for blockage extraction and DRC purposes.

For each metal layer, you can choose to create a core blockage that covers the whole macro area, but with a specified margin between the blockage area and the macro cell boundary. The extraction process trims back the blockage away from pins in the same layer, leaving space between the pins and blockage according to the spacing rule for that layer, treated as either thin or fat wire, depending on the related option settings.

Instead of a core blockage, you can choose to extract blockage from the existing metal in the CEL view, with merging of geometries that are close together. If two nearby geometries are within a specified spacing threshold, the extraction process fills in the space between the two shapes, thereby creating a single larger shape

for the blockage in the FRAM view.

If the CEL view contains multiple-layer pins, use the **-extract_pin_connectivity_through** option to specify the cut and via layers used in those pins. Then the command traces the connectivity through the specified cut and via layers to identify connections between pins on different layers.

If the macro cell was created by an external tool and does not already have pin identification information in the database, use the **-identify_macro_pin_by_pin_text** option, which causes the tool to identify the macro pins by considering the text in the text layers associated with the pin layers. If the layers used for pin text are different from the pin geometry layers, use the **-identify_macro_pin_by_pin_text** option, and use the respective command options to specify which text layers are associated with each pin layer (for example, **-m1_pin_text_layers** for the m1 layer).

As part of the FRAM extraction process, the tool determines the area within each pin where the center of a via can connect. This region, called the via region for the pin, is part of the generated FRAM view. This information gives the router guidance about where it can make connections to the pin. As a result, the router can find the via connection position quickly. Calculating the via region at extraction time saves significant runtime for router. However, this information only provides guidance. Even without the via region information, the router can still find the via connection position, but it must spend more runtime to find it.

The extraction process offers two methods for determining the via region for each pin, invoked by the following options:

```
-pin_must_connect_area_layer_number {layer number_name ...}  
-auto_pin_must_connect_area_threshold {layer width ...}
```

Using the "must connect area layer" method, you explicitly specify the geometry of the via region for each pin on a spare layer. Then you use the option to specify each pin layer and its corresponding via region layer.

Using the "automatic pin must connect area" method, the tool automatically determines the via region for each pin. This method is suitable for pins with irregular shapes such as fingered geometries. For each pin layer, you specify a width threshold. Areas within a pin having a width smaller than the threshold are excluded from allowing a via connection to be made. For a fat pin, the first-dimension fat contact threshold is used to calculate the allowed connection area, so that the router will not drop a fat contact on the thin part of fat pin.

If you use neither the "must connect area layer" option nor the "automatic pin must connect area" option, the command determines the via region of each pin by itself. If no via region is produced on a pin, the router must determine an acceptable connection to the pin by itself.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example extracts the blockage, pin, and via information from the CEL view of the macro cell called `my_macro` in the library called `arc_rams_LM`, thereby produced a FRAM view of the cell containing only the information needed for placement and routing. The macro does not already have pin identification information, so that information is extracted from the pin text associated with each pin. The pin text for layer `m3` is contained in layer number 33, not in the `m3` layer itself.

```
prompt> create_macro_fram -library_name arc_rams_LM \  
        -cell_name my_macro \  
        -identify_macro_pin_by_pin_text \  
        -m3_pin_text_layers 33
```

The following example performs FRAM view extraction for the macro `my_macro` in the `arc_rams_LM` library. For layers `m1` and `m2`, it creates a core blockage that covers the entire macro with a margin of 5.0 units away from the macro edges, and also trimmed back to expose the pins in the same layer with a margin equal to the fat metal spacing requirement.

```
prompt> create_macro_fram -library_name arc_rams_LM \  
        -cell_name my_macro \  
        -extract_blockage_by_block_core_with_margin \  
        {m1 5.0 m2 5.0} \  
        \
```

The following example performs FRAM view extraction for the macro `my_macro` in the `arc_rams_LM` library. For layers `m1` and `m2`, it creates the blockage by merging the existing metal layers in the CEL view. Geometries that have a spacing of 2.0 or less in the X direction or 2.5 or less in the Y direction are merged by filling in the gap between the geometries. The geometries are also trimmed back to expose the pins in the same layer with a margin equal to the thin metal spacing requirement. The router treats the real metals as thin wire for design rule checking purposes.

```
prompt> create_macro_fram -library_name arc_rams_LM \  
        -treat_all_blockage_as_thin_wire \  
        -cell_name my_macro \  
        -extract_blockage_by_merge_with_threshold \  
        {m1 2.0 2.5 m2 2.0 2.5} \  
        \
```

SEE ALSO

`extract_blockage_pin_via(2)`

create_mw_cel

Creates a Milkyway design.

SYNTAX

```
status create_mw_cel
[-view CEL | FRAM | FILL | err | ILM]
[-verbose]
mw_cel_name
```

Data Types

mw_cel_name string

ARGUMENTS

-view CEL | FRAM | FILL | err | ILM
Specifies the view name of the Milkyway design to create. The **CEL**, **FRAM**, **FILL**, **err** and **ILM** arguments are mutually exclusive; you can use only one. By default, the command creates a **CEL** view Milkyway design.

-verbose
Prints additional messages.

mw_cel_name
Specifies the name of the Milkyway design to create. The length of a Milkyway design name must be less than 127 characters. Do not specify a view name or version in the *mw_cel_name*. For example, *test.CEL* or *test.CEL;2* are not valid values for *mw_cel_name*. To specify the view, use **-view** option.

DESCRIPTION

This command creates a new Milkyway design. It automatically opens this newly-created Milkyway design and sets it as the current Milkyway design.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates a Milkyway design named *top* for CEL view.

```
prompt> create_mw_cel top
{"top"}

prompt> current_mw_cel
{"top"}
```

The following example creates a Milkyway design named *top* for err view.

```
prompt> create_mw_cel top -view err  
{"top"}
```

```
prompt> current_mw_cel  
{"top"}
```

SEE ALSO

```
close_mw_cel(2)  
copy_mw_cel(2)  
current_mw_cel(2)  
get_mw_cels(2)  
open_mw_cel(2)  
remove_mw_cel(2)  
rename_mw_cel(2)  
save_mw_cel(2)  
mw_cel_collection(2)
```

create_mw_lib

Creates a Milkyway library.

SYNTAX

```
status create_mw_lib
[-technology technology_file_name]
[-plib plib_file_name]
[-hier_separator sep]
[-bus_naming_style style]
[-mw_reference_library lib_list]
[-reference_control_file rc_file_name]
[-open]
libName
```

Data Types

<i>technology_file_name</i>	string
<i>plib_file_name</i>	string
<i>sep</i>	character
<i>style</i>	string
<i>lib_list</i>	string
<i>rc_file_name</i>	string
<i>libName</i>	string

ARGUMENTS

-technology *technology_file_name*
Specifies the name of the technology file for the newly created Milkyway library. This option and **-plib** are mutually exclusive.

-plib *plib_file_name*
Specifies the name of the plib file for the newly created Milkyway library. Valid file is of .plib format or .pdb format. This option and **-technology** are mutually exclusive. If routing directions in the .plib file are used to set preferred routing directions for the design library, you must specify the reference libraries by using the **-reference_control_file** or **-mw_reference_library** option.

-hier_separator *sep*
Specifies the character to be used as a separator in hierarchical names. The default hierarchical separator is slash (/).

-bus_naming_style *style*
Specifies the bus naming style for the library. The default bus naming style is [%d].

-mw_reference_library *lib_list*
Specifies a list of reference libraries to be used for the new library.

-reference_control_file *rc_file_name*
Specifies the reference control file used to set reference library

information for the new library.

-open

Opens the library after creation.

libName

Specifies the name of the Milkyway library to be created.

DESCRIPTION

This command creates a Milkyway library.

The **-technology** and **-plib** options are mutually exclusive, and at least one of them must be specified.

By default, the newly created Milkyway library is not open in the current session. To manipulate it, first run the **open_mw_lib** command. The library can also be opened by using the **-open** option, but to makes the scripts more reusable, you should use **open_mw_lib**.

All strings stored in this library, as well as in designs belonging to it, are case-sensitive, and all string operations are performed as case-sensitive.

A status indicating success or failure is returned.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates a Milkyway design library with a technology file and bus naming style as [%d]. The hierarchical separator is the default value slash (/).

```
prompt> create_mw_lib design -technology test.tf \  
-bus_naming_style {[%d]}  
1
```

SEE ALSO

close_mw_lib(2)
open_mw_lib(2)
create_lib_track(2)

create_net

Creates nets in the current design or its subdesign.

SYNTAX

```
status create_net  
[-power | -ground | -tie_high | -tie_low]  
net_list
```

Data Types

net_list list

ARGUMENTS

-power

Creates a power net.

By default, the tool creates a signal net.

The **-power**, **-ground**, **-tie_high**, and **-tie_low** options are mutually exclusive; you can specify only one.

-ground

Creates a ground net.

By default, the tool creates a signal net.

The **-power**, **-ground**, **-tie_high**, and **-tie_low** options are mutually exclusive; you can specify only one.

-tie_high

Creates a tie-high net.

By default, the tool creates a signal net.

The **-power**, **-ground**, **-tie_high**, and **-tie_low** options are mutually exclusive; you can specify only one.

-tie_low

Creates a tie-low net.

By default, the tool creates a signal net.

The **-power**, **-ground**, **-tie_high**, and **-tie_low** options are mutually exclusive; you can specify only one.

net_list

Specifies the names of the created nets. If you specify a hierarchical net name, the net is created in the specified instance. The net name must be unique within the current design or the subdesign where it is created. If you use a hierarchical net name, the parent instance must be unique; it cannot be an instance of a multiply-instantiated design. This is a required option.

DESCRIPTION

The **create_net** command creates new net objects in the current design or its subdesign, based on the *net_list* argument. The **create_net** command creates only

scalar (single bit) nets.

By default, this command creates signal nets.

To create a power net, use the **-power** option. To create a ground net, use the **-ground** option. To create a tie-high net, use the **-tie_high** option. To create a tie-low net, use the **tie_low** option. These four options are optional and mutually exclusive.

You can use this command to create the default tie-off nets, SNPS_LOGIC0 and SNPS_LOGIC1, if these nets do not already exist in the design. The type for these nets is determined by name only. SNPS_LOGIC0 and SNPS_LOGIC1 are special names in IC Compiler. They can only be of the type tie_low and tie_high, respectively.

Nets connect pins and ports in a design. When you create nets with **create_net**, they are not connected. To connect signal nets, use the **connect_net** command. To connect power and ground nets or tie-off nets to the power or ground rails, use the **derive_pg_connections** command.

To remove nets from the current design, use the **remove_net** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses **create_net** to create net objects in the current design:

```
prompt> create_net {N1 N2 N3 N4}
prompt> get_nets *
```

The following example uses **create_net** to create net objects in the mid subdesign. Note that mid1 is an instance of the mid design. The mid design must be unique for the **create_net** command to succeed.

```
prompt> create_net {mid1/N1 mid1/N2 mid1/N3 mid1/N4}
prompt> get_nets mid1*
```

The following example uses **create_net** to create a power net:

```
prompt> create_net -power VDD
prompt> get_nets -all VDD
```

SEE ALSO

all_connected(2)
connect_net(2)
derive_pg_connection(2)
disconnect_net(2)
remove_net(2)

create_net_shape

Creates a new net shape.

SYNTAX

```
collection create_net_shape
[-type wire | path | rect | poly]
-origin point
    | -points list_of_points
    | -bbox rect
    | -boundary boundary
[-length real]
[-width real]
[-path_type square | round | extend_half_width | octagon]
-layer layer
-net net_name
[-vertical]
[-route_type route_type]
[-datatype int]
[-avoid_short_segment]
```

Data Types

<i>point</i>	point
<i>list_of_points</i>	list of points
<i>rect</i>	rectangle
<i>boundary</i>	polygon
<i>real</i>	real
<i>layer</i>	collection
<i>net_name</i>	string
<i>route_type</i>	string
<i>int</i>	integer

ARGUMENTS

-type wire | path | rect | poly

Specifies the net shape type.

If you do not specify this option, the default net shape depends on how you specify the net shape. The net shape type is determined by using the following rules, in order of precedence:

1. If you use the **-origin** option, the net shape is **wire**.
The wire is horizontal, unless you also specify the **-vertical** option.
2. If you use the **-bbox** option, the net shape is **wire** if you also use the **-path_type**, **-route_type**, or **-vertical** options. If you do not use any of these additional options, the net shape is **rect**.
3. If you use the **-points** option, the net shape is **path**.
4. If you use the **-boundary** option, the net shape is **poly**.

-origin *point*

Specifies the origin of the net shape for wires.

When you specify this option, you must also specify the **-length** and **-width** options.

The **-origin**, **-points**, **-bbox**, and **-boundary** options are mutually exclusive. You must specify one of these options.

-points *list_of_points*

Specifies the point sequence of the net shape for paths.

The **-origin**, **-points**, **-bbox**, and **-boundary** options are mutually exclusive. You must specify one of these options.

-bbox *rect*

Specifies the bounding box of the net shape. You must specify the lower-left and upper-right corners of the rectangle by using the following syntax: {{llx lly} {urx ury}}.

The **-origin**, **-points**, **-bbox**, and **-boundary** options are mutually exclusive. You must specify one of these options.

-boundary *boundary*

Specifies the boundary of the net shape. You must specify at least three nonoverlapping points.

The **-origin**, **-points**, **-bbox**, and **-boundary** options are mutually exclusive. You must specify one of these options.

-length *real*

Specifies the length of the net shape for wires in user units.

This option is required when you specify the **-origin** option and ignored otherwise.

-width *real*

Specifies the width of the net shape for wires in user units.

This option is required when you specify the **-origin** or **-points** option and ignored otherwise.

-path_type *square | round | extend_half_width | octagon*

Specifies the alignment type of a wire or path end.

Possible values are:

<i>square</i>	- square, no extension
<i>round</i>	- round, half width extension
<i>extend_half_width</i>	- square, half width extension
<i>octagon</i>	- octagon, half width extension

The default is **square**.

-layer *layer*

Specifies the layer for the net shape. You can specify the layer by using the layer name from the technology file or by using the **get_layers** command.

This is a required option.

-net *net_name*

Specifies the net associated with the net shape.

This is a required option.

-vertical

Indicates that the wire is vertical.

By default, the net shape is horizontal.

-route_type route_type

Specifies the route type of the net shape. Valid values for this option are

user_enter	User entered
signal_route, signal_route_detail	Detail routing
signal_route_global	Global routing
pg_ring	Power or ground (PG) ring
pg_strap	PG strap
pg_macro_io_pin_conn	PG net that connects to the PG pin of an I/O pad cell or a macro cell
pg_std_cell_pin_conn	PG net that connects to the PG pin of a standard cell
clk_ring	Clock ring
clk_strap	Clock strap
clk_zero_skew_route	Clock zero skew route
bus	Bus
shield, shield_fixed	Fixed shield
shield_dynamic	Dynamic shield
fill_track, clk_fill_track	Fill track

By default, the route type is **signal_route**.

-datatype int

Specifies the data type number.

By default, the data type is 0.

-avoid_short_segment

Specifies that segments shorter than a half width should be avoided for paths.

DESCRIPTION

This command creates a shape object that is attached to a net and returns a collection containing the created object.

The valid shape types are:

Wire (horizontal or vertical)
Path
Rectangle
Polygon

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates a wire net shape.

```
prompt> create_net_shape -type wire -net VSS \  
-origin {0 0} -length 10 -width 2 -layer M1
```

The following example creates a path net shape.

```
prompt> create_net_shape -type path -net {VSS} \  
-path_type extend_half_width -layer M3 \  
-points {{845 715} {845 710} {860 710}} -width 0.230
```

The following example creates a rectangular net shape.

```
prompt> create_net_shape -type rect -net {CLOCK_Net27} \  
-bbox {{76 110} {82 115}} -layer METAL1
```

SEE ALSO

```
create_user_shape(2)  
create_placement_blockage(2)  
create_route_guide(2)  
create_via(2)  
create_terminal(2)  
create_text(2)
```

create_open_drc_error

Creates an error record for an error type in the open error type class.

SYNTAX

```
collection create_open_drc_error
-error_view mw_error_view
-type error_type
[-status error | fixed | ignored | waived]
[-info description]
[-net net]
[-details details]
```

Data Types

<i>mw_error_view</i>	list or collection of one item
<i>error_type</i>	string
<i>description</i>	string
<i>net</i>	list or collection of one item
<i>details</i>	list

ARGUMENTS

-error_view *mw_error_view*
Specifies the error view in which to create the error record. You may specify only a single error view; specifying more than one error view causes an error. This is a required option.

-type *error_type*
Specifies the error type for the new error object. You can specify the error type by using the id or name of the error type. The specified error type must be in the open error type class. This is a required option.

-status error | fixed | ignored | waived
Sets the current status of this error.
If you do not specify this option, the status is set to **error**.

-info *description*
Provides a full description of this error.
If you do not specify this option, the description is left empty.

-net *net*
Specifies the net in which the open was found. You can specify the net by providing a collection containing the net or by specifying the net name. You cannot use wildcards in the net name.
If you do not specify this option, no net is associated with the open error.

-details *details*
Specifies the error details, including shapes and layers.
To specify the error details, use the following syntax for the *details* argument:

```

details: {num_details detail+}
detail: { {node node_number }
          num_shapes {shape+}}
shape: {[{layer layer}]
        num_rectangles {rectangle+}}
rectangle: {llx lly urx ury}

```

An open type error is described by two or more disjoint nodes, where a node is a set of connected net shapes. If a node contains net shapes on different layers, you must specify a shape construct for each layer. You assign a number to each node for identification purposes.

This argument is optional; however, an error with no details is virtually useless. You can also add details by using the **add_open_drc_error_detail** command.

DESCRIPTION

This command creates a new error object of the specified error type, which must be associated with the open error type class. Every error object must be associated with one and only one error type. The error type association is made when the error object is created and cannot be changed. Therefore, you must create an error type before you can create error objects of that type.

The open error type class is for violations where routed shapes for a net are not fully connected. An open-type error is described by disjoint nodes, where each node is a set of connected net shapes. To fully describe the violation, you can also provide a description and the net on which the open occurs.

If successful, the command returns a collection containing the new error object.

EXAMPLES

The following example opens an error view named mydesign_mydrc.err, creates an error type named Open Net in the open error type class, and then creates an error record of the Open Net type:

```

prompt> set cellId [open_mw_cel -not_as_current mydesign_mydrc.err]
{mydesign_mydrc}
prompt> set typeId [create_drc_error_type -name "Open Net" \
    -class "open" -info "There are opens in the net" \
    -error_view $cellId]
1024
prompt> create_open_drc_error -type $typeId -error_view $cellId \
    -details {2
    {{node 1}
    1 {{layer METAL2} 1 {574.0700 430.3600 578.9700 430.3600}}}
    {{node 2}
    1 {{layer METAL4} 1 {574.0700 436.3600 574.0700 453.3600}}}}
{1280}

```

SEE ALSO

add_open_drc_error_detail(2)
create_drc_error_type(2)

```
list_drc_error_types(2)
report_drc_error_type(2)
get_drc_errors(2)
remove_drc_error(2)
```

create_open_locator_drc_error

Creates an error record for an error type in the openlocator error type class.

SYNTAX

```
collection create_open_locator_drc_error
-error_view mw_error_view
-type error_type
-point1 point_spec
-point2 point_spec
[-status error | fixed | ignored | waived]
[-info description]
[-net net]
[-referenced_drc_error drc_error]
```

Data Types

<i>mw_error_view</i>	list or collection of one item
<i>error_type</i>	string
<i>point_spec</i>	list
<i>description</i>	string
<i>net</i>	list or collection of one item
<i>drc_error</i>	list or collection of one item

ARGUMENTS

-error_view *mw_error_view*
Specifies the error view in which to create the error record. You may specify only a single error view; specifying more than one error view causes an error. This is a required option.

-type *error_type*
Specifies the error type for the new error object. You can specify the error type by using the id or name of the error type. The specified error type must be in the openlocator error type class. This is a required option.

-point1 *point_spec*
Specifies the first endpoint of the flyline describing the open locator. You specify the endpoint in the following format:
point: {[{node *node_number*}]
 [{{layer *layer*}}] x y}
At a minimum, you must specify the x- and y-coordinates of the endpoint. You can also specify the associated node number and layer. The node number specifies the node of the referenced open type error in which the endpoint is located, if any. The node number is ignored unless you specify the **-referenced_drc_error** option. This is a required option.

-point2 *point_spec*
Specifies the second endpoint of the flyline describing the open locator. You specify the endpoint in the following format:

create_open_locator_drc_error

```
point: {[{node node_number}]
        [{layer layer}] x y}
```

At a minimum, you must specify the x- and y-coordinates of the endpoint. You can also specify the associated node number and layer. The node number specifies the node of the referenced open type error in which the endpoint is located, if any. The node number is ignored unless you specify the -**referenced_drc_error** option.

This is a required option.

-status error | fixed | ignored | waived

Sets the current status of this error.

If you do not specify this option, the status is set to **error**.

-info *description*

Provides a full description of this error.

If you do not specify this option, the description is left empty.

-net *net*

Specifies the net in which the open was found. You can specify the net by providing a collection containing the net or by specifying the net name. You cannot use wildcards in the net name.

If you do not specify this option, no net is associated with the openlocator error.

-referenced_drc_error *drc_error*

Specifies the error object id for the open type error associated with this openlocator error. Because openlocator type errors describe a possible connection for an open type error, you can use this option to associate the openlocator error with the open error for the same net.

DESCRIPTION

This command creates a new error object of the specified error type, which must be associated with the openlocator error type class. Every error object must be associated with one and only one error type. The error type association is made when the error object is created and cannot be changed. Therefore, you must create an error type before you can create error objects of that type.

The openlocator error type class is for violations that describe opens in nets, not as disjoint nodes of connected net shapes, but as the potential connection flyline for an open. For this reason, where there exists a single open type error object for a net with one or more opens, there are *n* openLocator type error objects for a net with *n* opens (with *n*+1 disjoint nodes).

If successful, the command returns a collection containing the new error object.

EXAMPLES

The following example opens an error view named mydesign_mydrc.err, creates an error type named "Open Connection", and then creates an error record of the "Open Connection" error type:

```
prompt> set cellId [open_mw_cel -not_as_current mydesign_mydrc.err]
```

```

{mydesign_mydrc}
prompt> set openLocatorTypeId [create_drc_error_type \
    -name "Open Connection" -class "OpenLocator" \
    -info "A connection flyline for an open net" -error_view $cellId]
1024
prompt> set openTypeId [create_drc_error_type \
    -name "Open Net" -class "Open" \
    -info "Opens in the net" -error_view $cellId]
1025
prompt> set openErrId [create_open_drc_error
    -type $typeId -error_view $cellId \
    -net [get_nets myNetA] \
    -details {2
        {{node 1} 1 {{layer METAL2}
            1 {574.0700 430.3600 578.9700 430.3600}}}
        {{node 2} 1 {{layer METAL4}
            1 {574.0700 436.3600 574.0700 453.3600}}}]
{1280}
prompt> create_open_locator_drc_error -type $openLocatorTypeId \
    -error_view $cellId \
    -net [get_nets myNetA] \
    -referenced_drc_error $openErrId \
    -point1 {{node 1} {layer METAL2} 574.0700 430.3600} \
    -point2 {{node 2} {layer METAL4} 574.0700 436.3600}
{1281}

```

SEE ALSO

```

create_open_drc_error(2)
create_drc_error_type(2)
list_drc_error_types(2)
report_drc_error_type(2)
get_drc_errors(2)
remove_drc_error(2)

```


create_operating_conditions

Creates a new set of operating conditions in a library.

SYNTAX

```
int create_operating_conditions  
-name name -library library_name  
-process process_value  
-temperature temperature_value  
-voltage voltage_value  
[-tree_type tree_type]  
[-calc_mode calc_mode]  
[-rail_voltages rail_value_pairs]
```

Data Types

<i>name</i>	string
<i>library_name</i>	string
<i>process_value</i>	float
<i>temperature_value</i>	float
<i>voltage_value</i>	float
<i>tree_type</i>	string
<i>calc_mode</i>	string
<i>rail_value_pairs</i>	list

ARGUMENTS

-name *name*
Specifies the name of the new set of operating conditions.

-library *library_name*
Specifies the name of the library for the new operating conditions.

-process *process_value*
Specifies the process scaling factor for the operating conditions. Allowed values are 0.0 through 100.0.

-temperature *temperature_value*
Specifies the temperature value, in degrees Celsius, for the operating conditions. Allowed values are -300.0 through +500.0.

-voltage *voltage_value*
Specifies the voltage value for the operating conditions. Allowed values are 0.0 through 1000.0.

-tree_type *tree_type*
Specifies the tree type for the operating conditions. Allowed values are *balanced_tree* (the default), *best_case_tree*, or *worst_case_tree*. The tree type is used to estimate interconnect delays by providing a model of the RC tree.

-calc_mode *calc_mode*

For use only with DPCM libraries. Specifies the DPCM delay calculator mode for the operating conditions. Allowed values are *unknown* (the default), *best_case*, *nominal*, or *worst_case*. If you use the default value, the *worst_case* value will be used during analysis. If **-rail_voltages** is specified, the command sets corresponding (*worst_case*, *nominal*, and *best_case*) voltage values.

-rail_voltages *rail_value_pairs*

Specifies a list of name-value pairs that defines the voltage for each specified rail. The name is one of the rail names defined in the library; the value is the voltage to be assigned to that rail. By default, rail voltages are determined by the values given to them in the library. Use this option to override the default voltages for the specified rails.

DESCRIPTION

The **create_operating_conditions** command creates a new set of operating conditions in the specified library. A technology library contains a fixed set of operating conditions. This command allows you to create new and additional operating conditions.

To see the operating conditions defined for a library, use **report_lib**.

To set operating conditions on the current design, use **set_operating_conditions**.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates a new set of operating conditions called WC_CUSTOM in the library "tech_lib", specifying new values for process, temperature, voltage, parameter1, and tree type. By default, rail voltages remain as defined in the library.

```
prompt> create_operating_conditions -name WC_CUSTOM \  
-library tech_lib -process 1.2 -temperature 30.0 -voltage 2.8 \  
-parameter1 1.2 -tree_type worst_case_tree
```

The following example creates a new set of operating conditions called OC3 in the library "IBM_CMOS5S6_SC", specifying new values for process, temperature, voltage, and rail voltages for rails VTT and VDDQ. By default, the tree type *balanced_tree* is used.

```
prompt> create_operating_conditions -name OC3 \  
-lib IBM_CMOS5S6_SC -proc 1.0 -temp 100.0 -volt 4.0 \  
-rail_voltages {VTT 3.5 VDDQ 3.5}
```

SEE ALSO

`report_lib(2)`
`set_operating_conditions(2)`

create_pad_rings

Creates pad rings for pins in boundary pads.

SYNTAX

```
integer create_pad_rings  
[-create all | pg | specified_net]  
[-nets {collection_of_nets}]  
[-route_pins_on_layer number_or_name]  
[-min_shrink_routing_boundaries_for_all_boundary_pads distance]  
[-max_shrink_routing_boundaries_for_all_boundary_pads distance]  
[-undo]
```

Data Types

<i>collection_of_nets</i>	collection
<i>number_or_name</i>	string
<i>distance</i>	distance

ARGUMENTS

-create all | pg | specified_net

Determines which nets to route. Valid arguments are **all**, **pg**, and **specified_net**. Using **all** routes all nets. Using **pg** routes only power and ground nets. Using *specified_net* routes only the nets specified in the **-nets** option. The default is **all**.

-nets {collection_of_nets}

Specifies the nets to create pad rings. If you specify more than one net in a collection they must be separated by a space. This option is required if you are using the **-create** option with the **specified_net** value.

-route_pins_on_layer number_or_name

Specifies the layer on which to connect pins. The *number_or_name* value can be specified as either the layerNumber or maskName from the technology file. By default, the command considers all metal layers.

-min_shrink_routing_boundaries_for_all_boundary_pads distance

Specifies the minimum amount by which to shrink routing boundaries of all boundary pads to make the pad pins accessible for routing. Enter a positive number to shrink the routing boundaries or a negative number to expand them. A boundary pad is a pad "near enough" to the top-cell boundary. Routing boundaries are the two sides of the boundary pads that can be touched by the generated pad ring or rings. Note that the command does not change the physical size of the routing boundary pads. It accomplishes a shrinking of the boundary pads by changing how far into the pad the program looks to find pins. It assumes the boundary is rectangular. The default is **0.0**.

-max_shrink_routing_boundaries_for_all_boundary_pads distance

Specifies the maximum amount by which to shrink routing boundaries of all boundary pads to make the pad pins accessible for routing. See the **-min_shrink_routing_boundaries_for_all_boundary_pads** option, above, for how

this option works. The default is **0.0**.

-undo

Removes pad rings created by the last **create_pad_rings** run.

DESCRIPTION

This command creates pad rings for pins only in boundary pads. The generated rings do not cover pad pins. The width of the pad ring is the smaller width of the two pins for the wire being created. You can change the internal application of DRC rules by using the **set_preroute_drc_strategy** command.

Prerequisites: Before using this command, use the **connect_pg_nets** command to specify port-to-net connections for power and ground nets if they are not part of the netlist. Wires are created only between two pins on the same net and on the same layer.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates pad rings with VDDC and VSSC nets, for pins on layer M3, shrinking boundaries by a minimum distance of 5.0 and a maximum distance of 6.0.

```
prompt> create_pad_rings -create specified_net -nets {VDDC VSSC} \  
-route_pins_on_layer M3 \  
-min_shrink_routing_boundaries_for_all_boundary_pads 5.0 \  
-max_shrink_routing_boundaries_for_all_boundary_pads 6.0
```

SEE ALSO

```
create_power_straps(2)  
create_preroute_vias(2)  
create_rectangular_rings(2)  
preroute_instances(2)  
preroute_standard_cells(2)  
report_preroute_drc_strategy(2)  
set_preroute_drc_strategy(2)  
verify_pg_nets(2)
```

create_partition

Creates Verilog files corresponding to the partitions identified in the partition editor.

SYNTAX

```
status create_partition
[-read_verilog]
[-initialize_floorplan]
[-output_dir dir_name]
[-verbose]
```

Data Types

dir_name string

ARGUMENTS

-read_verilog

After the design is partitioned into verilog files, this causes the top level design Verilog file to be read into the system.

-initialize_floorplan

After the design is partitioned into verilog files, this causes the top level design Verilog file to be read into the system, and a basic top-level floorplan to be generated. This option is only valid while using the -read_verilog option.

-output_dir *dir_name*

When designs are created for the physical partitions, Verilog files for the top level and each of the subblocks are created. This option specifies the output directory in which these files are placed. It defaults to the directory **sub_blocks** in the current directory.

-verbose

The **create_partition** command outputs more detailed messages.

DESCRIPTION

This command is used to create Verilog files that represent the top level and subblocks in the design. In addition, this command can read in the top level Verilog for the design, creating a top-level database for this design. It can also create a basic top-level floorplan.

The Verilog files that are output from this command are subsets (substrings) of the Verilog text from input Verilog (from the **read_partition** command). In other words, if the input is 100 percent Verilog compliant, then the output is 100 percent Verilog compliant. And, if the input is not 100 percent Verilog compliant, then the output is equally not 100 percent Verilog compliant.

The final form is a connectivity-based partitioning that uses the top-level net

create_partition

connectivity within the design to create a partitioning. While the two simple forms of autopartitioning always follow the logic hierarchy, the connectivity-based autopartitioning has an option to break the logic hierarchy through grouping and ungrouping if it will result in better partitions. This version is the one to choose in cases where the RTL is structured in an unusual way that could benefit from restructuring for the physical implementation. This form also has options to choose balancing instance counts or area as an underlying goal.

EXAMPLES

This example shows partitioning a design and outputting the Verilog files.

```
prompt> read_partition -input_files {myDesign.v}
Partitioner Name: pfe0
Loading verilog file ../ICC_DATA/Netlist/myDesign.v
MYDESIGN: 73958 instances of 134 modules.
1
prompt> set_partition_data -auto_partition instance_count
After partitioning, list of sub-blocks is BLENDER_0 BLENDER_1 BLENDER_2 BLENDER_3
1
prompt> create_partition
1
```

SEE ALSO

```
create_partition(2)
set_partition_data(2)
list_partition_data(2)
initialize_floorplan(2)
```

create_pg_network

Creates or changes a power or ground network that connects multiple hierarchical cells.

SYNTAX

```
status create_pg_network  
-net net  
    | -create_net net_name [-power | -ground]  
cell_list
```

Data Types

<i>net</i>	string
<i>net_name</i>	string
<i>cell_list</i>	list

ARGUMENTS

-net *net*
Specifies an existing power or ground (PG) net for network creation. The net must be a scalar (single bit) net and can be at any level of the hierarchy. When you specify this option, the tool derives the PG net type from the existing net. Therefore, you should not specify the **-power** or **-ground** option with the **-net** option.
The base name of the net is used to create the nets in the specified hierarchy instances. For example, if the specified net is named "blk1/VDDC", its base name is "VDDC" and the nets and ports created in the specified hierarchy instances are named "VDDC".
This option and the **-create_net** option are mutually exclusive; you must one of these options.

-create_net *net_name*
Specifies the name of the power or ground net that is created to connect the specified hierarchical cell instances.
This option and the **-net** option are mutually exclusive; you must one of these options.

-power
Specifies that the created net is a power net.

-ground
Specifies that the created net is a ground net.

cell_list
Specifies a list of hierarchical cell instances to which the power or ground network is connected.
If any object in the list is not found or is not a hierarchical cell, the tool issues an error message and stops.
This is a required argument.

DESCRIPTION

This command creates a new hierarchical power or ground connection or adds cells to an existing power or ground network. Existing power or ground nets include nets created by the **create_net** command and nets imported by the **read_verilog** command. This command makes connections only to hierarchical cell instances and does not connect to leaf pins.

For each specified hierarchical cell, a new net and a new port are created. The port and net names are the same. The name is either specified by the **-create_net** option or is the same as the base name of the net specified by the **-net** option.

The tool issues an error message and stops if the port or net to be created already exists in a given cell.

The direction of created port is "inout".

EXAMPLES

The following example uses the **create_pg_network** command to expand an existing VDD connection to a list of specified cells. The **all_connected** command is used to verify the net connections before and after running the **create_pg_network** command.

```
prompt> all_connected [get_pin -all m2/VDD]
Warning: No pin objects matched 'm2/VDD' (SEL-004)
prompt> create_pg_network -net VDD "m2/m21 m1/m11"
prompt> all_connected [get_pin -all m2/VDD]
{VDD}
```

The following example uses the **create_pg_network** command to add an existing VDD connection to a single cell. The **all_connected** command is used to verify the net connections before and after running the **create_pg_network** command.

```
prompt> create_pg_network -net VDD "m2/m21"
prompt> get_nets -all -hierarchical VDD
{m2/VDD m2/m21/VDD}
prompt> all_connected [get_pin -all m2/m21/VDD]
{m2/m21VDD}
```

The following example uses the **create_pg_network** command to create a new VDDC connection to a list of specified cells. The **all_connected** command is used to verify the net connections before and after running the **create_pg_network** command.

```
prompt> all_connected [get_pin -all m2/VDDC]
Warning: No pin objects matched 'm2/VDDC' (SEL-004)
prompt> create_pg_network -create_net VDDC -type power "m2/m21 m1/m11"
prompt> all_connected [get_pin -all m2/VDDC]
{VDDC}
```

SEE ALSO

[create_net\(2\)](#)
[create_port\(2\)](#)
[current_design\(2\)](#)

```
connect_net(2)
disconnect_net(2)
remove_net(2)
```

create_physical_bus

Creates a physical bus with a list of nets.

SYNTAX

```
collection create_physical_bus  
name  
-nets net_list  
[-sort {ascending | descending | none}]  
[-delimiter delimiter]  
[-right_precedence]  
[-quiet]
```

Data Types

<i>name</i>	string
<i>net_list</i>	list
<i>delimiter</i>	string

ARGUMENTS

name

Specifies physical bus name.

-nets *net_list*

Specifies a list of nets to be members of newly created physical bus.

-sort {ascending | descending | none}

Specifies the sort mode. The valid values are: **ascending**, **descending**, and **none**.

If you specify *-sort ascending*, nets will be sorted by their names alphanumerically, from lowest to highest. If you specify *-sort descending*, nets will be sorted by their names alphanumerically, from highest to lowest. If you specify *-sort none*, no sort will be applied. By default *sort_mode* is *none*.

-delimiter *delimiter*

Specifies the delimiter characters to use for the names of nets.

The valid values of *delimiter* are: **[]**, **{}**, **<>**, **()**, **_** and **:**. By default *delimiter* is **[]**.

When nets are sorted, characters inside *delimiter* will be regarded as index. For example, Two nets Net[2] and Net[10] with *-sort ascending*, Net[2] will be in front of Net[10].

-right_precedence

Indicates that sorting of index inside *delimiter* is from right to left. By default the order is from left to right.

For example, there are six nets with *-sort ascending* and no option **-right_precedence**, the ordered list will be:

Net[0][1], Net[0][2], Net[1][1], Net[1][2], Net[2][1], Net[2][2],

While with *-sort ascending* and option **-right_precedence**, the ordered list will be:

Net[0][1], Net[1][1], Net[2][1], Net[0][2], Net[1][2], Net[2][2],

-quiet

Suppresses warning and error messages. Syntax error messages are not suppressed.

DESCRIPTION

The **create_physical_bus** command creates a new physical bus.

Physical bus is an ordered collection of flat nets, which will be used in bus driven routing.

If succeeded, this command would return a collection containing the new created physical bus. If failed, this command would return an empty string.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates a physical bus of name "bus1", and then use command **get_nets -of_object \$bus1** to see the elements in the physical bus.

```
prompt> set bus1 [create_physical_bus bus1 -sort ascending \  
-nets [get_nets UPC_DATA*]]  
{bus1}
```

```
prompt> get_nets -of_object $bus1  
{UPC_DATA[0] UPC_DATA[1] UPC_DATA[2] UPC_DATA[3] UPC_DATA[4] UPC_DATA[5] UPC_DATA[6]  
] UPC_DATA[7] UPC_DATA[8] UPC_DATA[9] UPC_DATA[10] UPC_DATA[11]}
```

SEE ALSO

get_nets(2)
get_physical_buses(2)
remove_physical_bus(2)
report_physical_bus(2)
update_physical_bus(2)

create_physical_buses_from_patterns

Automatically groups nets into buses based on their net name and group size criteria.

SYNTAX

```
status create_physical_buses_from_patterns
[-net_name_prefix string]
[-net_order ascending | descending]
[-minimum_nets integer]
[-maximum_nets integer]
[-divider integer]
[-no_braces]
[-no_brackets]
[-no_angle_brackets]
[-no_underlines]
[-no_colons]
[-no_parentheses]
```

ARGUMENTS

-net_name_prefix *string*
Specifies net name prefix for pattern matching.

-net_order ascending | descending
Specifies net sorting order (Default is ascending).

-minimum_nets *integer*
This is bus size constraint option. It specifies minimum number of nets per bus (default is 8).

-maximum_nets *integer*
This is bus size constraint option. It specifies maximum number of nets per bus (default is 128).

-divider *integer*
This is bus size constraint option. It specifies allowed numbers of signals in the bus (default is 1).

-no_braces
Don't use braces in pattern matching

-no_brackets
Don't use brackets in pattern matching

-no_angle_brackets
Don't use angle brackets in pattern matching

-no_underlines
Don't use underlines in pattern matching

`-no_colons`
Don't use colons in pattern matching

`-no_parentheses`
Don't use parentheses in pattern matching

DESCRIPTION

This command groups automatically nets into buses based on their net name and group size criteria. Defining buses with a net name pattern saves time for designs with many buses, because nets are usually grouped by their names. The net name pattern specification is done implicitly with set of options. User can specify the net name pattern where the net bit order and index are to be extracted. After the nets are grouped, each net group has a specified size. Only net groups with specified sizes are automatically defined as buses. The following net name patterns are supported: `..[i]..`, `..{i}..`, `..<i>..`, `..(i)..`, `.._i..`, `..:i..`.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example searches then nets with given name prefix and creates buses.

```
prompt> create_physical_buses_from_patterns -net_name_prefix ACMD  
1
```

SEE ALSO

`create_physical_bus(2)`
`get_physical_buses(2)`

create_pin_guide

Creates a pin guide for a plan group, or soft macro, or the current cell to constrain terminals generated by pin assignment to a specified bounding box.

SYNTAX

```
pin_guide create_pin_guide
{-bbox bounding_box_rect
  | -boundary rectilinear_boundary}
[-parents soft_macro_or_plan_group]
[-name pin_guide_name]
[-exclusive]
objects
```

Data Types

<i>bounding_box_rect</i>	x1 y1 x2 y2
<i>rectilinear_boundary</i>	boundary points
<i>soft_macro_or_plan_group</i>	collection
<i>pin_guide_name</i>	string
<i>objects</i>	collection

ARGUMENTS

-bbox *bounding_box_rect*

Specifies the bounding box of a rectangular pin guide.
Constrains pins associated with the pin guide to the region where this bounding box overlaps the edges of the parent plan group or soft macro.
The **-boundary** and **-bbox** options are mutually exclusive; use only one.
If the bounding box does not overlap an edge, it is ignored.
The unit is in micron.

-boundary *rectilinear_boundary*}

Specifies the boundary points of a rectilinear pin guide.
This option serves the same purpose as the **-bbox** option, except that it allows you to specify a rectilinear pin guide.
The *boundary* argument is a series of points that defines the boundary of the pin guide.
The **-boundary** and **-bbox** options are mutually exclusive; use only one.
The unit is in micron.

-parents *soft_macro_or_plan_group*

Specifies the parent plan group or soft macro object with which to associate the pin guide.
For virtual flat flow this will usually be a plan group. For hierarchical flow this will usually be a soft macro.
By default, the command creates the pin guide for the current cell.

-name *pin_guide_name*

Specifies the name of the pin guide to be created. If this option is not used a default name will be given to the pin guide.

-exclusive

If used, no terminal or soft macro pin can be placed within the pin guide except for those specified to be placed inside the pin guide at the time of its creation. If nets are specified at the time of pin guide creation, then the only pins allowed inside the pin guide are those that are connected to the nets or the feed through pins derived from the global routings of these nets.

objects

Specifies a collection of nets for a parent plan group or a collection of pins for a parent soft macro.
For a parent plan group, the terminals generated by pin assignment for these nets are constrained to the pin guide bounding box.
For a parent soft macro the terminals generated by pin assignment for these pins are constrained to the pin guide bounding box.

RETURNS

pin_guide

Specifies a created pin guide object.

DESCRIPTION

This command creates a pin guide for a plan group or soft macro to constrain terminals generated by pin assignment to a specified bounding box.

Pin guide shapes may be used to define feedthroughs, that is, added ports that do not correspond to existing logical ports, to route a net through a plan group or soft macro. When used in this way, the collection of objects supplied to the command should consist only of nets.

If a parent block of the guide contains a connection to the net, and the guide crosses the block boundary more than once, a port is associated with each crossing. One of the ports will correspond to the original port. Any additional ports created will be feedthrough ports.

If a parent block of the guide does not contain any connection to the net, then the pin guide must cross the block boundary in at least two separate locations. The net will be forced to route through this block and will have a feedthrough port in each of the areas where the guide crosses the block boundary. If a guide only crosses a non-connected block in one place, it is considered an error and this portion of the guide will be ignored. The guide should be a single contiguous shape (i.e. not disjoint), and can be either rectangular or rectilinear.

The user should take care that the guide represents a reasonable path for the feedthrough routing. For any guide, the number of crossings of the block boundary must be less than or equal to the number of net targets outside the block. For example, if the guide is an irregular shape that crosses the block boundary in three disjoint areas, then there must be at least three separate targets (ports) on the net outside the block, so that each new feedthrough port can be connected to at least one external target. If there are fewer external targets, say two in this case, it is considered an error and this portion of the guide will be ignored.

create_pin_guide

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates a pin guide with the name `abc` for all nets matching a pattern on a plan group named `mem1`.

```
prompt> set pin_guide [create_pin_guide -bbox {0 0 100 100}\  
-name abc \  
-parents mem1 [get_nets *clk*]
```

The following example creates an L-shape pin guide for all nets matching a pattern on the plan group `mem1`.

```
prompt> set pin_guide [create_pin_guide -boundary \  
{0 0} {100 0} {100, 30} {30 30} {30 100} {0 100} {0 0}} \  
-parents mem1 [get_nets *clk*]
```

SEE ALSO

```
get_pin_guides(2)  
report_pin_guides(2)
```

create_placement

Performs coarse placement on the current design.

SYNTAX

```
int create_placement
[-effort low | medium | high]
[-quick]
[-timing_driven]
[-congestion [-congestion_effort low | medium | high]]
[-check_only]
[-num_cpus number_of_cpus]
[-mpc]
[-ignore_scan]
```

Data Types

number_of_cpus integer

ARGUMENTS

- effort low | medium | high
Specifies the CPU effort level for coarse placement. The default effort level is **medium**.
- quick
Enables fast coarse placement suitable for floorplan exploration.
- timing_driven
Enables timing-driven placement mode.
- congestion
Enables congestion-driven placement mode.
- congestion_effort low | medium | high
Specifies the effort level for congestion mode. The default effort level is **medium**. Expect a significant increase in runtime for **high** effort. This option can only be used in conjunction with the **-congestion** option.
- check_only
Checks whether the design and the libraries have all of the necessary information to run the command. This option cannot be used with other options.
- num_cpus *number_of_cpus*
Specifies the number of CPUs used in parallel during coarse placement. The *number_of_cpus* is an integer value that is less than or equal to the number of free CPUs on your machine and greater than or equal to 1. If unspecified, the default is to use one CPU. Only one Galaxy-PSYN License is checked out. Note that only the core placement algorithm is multi-threaded. Other components of the create_placement command (design loading, timing, etc.) will still use only 1 CPU.

-mpc

Runs **create_placement** with minimum physical constraints. This option controls instructs the command to check for any floorplan in the design. If there is no floorplan, it creates a floorplan based on the minimum physical constraints. If you do not specify any physical constraints, the tool assumes a set of default values and creates a default floorplan based on the number of cells and the total areas needed. The default values are 60% cell density and 1:1 aspect ratio. The cell area information is taken from the physical cell library. Refer to the **set_mpc_options** command for details of how to set up physical constraints.

-ignore_scan

Ignores scan chain connections during placement.

DESCRIPTION

The **create_placement** command performs coarse placement on the current design.

Multicorner-Multimode Support

This command uses information from active scenarios only.

EXAMPLES

The following example runs the **create_placement** command in timing-driven mode:

```
prompt> create_placement -effort high -timing_driven
```

SEE ALSO

legalize_placement(2)
psynopt(2)
read_def(2)
write_def(2)

create_placement_blockage

Creates a new placement blockage.

SYNTAX

```
status create_placement_blockage
-bbox rectangle
[-type {hard | soft | partial}]
[-blocked_percentage percentage]
[-no_register]
[-no_pin]
[-blocked_layers layers]
[-no_hard_macro]
[-name blockage_name]
```

Data Types

<i>rectangle</i>	x1 y1 x2 y2
<i>percentage</i>	integer
<i>layers</i>	string
<i>blockage_name</i>	string

ARGUMENTS

-bbox *rectangle*
Specifies the coordinates of the bounding box of the blockage.

-type {hard | soft | partial}
Specifies the type of blockage to be created.
Valid values are:

- **hard** for hard blockage

With a hard blockage, the placer does not place any standard cells or hard macros in the specified region.

- **soft** for soft blockage

With a soft blockage, the coarse placer does not place any standard cells or hard macros in the specified region. However, cells may be placed there during optimization or legalization.

- **partial** for partial blockage

For a partial blockage, the amount of area used to place cells will be limited to the specified percent of the blockage area.

Note that when you create a partial blockage, you must also specify the blocked percentage value by using the **-blocked_percentage** option.

The default value is **hard**.

This option is mutually exclusive with the **-no_hard_macro** and **-no_pin** options.

-blocked_percentage *percentage*

Specifies the percentage blockage for a partial blockage.
This option can only be used with the **partial** blockage type.

-no_register

Specifies register blockage. Restricts the coarse placer from placing any register cells within the specified blockage area.
When the **-no_register** option is specified jointly with type option, only partial blockages is affected. Hard or Soft blockages won't be affected, as the coarse placer does not place any cells within the area specified by soft and hard blockages. Only coarse placer honors the attribute set by this option.
This option is mutually exclusive with the **-no_hard_macro** and **-no_pin** options.

-no_pin

Specifies pin blockage area where the global router does not route and the pin placer does not assign pins during pin assignment.
Note that when you create a pin blockage, you can also specify the blocked layers by using the **-blocked_layers** option. If you do not specify this option, the default is to block all layers.
This option is mutually exclusive with the **-no_hard_macro**, **-no_register**, and **-type** options.

-blocked_layers *layers*

Specifies the layers for which routing to pins are blocked.
This option can only be used jointly with the **-no_pin** option.

-no_hard_macro

Specifies hard macro blockage area where the placer does not place hard macros.
This option is mutually exclusive with the **-no_pin**, **-no_register**, and **-type** options.

-name *blockage_name*

Specifies the optional name of the blockage. If you specify a name for the blockage, you can use it later in the flow to get the blockage by name.

DESCRIPTION

This command creates a new placement blockage, which is used to control the placement of cells in a specified rectangular when the placer is run.

Snapping of the bounding box is done automatically using global snap settings.

See the description of the **-type** option and the relevant placer documentation for more information.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates a soft placement blockage.

```
prompt> create_placement_blockage -bbox {0 0 100 100} -type soft
```

SEE ALSO

```
create_net_shape(2)  
create_route_guide(2)  
create_terminal(2)  
create_text(2)  
create_user_shape(2)  
create_via(2)
```

create_plan_groups

Creates instance plan groups in the design.

SYNTAX

```
int create_plan_groups
[-coordinate {coordinates_list}]
[-rectangle rectangle_area]
[-polygon {polygon_area}]
[-dimension {width height}]
[-target_aspect_ratio aspect_ratio]
[-target_utilization utilization]
[-is_fixed]
[-color range_0_to_63]
[-cycle_color]
logic_cell_list
```

Data Types

<i>coordinates_list</i>	list
<i>rectangle_area</i>	list
<i>polygon_area</i>	list
<i>aspect_ratio</i>	float
<i>utilization</i>	float
<i>range_0_to_63</i>	string
<i>logic_cell_list</i>	list

ARGUMENTS

- coordinate {*coordinates_list*}**
Specifies the coordinates of a plan group. The areas in the list are rectangles that should overlap or adjoin. Each rectangle in the plan group is identified by its lower-left and upper-right coordinates. The format for each rectangle is {*llx1 lly1 urx1 ury1*}. Substitute the defining points of the rectangles you want for *coordinates_list*. Every combination of {*llx1 lly1 urx1 ury1*} defines a target placement area for the objects. By default, this option is off. The coordinates are in microns relative to the chip origin.
- rectangle *rectangle_area***
Creates a rectangular plan group. Substitute the defining points of the rectangle you want for *rectangle_area*. A rectangle is identified by its lower-left and upper-right coordinates. The format is {*llx1 lly1*} {*urx1 ury1*}. By default, this option is off.
- polygon {*polygon_area*}**
Creates a rectilinear plan group. Substitute the defining points of the polygon you want for *polygon_area*. A polygon is identified by specifying the coordinates of all its points. The format is {{*x1 y1*} {*x2 y2*} ...}. By default, this option is off.
- dimension {*width height*}**
Specifies the dimension of plan groups. Plan groups are put outside of the

core area. The numbers are in microns. By default, this option is off.

`-target_aspect_ratio aspect_ratio`

Specifies the value the height to width aspect ratio. Together with the **-target_utilization** option, the real width and height is calculated out, and plan groups are put outside of the core area. The value of *aspect_ratio* is height divided by width . The default is **1.0**.

`-target_utilization utilization`

Specifies the value of the plan group area size divided by the total of the area size of the cells. The default value is the average design utilization.

`-is_fixed`

Specifies that the plan group is in a fixed location, and the shaping will ignore it. By default, this option is off.

`-color range_0_to_63`

Specifies the plan group color. Substitute a value for *range_0_to_63* that is in the range of **0** to **63**, inclusive. The default is **no_color** which is the special value for no user specified color.

`-cycle_color`

Allows the tool to automatically assign a color. By default, this option is off.

`logic_cell_list`

Specifies hierarchy cell names or a collection returned by the **get_cells** command. Plan groups are created on all valid hierarchy cells in the list. Only one logic cell is allowed if you use the **-rectangle**, **-polygon**, or **-coordinate** option. The tool does not support nested plan groups, meaning logical nests, not geometry nests. That is, a plan group on top/module cannot be created if plan group on top/module/third already exists.

The command returns an empty collection if no plan groups are successfully created by the given *logic_cell_list*. By default, this option is off.

DESCRIPTION

This command defines exclusive region-based placement constraints for coarse placement. Plan groups restrict the placement of cells to a specific region of the core area.

The tool guarantees that these cells are placed completely within the instance plan group. It never places cells that do not belong to the plan group inside it. Thus, the tool can be capable of converting the plan group to a soft macro later.

If you specify the **-rectangle**, **-polygon**, or **-coordinate** option, this command creates a plan group with an explicit location. If you specify the **-dimension** option it creates plan groups with the given bounding box and places them above the top of the core area.

If more than one placeable area is defined in the **-coordinate** option, the coarse placement engine can place cells in any of those areas. These placeable areas should overlap or adjoin. One application of this is to define a rectilinear plan group by breaking down the original rectilinear plan group into multiple rectangles, and

listing the defining coordinates of each rectangle in *coordinates_list*. The number of parameters in the list must be a multiple of 4.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The example creates a rectilinear plan group for a logic cell on *timer/decode0*. The rectilinear bound is created as a set of two rectangles defined by coordinates `{{10 10} {30 20}}` and `{{20 20}{30 30}}` with a common edge.

```
prompt> create_plan_groups \  
-coordinate {10 10 30 20 20 20 30 30} timer/decode0
```

SEE ALSO

```
get_cells(2)  
get_plan_groups(2)  
remove_plan_groups(2)
```

create_port

Creates ports in the current design or its subdesign.

SYNTAX

```
status create_port  
port_list  
[-direction dir]
```

Data Types

```
port_list      list  
dir            string
```

ARGUMENTS

port_list
Specifies names of ports created in the current design. Each port name must be unique within the current design.

-direction *dir*
Specifies the signal flow of the created port. The possible values are **in**, **out**, or **inout**.
The default is **in**.

DESCRIPTION

The **create_port** command creates new port objects in the current design or its subdesign. The **create_port** command creates only scalar or single bit ports.

Ports are the external connection points on a design. To connect ports to nets inside a design, use **connect_net**. To remove ports from the current design, use **remove_port**.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses **create_port** to create ports in the current design:

```
prompt> current_design  
Current design is 'TOP'.  
{ "TOP" }
```

```
prompt> create_port -direction "in" {A1 A2 A3 A4}  
Creating port 'A1' in design 'TOP'.  
Creating port 'A2' in design 'TOP'.
```

```
Creating port 'A3' in design 'TOP'.
Creating port 'A4' in design 'TOP'.
1
```

```
prompt> get_ports *
{A1 A2 A3 A4}
```

The following example uses **create_port** to create ports in the MID subdesign. U1 is an unique instance of design MID.

```
prompt> current_design
Current design is 'TOP'.
{"TOP"}
```

```
prompt> create_port -direction "in" {U1/A1 U1/A2 U1/A3 U1/A4}
Creating port 'A1' in design 'MID'.
Creating port 'A2' in design 'MID'.
Creating port 'A3' in design 'MID'.
Creating port 'A4' in design 'MID'.
1
```

```
prompt> get_pins U1/*
{U1/A1 U1/A2 U1/A3 U1/A4}
```

`create_port` command create ports in the design 'MID'. When you want to verify the ports are created, you use `get_pins` on the pins of instance of the design MID to get them.

SEE ALSO

```
all_connected(2)
connect_net(2)
current_design(2)
disconnect_net(2)
remove_port(2)
```

create_power_domain

Creates a power domain, which provides a power supply distribution network.

SYNTAX for UPF Mode

```
string create_power_domain  
domain_name  
[-elements list]  
[-include_scope]  
[-scope instance_name]
```

SYNTAX for Non-UPF Mode

```
status create_power_domain  
domain_name  
[-power_down]  
[-power_down_ctrl object_list]  
[-power_down_ack object_list]  
[-object_list object_list]
```

Data Types for UPF Mode

<i>domain_name</i>	string
<i>list</i>	list
<i>instance_name</i>	string

Data Types for Non-UPF Mode

<i>domain_name</i>	string
<i>object_list</i>	list

ARGUMENTS

domain_name

UPF Mode

Specifies the name of the power domain to be created. The name should be a simple (non-hierarchical) name.

The power domain cannot be created if there is a power domain with the same name in the specified scope or if there is a hierarchical or leaf cell instance or port with the same name in the specified scope.

Non-UPF Mode

Specifies the name of the new power domain to be created.

-elements *collection*

Specifies a collection of cells (hierarchical, blackbox/macro, or pad) that are added as an extent of the power domain. Specified cells cannot be added in other power domains of the same scope.

If you do not use either the **-elements** or the **-include_scope** option, the power domain consists of the current scope and any of its children not specified as elements in another **create_power_domain** command.

create_power_domain

-include_scope
Includes the scope of the power domain in the extent of the power domain. This means all elements within the current scope get the same supply as the power domain, but they are not explicitly added to it.

-scope *instance_name*
Specifies in which scope the power domain is to be created. The instance name is the name of a hierarchical cell. By default, the power domain is created in the current scope.

-power_down
Specifies that the new power domain is a power-down domain. You must use this option if you use the **-power_down_ctrl** option.

-power_down_ctrl *object_list*
Specifies the power-down control net for the new power domain. If you use this option you must also use the **-power_down** option.

-power_down_ack *object_list*
Specifies the power-down acknowledge net for the new power domain. If you use this option you must also use the **-power_down_ctrl** option.

-object_list *object_list*
Specifies a list of hierarchical cells that are associated with the new power domain. For each design, there can be only one top-level domain. Each hierarchical cell can be associated with only one power domain. By default, the new power domain is assumed to be the top-level domain.

DESCRIPTION

UPF Mode

This command creates a power domain in the specified scope. A power_domain is a collection of design elements that share a primary power and ground power net. The logic hierarchy level where a power domain is created is called the scope of the power domain. The set of design elements that belong to a power domain are called the "extent" of that power domain. A hierarchical cell is an example of an element. Although a design element can be in the scope of several power domains, it can be in the extent of only one power domain.

A power domain can have several supply nets, which are connected to a power domain via a supply port. A power switch of a power_domain can be used to turn on and off the power supply of part or all of the power domain. You can create a supply net, supply port, and power switch by using the **create_supply_net**, **create_supply_port** and **create_power_switch**, respectively.

When this command succeeds, it returns the full name of the power domain (from the current scope). When it fails, it returns a null string.

Non-UPF Mode

This command creates a new power domain for the current design. The power domains

for each design must be uniquely named. There can be only one top-level power domain in a design. The tool creates an always-on power domain.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

UPF Mode

The following example creates two power domains in the scope named INST1.

```
prompt> create_power_domain PD1 -elements INST1/SUB_INST -scope INST1
INST1/PD1
prompt> create_power_domain PD2 -elements INST1/SUB_INST -scope INST1
Error: Could not add cell 'INST1/SUB_INST' to power domain 'PD2'.
(MWUI-402)
prompt> create_power_domain PD2 -scope INST1 -include_scope
INST1/PD2
```

Non-UPF Mode

The following example creates a top-level power domain.

```
prompt> create_power_domain TOP_DOMAIN
```

The following example creates a lower-level power domain.

```
prompt> create_power_domain SUB_DOMAIN -power_down \
-power_down_ctrl [get_nets pd_ctrl] \
-power_down_ack [get_nets pd_ack] \
-object_list [get_cells mid1]
1
```

SEE ALSO

```
remove_power_domain(2)
report_power_domain(2)
```

create_power_net_info

Creates a power net.

SYNTAX

```
status create_power_net_info  
power_net_name  
-power | -gnd  
[-switchable]  
[-nominal_voltages nominal_voltage_list]  
[-voltage_ranges voltage_range_list]
```

Data Types

<i>power_net_name</i>	string
<i>nominal_voltage_list</i>	list
<i>voltage_range_list</i>	list

ARGUMENTS

power_net_name
Specifies the name of the new power net information to be created.

-power
Specifies that the type of the new power net is "power". This option is mutually exclusive with the **-gnd** option.

-gnd
Specifies that the type of the new power net is "gnd". This option is mutually exclusive with the **-power** option.

-switchable
Specifies that the power net can be cut off externally, or, if driven by an internal switch, cut off internally. This option can only be used with **-power**.

-nominal_voltages *nominal_voltage_list*
Specifies the list of nominal voltages that the power net has been designed to operate. The actual operating voltage of the power net can deviate from the nominal within a certain tolerance as specified by *voltage_range_list*. This option must be specified when using the **-voltage_ranges** option. This option can only be used with **-power**.

-voltage_ranges *voltage_range_list*
Specifies the list of allowed voltage ranges around the nominal voltages that this power net has been designed to operate. This option must be specified when using the **-nominal voltages** option. This option can only be used with **-power**.

DESCRIPTION

The **create_power_net_info** command creates new power net information for the current

design. The power net is either of type **power** or of type **gnd**.

EXAMPLES

The following examples use the command to create power nets:

```
prompt> create_power_net_info VSS_net -gnd
1

prompt> create_power_net_info VDD1_net -power
1

prompt> create_power_net_info VDD2_net -power \
        -switchable -nominal_voltages {0.9 1.08} \
        -voltage_ranges {0.88 0.92 1.05 1.10}
1
```

SEE ALSO

```
remove_power_net_info(2)
report_power_net_info(2)
set_voltage(2)
```


create_power_straps

Creates power straps in a design.

SYNTAX

```
status create_power_straps
-nets collection_of_nets | -undo
[-direction horizontal | vertical]
[-start_at distance]
[-layer number_or_name]
[-width distance]
[-
configure groups_and_step | groups_and_stop | step_and_stop | rows | macros | group
s_and_stop | step_and_stop | rows | macros]
[-num_groups int]
[-step distance]
[-stop distance]
[-pitch_within_group distance]
[-
start_low_ends boundary | first_targets | coordinate | last_targets | first_targets
| coordinate | last_targets]
[-start_low_ends_coordinate distance]
[-
start_high_ends boundary | first_targets | coordinate | last_targets | first_target
s | coordinate | last_targets]
[-start_high_ends_coordinate distance]
[-
extend_low_ends to_first_target | to_boundary_and_generate_pins | force_to_boundary
_and_generate_pins | off | to_boundary_and_generate_pins | force_to_boundary_and_ge
nerate_pins | off]
[-
extend_high_ends to_first_target | to_boundary_and_generate_pins | force_to_boundar
y_and_generate_pins | off | to_boundary_and_generate_pins | force_to_boundary_and_g
enerate_pins | off]
[-num_placement_strap int]
[-increment_x_or_y distance]
[-special_via_rule
[-special_via_x_offset distance]
[-special_via_y_offset distance]
[-offset_both_sides_for_special_via]
[-special_via_x_size distance]
[-special_via_y_size distance]
[-special_via_x_step distance]
[-special_via_y_step distance]]
[-advanced_via_rules]
[-special_rules string]
[-look_inside_std_cells
[-std_cells collection_of_cells]]
[-keep_floating_wire_pieces]
[-ignore_cell_boundary]
[-clip_at_top_cell_boundaries]
[-do_not_merge_targets]
[-optimize_wire_locations]
```

```
[-ignore_parallel_targets
[-define_parallel_targets_by_wire_directions]]
[-do_not_route_over_macros]
[-extend_for_multiple_connections
[-extension_gap distance]]
[-mark_as_std_cell_pin_connections | -mark_as_ring]
```

Data Types

<i>collection_of_nets</i>	collection
<i>distance</i>	distance
<i>number_or_name</i>	layer
<i>int</i>	integer
<i>collection_of_cells</i>	collection

ARGUMENTS

-nets *collection_of_nets*
 Specifies nets on which to create straps. You must specify at least one power or ground net. If you specify more than one value, separate the values with a space. This is a required option, unless you specify the **?undo** option.

-undo
 Removes straps created by the last **create_power_straps** run. If you specify this option, it supersedes all other options.

-direction horizontal | vertical
 Specifies the strap direction. Valid values are horizontal and vertical. The default is horizontal.

-start_at *distance*
 Specifies the starting strap coordinate. The unit for the distance value is the unit specified in the technology file.
 For a horizontal strap, this is the y-coordinate. For a vertical strap, this is the x-coordinate. The default is 0.0.

-layer *number_or_name*
 Specifies the metal layer on which to create straps. The value can be specified either as the layerNumber or maskName from the technology file. The default is to select the first layer for the specified direction.

-width *distance*
 Specifies the strap width. The unit for the distance value is the unit specified in the technology file. The default is the width of the layer.

-configure groups_and_step | groups_and_stop | step_and_stop | rows | macros | groups_and_stop | step_and_stop | rows | macros
 Specifies the strap configuration. Valid values are the following:

- **groups_and_step** (default) - Places straps based on the number of groups specified by the **-num_groups** option and the spacing specified by the **-step** option.
- **groups_and_stop** - Places straps based on the number of groups specified by

the **-num_groups** option and the stop point specified by the **-stop** option. The tool uses this information to determine the strap spacing.

step_and_stop - Places straps based on the spacing specified by the **-step** option and the stop point specified by the **-stop** option. The tool uses this information to determine the number of groups.

- **rows** - Creates straps over the standard cell areas. The tool automatically determines the strap locations.

- **macros** - Creates straps over the macro cell pins. The tool automatically determines the strap locations.

You can specify only one of these configurations.

-num_groups *int*

Specifies the number of strap groups to create. Do not use this option if you are using the **step_and_stop** configuration. The default is 1.

-step *distance*

Specifies the distance from the reference strap of one group to the reference strap of an adjacent group. The unit for the distance value is the unit specified in the technology file.

For vertical straps, specifying a positive value places the next strap in the group to the right of the current strap and specifying a negative value places the next strap to the left of the current strap. For horizontal straps, specifying a positive value places the next strap above the current strap and specifying a negative value places the next strap below the current strap. The default is 0.0. Do not use this option if you are using the **groups_and_stop** configuration.

-stop *distance*

Specifies the location where strap placement is to stop. The unit for the distance value is the unit specified in the technology file. The default is 0.0. Do not use this option if you are using the **groups_and_step** configuration.

-pitch_within_group *distance*

Specifies the distance to place between the center lines of two adjacent straps in the same group. The unit for the distance value is the unit specified in the technology file.

If the straps run vertically, specifying a positive value places the next strap to the right of the first and specifying a negative value places the next strap to the left of the first. If the straps run horizontally, specifying a positive value places the next strap above the first and a negative value places the next strap below the first. The default is the pitch of the layer defined in the technology file.

-start_low_ends *boundary | first_targets | coordinate | last_targets | first_targets | coordinate | last_targets*

Specifies the location for the low strap ends. Valid arguments are

- **boundary** (default) - Places the low end of the strap at the core boundary.

- **first_targets** - Places the low end of the strap at the first target on the same net. For horizontal straps, this is the first target found by moving left from the initial strap. For vertical straps, this is the first target

found by moving down from the initial strap.

- **coordinate** - Places the low end of the strap at the coordinate specified by the **-start_low_ends_coordinate** option.

- **last_targets** - Places the low end of the strap at the last target on the same net. For horizontal straps, this is the leftmost target. For vertical straps, this is the lowest target.

You can specify only one of these values.

-start_low_ends_coordinate *distance*

Specifies the coordinate for the low end of the strap when using **-start_low_ends_coordinate**. The unit for the distance value is the unit specified in the technology file. For horizontal straps, this is the x-coordinate. For vertical straps, this is the y-coordinate. The default is 0.0.

-start_high_ends *boundary | first_targets | coordinate | last_targets | first_targets | coordinate | last_targets*

Specifies the location for the high strap ends. Valid arguments are

- **boundary** (default) - Places the high end of the strap at the core boundary.

- **first_targets** - Places the high end of the strap at the first target on the same net. For horizontal straps, this is the first target found by moving right from the initial strap. For vertical straps, this is the first target found by moving up from the initial strap.

- **coordinate** - Places the high end of the strap at the coordinate specified by the **-start_high_ends_coordinate** option.

- **last_targets** - Places the high end of the strap at the last target on the same net. For horizontal straps, this is the rightmost target. For vertical straps, this is the highest target.

You can specify only one of these values.

-start_high_ends_coordinate *distance*

Specifies the coordinate for the high end of the strap when using **-start_high_ends_coordinate**. The unit for the distance value is the unit specified in the technology file. For horizontal straps, this is the x-coordinate. For vertical straps, this is the y-coordinate. The default is 0.0.

-extend_low_ends *to_first_target | to_boundary_and_generate_pins | force_to_boundary_and_generate_pins | off | to_boundary_and_generate_pins | force_to_boundary_and_generate_pins | off*

Specifies how to extend straps from the low end. Valid values are

- **to_first_target** (default) - Extends the straps from the low end until the strap reaches the first target on the same net.

- **to_boundary_and_generate_pins** - Extends the straps from the low end until the strap reaches the core boundary and then generates a pin. The strap stops when it reaches the first target on the same net.

- **force_to_boundary_and_generate_pins** - Extends the straps from the low end until the strap reaches the core boundary and then generates a pin. The strap does not stop when it reaches the first target on the same net.

- **off** - Does not extend the straps from the low end.
You can specify only one of these values.

```
-extend_high_ends to_first_target | to_boundary_and_generate_pins |  
force_to_boundary_and_generate_pins | off | to_boundary_and_generate_pins |  
force_to_boundary_and_generate_pins | off  
Specifies how to extend straps from the high end. Valid values are
```

- **to_first_target** (default) Extends the straps from the high end until the strap reaches the first target on the same net.

- **to_boundary_and_generate_pins** - Extends the straps from the high end until the strap reaches the core boundary and then generates a pin. The strap stops when it reaches the first target on the same net.

- **force_to_boundary_and_generate_pins** - Extends the straps from the high end until the strap reaches the core boundary and then generates a pin. The strap does not stop when it reaches the first target on the same net.

- **off** - Does not extend the straps from the high end.
You can specify only one of these values.

```
-num_placement_strap int  
Specifies the number of times strap groups are to be placed. Valid values are  
greater than or equal to 1. The default is 1.
```

```
-increment_x_or_y distance  
Specifies the distance between the first strap and subsequent straps when the  
-num_placement_strap option is greater than 1. The unit for the distance  
value is the unit specified in the technology file. For horizontal straps,  
this is the y-distance. For vertical straps, this is the x-distance. The  
default is 0.0.
```

```
-special_via_rule  
Creates a matrix of small via arrays instead of one large via array at wire  
intersections. Using smaller via arrays could free some routing space. You  
specify the x- and y-offsets, x- and y-size, and x- and y- step values by  
using the special via options described below. If the specified values  
violate design rules, vias are not created.
```

```
-special_via_x_offset distance  
Specifies the x-offset from the lower-left corner of the wire intersection  
area where you want to create the first special via. The unit for the distance  
value is the unit specified in the technology file. The default is 0.0.
```

```
-special_via_y_offset distance  
Specifies the y-offset from the lower-left corner of the wire intersection  
area where you want to create the first special via. The unit for the distance  
value is the unit specified in the technology file. The default is 0.0.
```

-offset_both_sides_for_special_via
 Applies the x- and y-offsets to both lower-left and upper-right corner of the wire intersection area. By default, the offsets are applied only to the lower-left corner of the wire intersection area.

-special_via_x_size *distance*
 Specifies the horizontal size of the special via arrays. The unit for the distance value is the unit specified in the technology file. The default is 0.0.

-special_via_y_size *distance*
 Specifies the vertical size of the special via arrays. The unit for the distance value is the unit specified in the technology file. The default is 0.0.

-special_via_x_step *distance*
 Specifies the horizontal distance between the left sides of adjacent via arrays. The unit for the distance value is the unit specified in the technology file. The default is 0.0.
 A via is not created if it would violate DRC violations.

-special_via_y_step *distance*
 Specifies the vertical distance between the bottom sides of adjacent via arrays. The unit for the distance value is the unit specified in the technology file. The default is 0.0. A via is not created if it would violate DRC violations.

-advanced_via_rules
 Uses advanced via rules that have been previously set by the **set_preroute_advanced_via_rule** command.

-special_rules *string*
 Uses special rules specified by the **set_preroute_special_rules** command. You must enter the string exactly as it was specified in the **?name** option of **set_preroute_special_rules**.

-look_inside_std_cells
 Creates vias to connect straps to pins inside standard cells. By default, straps are not connected to pins inside standard cells.

-std_cells *collection_of_cells*
 Specifies the standard cells to consider when connecting the straps. If *collection_of_cells* is not empty, the **-look_inside_std_cells** option is turned on automatically, and only pins of the specified cells are considered as targets for the straps.
 If this option is omitted or the collection is empty, either all standard cells or none of them are considered, depending on whether the **-look_inside_std_cells** option is used.

-keep_floating_wire_pieces
 Keeps extra wire pieces that are not connected to other wires.

-ignore_cell_boundary
 Keeps out-of-bound straps.

-clip_at_top_cell_boundaries
Cuts straps at the top cell boundaries. This is useful to create straps within the boundaries of a rectilinear design.

-do_not_merge_targets
Disables merging of target objects before creating connections.

-optimize_wire_locations
Saves more tracks for detail routing by adjusting the created straps by small distances horizontally or vertically.

-ignore_parallel_targets
Does not connect parallel targets. By default, parallel targets are defined by their x- and y-dimensions. A target is parallel to a horizontal strap if its x-dimension is greater than or equal to its y-dimension. A target is parallel to a vertical strap if its y-dimension is greater than or equal to its x-dimension.

-define_parallel_targets_by_wire_directions
Defines parallel targets by their wire directions. A target is parallel to a horizontal strap if its wires are horizontal. A target is parallel to a vertical strap if its wires are vertical. By default, parallel targets are defined by their x- and y-dimensions.

-do_not_route_over_macros
Prevents the prerouter from generating wires over macro cells. If you use this option, the tool removes the parts of a strap that go over a macro.

-extend_for_multiple_connections
Extends the low and high ends of a strap to connect more targets on the same net. The extension continues until either there are no more targets in the respective direction or the next target exceeds the spacing threshold specified by the **-extension_gap** option.

-extension_gap *distance*
Connects adjacent targets that are closer than the *distance* value. The unit for the distance value is the unit specified in the technology file. This value is the space threshold for multiple connections.

-mark_as_std_cell_pin_connections
Marks the created power or ground straps as the "pg_std_cell_pin_conn" type. By default, they created are marked as the "pg_strap" type. This option is mutually exclusive with the **-mark_as_ring** option.

-mark_as_ring
Marks the created power or ground straps as the "pg_ring" type. By default, they created are marked as the "pg_strap" type. This option is mutually exclusive with the **-mark_as_std_cell_pin_connections** option.

DESCRIPTION

This command creates power straps in a design. Use a few wide straps rather than many thin straps to improve the placement quality and decrease the placement runtime. This command can automatically connect the straps to the closest power or

ground ring at, or beyond, both ends of the straps. All pins generated by prerouting commands are marked as fixed to prevent the **create_power_straps** command from moving them during other operations. You can use the **set_preroute_drc_strategy** command to change the internal application of DRC rules.

Note: You must use the **derive_pg_connection** command to create the port-to-net connections for power and ground nets that are not in the netlist.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows how to create 7 vertical power straps for the VDDC and VSSC nets on the M3 layer with a width of 5 uu. The first strap starts at x-coordinate 530. There is a spacing of 100 uu between straps. The low ends start at the y-coordinate 280. The high ends end at y-coordinate 980. The straps extend for multiple connections in the radius of 1000 uu.

```
prompt> create_power_straps\  
-nets {VDDC VSSC}\  
-direction vertical\  
-layer M3\  
-width 5\  
-start_at 530\  
-num_placement_strap 7\  
-increment_x_or_y 100\  
-start_low_ends coordinate\  
-start_low_ends_coordinate 280.0\  
-start_high_ends coordinate\  
-start_high_ends_coordinate 980.0\  
-extend_for_multiple_connections\  
-extension_gap 1000
```

The following example shows how to create power straps configured by placement rows following the set of special rules named "specGS".

```
prompt> create_power_straps\  
-nets {VSS VDD}\  
-configure rows\  
-direction horizontal\  
-layer PM\  
-width 8.0\  
-pitch_within_group 10.08\  
-step 20.16\  
-extend_low_ends to_first_target\  
-extend_high_ends to_first_target\  
-keep_floating_wire_pieces\  
-ignore_parallel_targets\  
-special_rules specGS
```


For more information about special rules, see the **set_preroute_special_rules** man page.

The following example shows how to create power straps configured by macro cell pins following the set of special rules named "specLMVmem".

```
prompt> create_power_straps\  
-nets {VDD VSS}\  
-configure macros\  
-direction vertical\  
-layer M4\  
-advanced_via_rules\  
-extend_low_ends off\  
-extend_high_ends off\  
-special_rules specLMVmem
```

For more information about special rules, see the **set_preroute_special_rules** man page.

SEE ALSO

```
create_pad_rings(2)  
create_preroute_vias(2)  
create_rectangular_rings(2)  
preroute_instances(2)  
preroute_standard_cells(2)  
report_preroute_drc_strategy(2)  
set_distributed_route(2)  
set_preroute_advanced_via_rule(2)  
set_preroute_drc_strategy(2)  
set_preroute_special_rules(2)  
verify_pg_nets(2)
```

create_power_switch

Creates a power switch at the specified power domain. This command is supported only in UPF mode.

SYNTAX

```
string create_power_switch  
switch_name  
-domain domain_name  
-output_supply_port {port_name supply_net_name}  
-input_supply_port {port_name supply_net_name}  
-control_port {port_name net_name}  
[-ack_port {port_name net_name [{boolean_function}]}]  
[-ack_delay {port_name delay}]  
-on_state {state_name input_supply_port {boolean_function}}  
[-off_state {state_name {boolean_function}}]
```

Data Types

<i>switch_name</i>	string
<i>domain_name</i>	string
<i>port_name</i>	string
<i>supply_net_name</i>	string
<i>net_name</i>	string
<i>boolean_function</i>	list
<i>delay</i>	string
<i>state_name</i>	string
<i>input_supply_port</i>	string

ARGUMENTS

switch_name
Specifies the name of the power switch to be created. The name should be a simple (non-hierarchical) name. If a power switch already exists with the same name in the specified power domain, the power switch cannot be created. This option is required.

-domain *domain_name*
Specifies which power domain contains the power switch. If the power domain with the specified name does not exist in the current scope, the command fails. This option is required.

-output_supply_port {*port_name* *supply_net_name*}
Specifies the name of the output port of the power switch and the supply net where the port connects. If a port exists with the same name on the power switch, the command fails. If there is no supply net with the same name in the current scope, the command fails. This option is required.

-input_supply_port {*port_name* *supply_net_name*}
Specifies the name of the input port of the power switch and the supply net

where the port connects. If a port exists with the same name on the power switch, the command fails. If there is no supply net with the same name in the current scope, the command fails. This option is required and can be specified more than once. One power switch can have multiple input supply ports.

`-control_port {port_name net_name}`

Specifies the name of the control port of the power switch and the logical net where this port connects. If a port with the specified name already exists on the power switch, the command fails. If a net with the specified name does not exist, the command fails.

This option is required and can be specified more than once. One power switch can have multiple control ports.

`-ack_port {port_name net_name [{boolean_function}]}`

Specifies the name of the acknowledge port of the power switch and the logical net where this port connects. If a port with the specified name already exists on the power switch, the command fails. If a net with the specified name does not exist, the command fails.

If this option is not specified, the power switch will not have an acknowledge port. Optionally, a Boolean function can also be specified.

This option can be specified multiple times.

`-ack_delay {port_name delay}`

Specifies the acknowledge port on the switch and the corresponding acknowledge delay.

This option can be specified multiple times.

`-on_state {state_name input_supply_port {boolean_function}}`

Specifies a named on state, the relevant input supply port, and its Boolean function.

This option can be specified multiple times.

`-off_state {state_name {boolean_function}}`

Specifies a named off state and its relevant Boolean function.

This option can be specified multiple times.

DESCRIPTION

The `create_power_switch` command enables you to create a power switch at the specified power domain. The switch is created within the scope of the power domain. Each power switch must be connected with an input supply net and an output supply net. The power switch can be connected with an acknowledge logical net and several control logical nets. Each net is connected with a power switch via a switch port. The switch ports are automatically created if the power switch is created successfully.

This command returns the full name of the power switch (from the current scope) upon success and the command returns a null string upon failure.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates power switch SW1 within power domain PD1:

```
prompt> create_power_switch SW1 -domain PD1 \  
        -output_supply_port {vout VNO2} \  
        -input_supply_port {vin1 VNI1} \  
        -input_supply_port {vin2 VNI2} \  
        -control_port {ctrl_small ON1} \  
        -control_port {ctrl_large ON2} \  
        -ack_port {ack_p ACKN {ctrl_small & !ctrl_large}} \  
        -ack_delay {ack_p 1} \  
        -on_state {full_s vin1 {ctr_small}} \  
        -off_state {off_s {!ctr_small}}
```

SEE ALSO

```
remove_power_switch(2)  
report_power_switch(2)
```

create_power_switch_array

Adds power switch cells into the design in an array pattern.

SYNTAX

```
status create_power_switch_array
-lib_cell lib_cell_or_power_switch
[-voltage_area voltage_area]
[-bounding_box rectangle_{left_bottom_right_top}]
[-relative_to_voltage_area]
[-offset_to_va list]
[-design hierarchy_name]
-x_increment dx
-y_increment dy
[-start_row index]
[-start_column index]
[-snap_to_row_and_tile]
[-orientation {N | W | S | E | FN | FE | FS | FW}]
[-respect list]
[-pattern normal | staggered]
[-prefix prefix_name]
```

Data Types

<i>lib_cell_or_power_switch</i>	collection or list of one item
<i>voltage_area</i>	collection or list of one item
<i>hierarchy_name</i>	string
<i>dx</i>	float
<i>dy</i>	float
<i>index</i>	integer
<i>prefix_name</i>	string

ARGUMENTS

- lib_cell** *lib_cell_or_power_switch*
Specifies the library cell to be used as the power switch cell. For UPF mode, specify a power switch. If you specify a library cell in UPF mode, the switch cells will be placed correctly, but the power and ground pin connections might not be automatically derived by the **derive_pg_connection** command. For non-UPF mode, you must specify a library cell. This is a required option.
- voltage_area** *voltage_area*
Specifies the voltage area in which to place the switch cells. For UPF mode, use a power domain name. For non-UPF mode, use a voltage area name. The tool uses this option and the **-design** option to determine where to insert the switch cells.
- If you specify the **-voltage_area** option, but not the **-design** option, the tool finds a logic hierarchy that matches the voltage area and inserts the switch cells there.
 - If you specify both the **-voltage_area** and **-design** options, the tool verifies that these specifications are consistent.

- If you specify the **-design** option, but not the **-voltage_area** option, the tool inserts the switch cells in the specified logic hierarchy.
- If you do not specify either the **-voltage_area** or **-design** option, the tool inserts the switch cells in the top-level design.

-bounding_box *rectangle_{left_bottom_right_top}*

Specifies the bounding box in microns of the grid to be considered.

If you specify both the **-voltage_area** and **-bounding_box** options, switch cells are added to the intersecting region of the specified bounding box and voltage area.

-relative_to_voltage_area

Indicates that the specified bounding box coordinates are relative to the lower-left corner of the voltage area's bounding box. This option works only with the **-voltage_area** option.

If you do not specify this option, the specified bounding box coordinates are relative to the design's origin. If you specify this option without the **-voltage_area** option, it is ignored.

-offset_to_va *list*

Specifies the offset list relative to the voltage area. By default, the switch cells are to be placed all over the given voltage area. If this option is specified, the insertion area shrinks from the original voltage area by the given offset. A negative value is not allowed. The list can have 1, 2 or 4 values. The unit is in microns. This option is ignored if the **-bounding_box** option is specified.

- If you specify only 1 value, this value is applied to the all edges of voltage area.
- if you specify 2 values, the first value is horizontal value and the second is vertical value.
- If four values are given, these mean {left_offset bottom_offset right_offset top_offset}.

-design *hierarchy_name*

Specifies where to insert the switch cells in the logical hierarchy.

The tool uses this option and the **-voltage_area** option to determine where to insert the switch cells. For details, see the description for the **-voltage_area** option.

-x_increment *dx*

Specifies the step in the x-direction in microns. Note that when the specified value is less than or equal to 0, it is adjusted to the width of bounding box or voltage area. For disjoint voltage area, this might cause problems. This is a required option.

-y_increment *dy*

Specifies the step in the y-direction in microns. Note that when the specified value is less than or equal to 0, it is adjusted to the height of the cell. This is a required option.

-start_row *index*

When a switch cell is created and placed on a certain grid point, the index of the row this grid point is located in is added to the cell instance name. You can use this option to control the index of the first row. The default value is 0.

`-start_column index`

When a switch cell is created and placed on a certain grid point, the index of the column this grid point is located in is added to the cell instance name. You can use this option to control the index of the first column. The default value is 0.

`-snap_to_row_and_tile`

Causes the grid to snap to row and tile. This includes the bounding box as well as the x- and y-increments. This option applies only to standard cells, and it causes cells to be placed in legal orientations allowed by the rows. Note that this means the **-orientation** option is not strictly followed.

`-orientation {N | W | S | E | FN | FE | FS | FW}`

Specifies the orientation of the placement of the switch cells. The valid values are **N**, **W**, **S**, **E**, **FN**, **FE**, **FS**, and **FW**.

The default orientation is **N**. Note that this orientation might not be respected if you use the **-snap_to_row_and_tile** option. In this case, the orientation applies to the cell only when it is legal for the row that a cell is in.

`-respect list`

Specifies what type of objects are avoided by switch cell placement. In other words, these objects cut out nodes in the insertion grid. The argument is a list of one or more of the following strings: **hard_blockage**, **soft_blockage**, **blockage**, **macro**, **fixed_cells**, and **all**.

If you do not specify this option, the switch cell placement does not honor any placement blockages or macros.

`-pattern normal | staggered`

Specifies the insertion pattern for the switch cells. If you specify **normal**, the tool adds a switch cell on every point of the insertion grid. If you specify **staggered**, the tool inserts switch cells on alternating grid points. The default is **normal**.

`-prefix prefix_name`

Specifies a name prefix for all the inserted switch cells.

If you do not specify this option, the default prefix is headerfooter<num>_ where <num> is a unique number. Thus the default names of the inserted cells are headerfooter<num>_<ref_lib_cell_name>R<row_num>C<column_num>. This allows you to do pattern matching to select cells on a row or column basis. Note that the row or column numbers might not be the same for all cells with same y-location or x-location, respectively. For example, if you use the **-respect** option, the grid is cut out at blockages and macros, which causes a new row or column number to be used.

DESCRIPTION

This command inserts the specified reference library cells onto the locations specified by the **-bounding_box**, **-x_increment**, and **-y_increment** options. The insertion is such that the lower-left corner of the cells is on the grid points. If the grid does not align to the minimum grid, the grid is automatically minimally adjusted to be on the minimum grid. The minimum grid is the lithography grid. If you use the **-snap_to_row_and_tile** option and the specified reference library cell is a standard cell, the grid is automatically adjusted to be on appropriate row sites. A

message is printed notifying you of this change. However, this command does not guarantee that placed cells will be legal. If legalization is desired, you can specify the grid such that cells are placed in legal locations or you can run the legalizer after this command. The inserted cells are marked as fixed placed.

This command respects both the defined grid and the voltage area specified by the **-voltage_area** option. In other words, if a grid is partially in and out of the specified voltage area, the cells are only placed inside the voltage area. If you do not specify the **-voltage_area** option, the cells are placed only at the top level.

After insertion, you can use the **remove_cell** command to remove the switch cells.

EXAMPLES

The following example creates and places switch cells in the INST voltage area using the GSWITCH library cell. The cells are placed on a grid whose y-step is 8 microns and x-step is 40 microns. The new cell instances created by this command will have prefix "INSTHF" in their names.

```
prompt> create_power_switch_array \  
-lib_cell "GSWITCH" \  
-y_increment 8 -x_increment 40 \  
-voltage_area "INST" \  
-prefix "INSTHF"
```

The following example is similar to the previous example, but it adds cells only to the intersection of the INST voltage area and the specified bounding box area.

```
prompt> create_power_switch_array \  
-lib_cell "GSWITCH" \  
-bounding_box {110 200 250 400} \  
-y_increment 8 -x_increment 40 \  
-voltage_area "INST" -prefix "INSTHF"
```

The following example adds switch cells within the specified bounding box at the top level, outside of any voltage area, using the GSWITCH library cell. The row and column index start from 10 and 20, respectively.

```
prompt> create_power_switch_array \  
-lib_cell "GSWITCH" \  
-bounding_box {110 200 250 400} \  
-y_increment 8 -x_increment 40 \  
-start_row 10 -start_column 20 \  
-prefix "INSTHF"
```

For UPF mode, the name of the power switch in the logic design is relative to the current scope. If the current scope is TOP, there is a power switch named PS_A under scope SC_A and a power switch named SC_A under scope TOP.

```
prompt> create_power_switch_array \  

```



```
-lib_cell "SC_A/PS_A" \  
-bounding_box {110 200 250 400} \  
-y_increment 8 -x_increment 40 \  
-start_row 10 -start_column 20 -prefix "INSTHF"
```

SEE ALSO

`remove_cell(2)`

create_power_switch_ring

Creates multithreshold-CMOS cells around a voltage area or a macro in a ring pattern, either a full ring or part of a ring. If the pattern is part of ring, specify the startpoint and endpoint on the boundary. The switch cells are placed counterclockwise.

SYNTAX

```
status create_power_switch_ring
-switch_lib_cell lib_cells_or_power_switches
[-outer_corner_lib_cell outer_corner_lib_cell_name]
[-inner_corner_lib_cell inner_corner_lib_cell_name]
[-area_obj voltage_area_or_macro
  | -polygon {{llx_lly_urx_ury}_or_{{x1_y1}_{{x2_y2}_{{x3_y3}}...}}]
[-design hierarchy_name]
[-prefix string]
[-switch_orientation N | W | S | E | FN | FE | FS | FW]
[-outer_corner_orientation N | W | S | E | FN | FE | FS | FW]
[-inner_corner_orientation N | W | S | E | FN | FE | FS | FW]
[-density float]
[-offset {{f1 f2...}}]
[-start_point {{x_y}}]
[-end_point {{x_y}}]
[-check_overlap]
[-filler_lib_cell {{pattern}}]
```

ARGUMENTS

-switch_lib_cell *lib_cells_or_power_switches*
This is a required argument. Specify the library cell to be used as the switch cell. In the UPF mode specify the power switch or the library cell. If you specify the library cell in the UPF mode, though the switch cells are placed correctly, the power and ground connections might not be derived automatically by the **derive_pg_connection** command. In the non-UPF mode, specify the library cell.

-outer_corner_lib_cell *outer_corner_lib_cell_name*
Specify a single collection that contains a corner cell to be placed at an outer (convex) corner of the switch cell ring. Outer corner cell insertion is skipped if you do not specify this option. A rectangle has an outer corner only.

-inner_corner_lib_cell *inner_corner_lib_cell_name*
Specify a singleton collection that contains a corner cell to be placed at an inner (concave) corner of the switch cell ring. Inner corner cell insertion is skipped if you do not specify this option.

-area_obj *voltage_area_or_macro*
Specify the voltage area or macro around which the ring of switch cells is to be created. If you specify a voltage area with this option, use the power domain name in the UPF mode. If the target area object is a disjoint voltage area, power switches can be inserted around all these disjoint polygons. If

this option is not specified, the switch cells are created around the default voltage area.

-polygon {llx_lly_urx_ury}_or_{{x1_y1}_{x2_y2}_{x3_y3}...}

Specify a user-defined rectilinear polygon area, and switch cells are inserted around that polygon. The format of a rectangle specification is {llx_lly_urx_ury}. Or the coordinates of all its corners should be specified in the format of {{x1_y1}_{x2_y2}_{x3_y3}...}. The units of the polygon are in microns. This option and the **-area_obj** option are mutually exclusive.

-design *hierarchy_name*

Specifies the hierarchy name under which the header or footer is inserted. If this option is not specified, a) For a voltage area, the command finds a logic hierarchy that matches the voltage area and inserts the cells. b) For a macro, the command inserts the cells under the top level.

Note: If power switches are inserted around the voltage area and under the top level, use the **-design** option.

-prefix *string*

Specify a name prefix for all cells inserted by this command. Without this option, the default prefix is RINGMTCMOS. The name of the inserted switch cell is *prefix_SWITCH_<num>_refLibCellName* where *num* is an incremental unique integer. Filler cells and corner cells have a similar naming convention, *prefix_FILLER_<num>_refLibCellName* and *prefix_CORNER_<num>_refLibCellName*.

-switch_orientation N | W | S | E | FN | FE | FS | FW

Specify the orientation of the switch cells placed on the bottom edge of the ring. The default orientation is N.

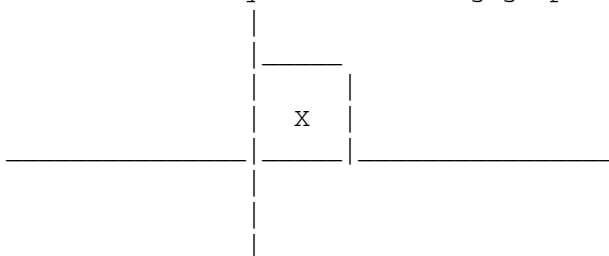
N	rotate 0
W	rotate 90
S	rotate 180
E	rotate 270
FN	rotate 0 and mirror X
FS	rotate 0 and mirror Y
FW	rotate 90 and mirror X
FE	rotate 90 and mirror Y

-outer_corner_orientation N | W | S | E | FN | FE | FS | FW

Specify the orientation of the outer corner cells placed on the bottom-left corner (bottom-left corner should always be an outer corner). The default orientation is N.

-inner_corner_orientation N | W | S | E | FN | FE | FS | FW

Specify the orientation of inner-corner cells placed on the corner which is very similar to the outer corner cells, except it is a concave corner. This is illustrated by the following graph.



-density *float*

Specify a floating point number between 0.0 and 1.0 to indicate the multithreshold-CMOS power switch cell density of the ring. The default is 1.0. Higher density implies more switch cells to be inserted.

-offset {*f1 f2...*} **-offset** {*f1 f2...*}

Define an additional distance to shrink or enlarge the boundary of the ring with respect to the voltage area or macro boundary. The unit is micron and the default offset is 0. A positive or negative value can be specified. Offset adjustment must maintain the relative shape of the boundary, otherwise an error is issued. Positive offset value implies enlarging the boundary. If only one value is specified, the value is applied to all edges. If only two values are specified, the first value is considered the horizontal value and the second, the vertical value. To specify a different offset value for a different edge, a set of values can be specified. If more than two values are specified, they must match the number of edges of the ring. The first edge is the horizontal edge with the smallest y-coordinate followed by the smallest x-coordinate followed by the second edge, in a counter clockwise traversal.

-start_point {*x_y*}

Specify the startpoint of the ring. The startpoint must be on the boundary edges of the given polygon. If the **-offset** option is specified, the polygon boundary should be reduced or enlarged. By default, the startpoint of the ring is the left-most and bottom-most point of the given polygon.

-end_point {*x_y*}

Specify the endpoint of the ring. If the **-offset** option is specified, the polygon boundary should be reduced or enlarged. The endpoint must be on the boundary edge of the given polygon. By specifying this option, you can define a segment ring. By default, the ring is the entire boundary of the given polygon. Note that if the target area object is a disjoint voltage area, the startpoint and endpoint should be on the same polygon. Power switches will only be inserted around that polygon, leaving all other polygons untouched.

-check_overlap

Specify to check the overlap of the ring. When you specify this option the tool checks if any inserted power switch cells, filler cells or corner cells in the ring, overlap with other objects. If an overlap is found, the tool issues warning messages. By default, the command does not check overlapping of the ring.

DESCRIPTION

This command performs ring-based multithreshold-CMOS cell insertion using the specified options. The command creates is a ring consisting of switch cells with optional filler cells and corner cells. The ring is constructed around the given voltage area or macro. The number of switch cells inserted is controlled by the density option. The switch ring, filler and corner cells are marked FIXED by this command.

The placement of power switch cells are rotated by 90/-90 degrees when changing from a vertical edge to a horizontal edge. The height of multithreshold-CMOS switch cells, filler cells and corner cells are identical. The width of the switch cells

and filler cells are a multiple of the placement site (unit tile). Corner cells have a square boundary.

You should specify the **-density**, **-offset**, and **-list_of_filler_cells** options for a voltage area and the orientation of the cells to form a ring.

EXAMPLES

The following example creates a multithreshold-CMOS ring using switch library cell, GSWITCH, around an object INST, which can be a voltage area or a macro. The space between the ring and the area boundary is 8 microns. All the newly created cell instances, switch cells, corner cells and filler cells, have INST_MTCMOS_ as the prefix.

```
prompt> create_power_switch_ring -area_obj INST \  
      -switch_lib_cell GSWITCH -offset 8 -prefix INST_MTCMOS_
```

The following example creates the multithreshold-CMOS ring with density 0.5 from the startpoint {100 200} to {500 600} counterclockwise. The space between two switch cells are filled using the library cell FILLER. Each convex corner on the ring is placed using library cell CORNER.

```
prompt> create_power_switch_ring -area_obj INST \  
      -switch_lib_cell GSWITCH -density 0.5 \  
      -start_point {100 200} -end_point {500 600} \  
      -corner_lib_cell CORNER -filler_lib_cells FILLER
```

SEE ALSO

```
create_power_switch_array(2)  
explore_power_switch(2)  
replace_power_switch(2)  
optimize_power_switch(2)
```

create_preroute_vias

Creates vias between specified layers.

SYNTAX

```
status create_preroute_vias
[-nets {collection_of_nets}]
[-buses list_of_bus_names]
[-from_layer number_or_name]
[-to_layer number_or_name]
[-object_shapes {collection_of_shapes}]
[-from_object_strap]
[-from_object_user]
[-from_object_std_pin]
[-from_object_std_pin_connection]
[-from_object_macro_io_pin]
[-from_object_macro_io_pin_connection]
[-from_object_ring]
[-from_object_bus]
[-to_object_strap]
[-to_object_user]
[-to_object_std_pin]
[-to_object_std_pin_connection]
[-to_object_macro_io_pin]
[-to_object_macro_io_pin_connection]
[-to_object_ring]
[-to_object_bus]
[-connect_to_targets_on_all_layers_in_between]
[-ignore_parallel_targets]
[-do_not_merge_targets]
[-special_via_rule]
[-special_via_x_offset distance]
[-special_via_y_offset distance]
[-offset_both_sides_for_special_via]
[-special_via_x_size distance]
[-special_via_y_size distance]
[-special_via_x_step distance]
[-special_via_y_step distance]]
[-within {{llx lly} {urx ury}}]
[-advanced_via_rules]
[-optimize_via_locations]
[-undo]
[-
mark_as strap | user_defined | standard_cell_pin_connection | macro_io_pin_connecti
on | ring | bus]
```

Data Types

<i>collection_of_nets</i>	collection
<i>list_of_bus_names</i>	list
<i>number_or_name</i>	string
<i>collection_of_shapes</i>	collection
<i>distance</i>	distance

create_preroute_vias

ARGUMENTS

-nets *{collection_of_nets}*

Specifies the nets on which to create vias. If you specify more than one value, separate the values with a space. If you do not use the **-nets** or **-buses** option, the command creates vias on the power and ground nets.

-buses *list_of_bus_names*

Specifies the names of the bus nets on which to create vias. If you specify more than one name, separate the names with a comma. If you do not use the **-nets** or **-buses** option, the command creates vias on the power and ground nets.

-from_layer *number_or_name*

Specifies the source layer from which vias are dropped. The value can be specified as either a layerNumber or maskName from the technology file. The default is to drop the vias from the first vertical layer.

-to_layer *number_or_name*

Specifies the target layer to which vias are dropped. The value can be specified as either a layerNumber or maskName from the technology file. The default is to target the first horizontal layer.

-object_shapes *{collection_of_shapes}*

Specifies the shapes on which to create vias. If you specify more than one value, separate the values with a space. If you specify this option, all **-from_object_<route_type>** and **-to_object_<route_type>** options will be ignored. The command creates vias between selected shapes on the same net specified in **-nets** or **-buses**, and layer specified in the **-from_layer** and **-to_layer**.

-from_object_strap

Specifies that vias are dropped from straps. Power and ground straps are created by the **create_power_straps** command. Clock straps are created by the **create_clock_mesh** command.

You must specify at least one source object. See the **DESCRIPTION** section for more information about using this option.

-from_object_user

Specifies that vias are dropped from user-defined routes.

You must specify at least one source object. See the **DESCRIPTION** section for more information about using this option.

-from_object_std_pin

Specifies that vias are dropped from standard cell pins.

You must specify at least one source object. See the **DESCRIPTION** section for more information about using this option.

-from_object_std_pin_connection

Specifies that vias are dropped from standard cell pin connections. Standard cell pin connections are created by the **preroute_standard_cells** command.

You must specify at least one source object. See the **DESCRIPTION** section for more information about using this option.

-from_object_macro_io_pin

Specifies that vias are dropped from macro I/O pins.

You must specify as least one source object. See the **DESCRIPTION** section for more information about using this option.

-from_object_macro_io_pin_connection

Specifies that vias are dropped from macro I/O pin connections. Macro I/O pin connections are created by the **preroute_instances** command.

You must specify as least one source object. See the **DESCRIPTION** section for more information about using this option.

-from_object_ring

Specifies that vias are dropped from power rings, ground rings, and pad rings. Rings are created by the **create_rectangular_rings** and **create_pad_rings** command.

You must specify as least one source object. See the **DESCRIPTION** section for more information about using this option.

-from_object_bus

Specifies that vias are dropped from buses.

You must specify as least one source object. See the **DESCRIPTION** section for more information about using this option.

-to_object_strap

Specifies that vias are dropped to straps. Power and ground straps are created by the **create_power_straps** command. Clock straps are created by the **create_clock_mesh** command.

You must specify at least one target object. See the **DESCRIPTION** section for more information about using this option.

-to_object_user

Specifies that vias are dropped to user-defined routes.

You must specify as least one target object. See the **DESCRIPTION** section for more information about using this option.

-to_object_std_pin

Specifies that vias are dropped to standard cell pins.

You must specify at least one target object. See the **DESCRIPTION** section for more information about using this option.

-to_object_std_pin_connection

Specifies that vias are dropped to standard cell pin connections. Standard cell pin connections are created by the **preroute_standard_cell** command.

You must specify at least one target object. See the **DESCRIPTION** section for more information about using this option.

-to_object_macro_io_pin

Specifies that vias are dropped to macro I/O pin.

You must specify at least one target object. See the **DESCRIPTION** section for more information about using this option.

-to_object_macro_io_pin_connection

Specifies that vias are dropped to macro I/O pin connections. Macro I/O pin connections are created by the **preroute_instances** command.

You must specify at least one target object. See the **DESCRIPTION** section for more information about using this option.

-to_object_ring
 Specifies that vias are dropped to power rings, ground rings, and pad rings. Rings are created by the **create_rectangular_rings** and **create_pad_rings** command.
 You must specify as least one target object. See the **DESCRIPTION** section for more information about using this option.

-to_object_bus
 Specifies that vias are dropped to buses.
 You must specify as least one target object. See the **DESCRIPTION** section for more information about using this option.

-connect_to_targets_on_all_layers_in_between
 Drops vias from the source layer to all layers between the source and target layers. By default, vias are dropped only to the target layer.

-ignore_parallel_targets
 Keeps vias from being created between two parallel objects.

-do_not_merge_targets
 Keeps target objects from being merged before vias are created.

-special_via_rule
 Creates a matrix of small via arrays instead of one large via array at wire intersections. Using smaller via arrays could free some routing space. You specify the x- and y-offsets, x- and y-size, and x- and y-step values by using the special via options described below. If the specified values violate design rules, vias are not created.

-special_via_x_offset *distance*
 Specifies the x-offset from the lower-left corner of the wire intersection area where you want to create the first special via. The default is 0.0.

-special_via_y_offset *distance*
 Specifies the y-offset from the lower-left corner of the wire intersection area where you want to create the first special via. The default is 0.0.

-offset_both_sides_for_special_via
 Applies the x- and y- offsets to both the lower-left and upper-right corner of the wire intersection area. By default, the offsets are applied only to the lower-left corner of the wire intersection area.

-special_via_x_size *distance*
 Specifies the horizontal size of the special via arrays. The distance is the x-length in user units. The default is 0.0.

-special_via_y_size *distance*
 Specifies the vertical size of the special via arrays. The distance is the y-length in user units. The default is 0.0.

-special_via_x_step *distance*
 Specifies the horizontal distance between the left sides of adjacent via arrays. The distance is the x-length in user units. The default is 0.0.
 A via is not created if it would cause DRC violations.

-special_via_y_step distance
 Specifies the vertical distance between the bottom sides of adjacent via arrays. The distance is the y-length in user units. The default is 0.0. A via is not created if it would cause DRC violations.

-within {{llx lly} {urx ury}}
 Specifies the rectangular area within which to create vias. A rectangle is a collection of two points that specifies the lower-left and upper-right corners. A point is a collection of x- and y- coordinates. The value to substitute for rectangle is in the form {{llx lly} {urx ury}}, where llx is the x-coordinate of the lower-left corner and lly is its y-coordinate, and urx is the x-coordinate of the upper-right corner and ury is its y-coordinate. By default, the top cell boundary is used.

-advanced_via_rules
 Uses advanced via rules that have been previously set by the **set_preroute_advanced_via_rule** command.

-optimize_via_locations
 Saves more tracks for the detail routing when it can adjust vias by small horizontal or vertical distances.

-undo
 Removes vias created by the last **create_preroute_vias** run.

-mark_as strap | user_defined | standard_cell_pin_connection | macro_io_pin_connection | ring | bus
macro_io_pin_connection | ring | bus" Marks the created vias with the specified attribute. Valid values are strap, user_defined, standard_cell_pin_connection, macro_io_pin_connection, ring, and bus. The default is the same type as the source object of the via. You can use this option to override the default type.

DESCRIPTION

This command creates vias between specified layers on either power and ground nets, specified nets, or specified bus nets. You can change the internal application of DRC rules by using the **set_preroute_drc_strategy** command.

You must specify at least one source object type and one target object type.

You use the following options to specify the source object type:

-from_object_strap
-from_object_user
-from_object_std_pin
-from_object_std_pin_connection
-from_object_macro_io_pin
-from_object_macro_io_pin_connection
-from_object_ring
-from_object_bus

You use the following options to specify the target object type:

```

-to_object_strap
-to_object_user
-to_object_std_pin
-to_object_std_pin_connection
-to_object_macro_io_pin
-to_object_macro_io_pin_connection
-to_object_ring
-to_object_bus

```

You can combine any number of source and target object type options as long as you use at least one source and one target option.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates vias with nets VDDC and VSSC, between strap objects on layers M2 and M3, within the boundaries of a rectangle with lower-left corner coordinates of {10.0 10.0} and upper-right corner coordinates of {1400.0 1400.0}. The created vias have a type of user_defined.

```

prompt> create_preroute_vias \
-nets {VDDC VSSC} \
-from_layer M2 \
-to_layer M3 \
-to_object_strap \
-from_object_strap \
-within {{10.0 10.0} {1400.0 1400.0}}
-mark_as user_defined

```

SEE ALSO

```

create_pad_rings(2)
create_power_straps(2)
create_rectangular_rings(2)
preroute_instances(2)
preroute_standard_cells(2)
report_preroute_drc_strategy(2)
set_preroute_advanced_via_rule(2)
set_preroute_drc_strategy(2)
verify_pg_nets(2)

```

create_pst

Creates a power state table (PST), using a specific order of supply nets.

SYNTAX

```
string create_pst  
table_name  
-supplies list
```

Data Types

<i>table_name</i>	string
<i>list</i>	list

ARGUMENT

table_name
Specifies the name of the power state table.
This argument is required.

-supplies *list*
Specifies a list of supply nets or ports to include in each power state table in the design.
This option is required.

DESCRIPTION

The **create_pst** command creates a power state table using a specific order of supply nets. A power state table is used for implementation specifically for synthesis, analysis, and optimization. The power state table defines the legal combinations of states, which are those combinations of states that can exist at the same time during the operation of the design.

The power state table has no simulation semantics. It is tool-dependent as to whether the simulation tools report an error if an illegal (unspecified) combination of states occurs.

An error occurs if a specified supply net has not been created before the **create_pst** command is run.

This command returns the name of the power state table if it is created, or the null string if the power state table is not created.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following is an example of the **create_pst** command:

```
prompt> create_pst MyPowerStateTable -supplies {PN1 PN2 SOC/OTC/PN3}
```

SEE ALSO

```
add_port_state(2)  
add_pst_state(2)  
report_pst(2)
```

create_qor_snapshot

Creates a QoR snapshot of timing, physical, constraints, clock, and power data on active scenarios and stores it in the location specified by the **icc_snapshot_storage_location** variable.

NOTE: This is the new version of the **create_qor_snapshot** command that supports both multicorner-multimode designs as well as non-multicorner-multimode designs.

SYNTAX

create_qor_snapshot

```
[-name name]  
[-power]  
[-clock_tree]  
[-show_all]  
[-save_mw]  
[-significant_digits digits]
```

ARGUMENTS

-name *name*
Specifies the name of the QoR snapshot. The snapshot name should start with a letter. It can be used to identify the particular QoR snapshot.

-power
Records dynamic and static power (optional).

-clock_tree
Records clock tree information (optional).

-show_all
Show all clock groups, default only show clock groups that are violated (optional).

-save_mw
Save design using snapshot name (optional).

DESCRIPTION

The **create_qor_snapshot** command generates and report data of timing, DRC, area, clock, power, etc on active scenarios.

Multicorner-Multimode Support

This command uses information from active scenarios only.

EXAMPLES

In the following example, **create_qor_snapshot** will record the QoR snapshot named *preroute* from the current design status and save it under design/snapshot directory.

```
create_qor_snapshot  
516
```

```
prompt> create_qor_snapshot -name preroute
```

SEE ALSO

```
remove_qor_snapshot(2)  
report_qor_snapshot(2)  
icc_snapshot_storage_location(3)
```

create_qtm_clock

Creates a quick timing model (QTM) clock port.

SYNTAX

```
string create_qtm_clock  
-type clock_type  
clock_list
```

Data Types

```
clock_type      string  
clock_list      list
```

ARGUMENTS

```
-type clock_type  
    Specifies the type of clock. The clock can be one of the following types:  
    clock or generated_clock. This will help in defining an existing port or  
    creating an internal pin as a clock port.  
  
clock_list  
    Specifies the list of QTM clock ports that you want to create.
```

DESCRIPTION

This command creates a QTM clock port. You can create a single clock port or a list of clock ports with this command. If there is an existing **create_qtm_port** command with the same port name, then the clock port will have the same direction as the previously created port. If there no existing port with the same clock port name, then a new port definition will be created with the direction as internal.

Please note that this command doesn't create a clock definition. It just creates an internal clock pin or designates an existing QTM port as clock port.

To show the information about the current QTM model, use **report_qtm_model** command.

For a basic description about QTM, please refer to **create_qtm_model** man page. For a more detailed description about QTM, please refer to the *ICC Design Planning User Guide*.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example creates a clocked QTM port CLK.

```
create_qtm_clock  
518
```



```
prompt> create_qtm_clock CLK -type clock
```

The following example creates QTM output clock port C.

```
prompt> create_qtm_port {C} -type output
```

```
prompt> create_qtm_clock {C} -type clock
```

The following example creates QTM internal generated clock pin C_int.

```
prompt> create_qtm_clock C_int -type generated_clock
```

SEE ALSO

```
create_qtm_model(2)  
report_qtm_model(2)  
save_qtm_model(2)  
set_qtm_port_drive(2)  
set_qtm_port_load(2)
```

create_qtm_constraint_arc

Creates a constraint arc for a quick timing model (QTM).

SYNTAX

```
string create_qtm_constraint_arc  
[-name arc_name]  
[-setup] [-hold]  
-from port_name [-to port_spec]  
-edge triggering_edge  
[-path_type name]  
[-path_factor multiplication_factor]  
[-value constraint_value]  
[-input_transition_fall ftrans]  
[-input_transition_rise rtrans]
```

Data Types

<i>arc_name</i>	string
<i>port_name</i>	string
<i>port_spec</i>	list
<i>triggering_edge</i>	string
<i>name</i>	string
<i>multiplication_factor</i>	float
<i>constraint_value</i>	float

ARGUMENTS

-name *arc_name*
Specifies the name of the constraint arc.

-setup
Creates a setup arc.

-hold
Creates a hold arc.

-from *port_name*
Specifies the constraining port name. Define this port as a clock port.

-to *port_spec*
Specifies the constrained port. Define this port as an input/inout port.

-edge *triggering_edge*
Specifies the triggering edge of the clock (rise/fall).

-path_type *name*
Specifies the path type you want to use to set the delay.

-path_factor *multiplication_factor*
Specifies the multiplication factor for the path type.

`-value constraint_value`

Specifies the delay value in terms of the time units you use for the model.

DESCRIPTION

This command allows you to create a constraint arc in a QTM. The **-from** port denotes the constraining port, and the **-to** port denotes the constrained port. The **-from** port must be defined as a clock port and the **-to** port must be an input/inout port.

The **-setup** and **-hold** switches are used to specify either a setup arc or a hold arc. You must use one of the switches and cannot use both simultaneously. To specify the triggering edge of the clock, use the **-edge** option. The option takes rise/fall as valid values.

To specify the delay value, use **-path_type**. You can specify the delay as a value in terms of the time units used in the design. Use either the **-path_type** or the **-value** option; you cannot use both simultaneously.

If the constraint arc is a setup arc, the QTM parameter global setup time is added to the value of the specified setup. If the constraint arc is a hold arc, the global hold time is added to the value of hold specified. You must define the global setup time before creating any setup arc and global hold time before creating any hold arc. The command checks to see if these parameters are defined. Global setup and time, and global hold time are defined using the **set_qtm_global_parameter** command.

To see the information about the current QTM model, use **report_qtm_model** command.

For basic description about QTM, please refer to **create_qtm_model** man page. For a more detailed description about QTM please refer to the *ICC Design Planning User Guide*.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following command creates a setup arc from the positive edge of constraining pin CLK to IN1 (constrained pin) with a delay equivalent to 2 times that of path type path1.

```
prompt> create_qtm_constraint_arc -setup -edge rise -from CLK -to IN1 -  
path_type path1 -path_factor 2
```

The following command creates a hold arc from the positive edge of constraining pin CLK to IN1 (constrained pin) with a delay equivalent to 2 times that of path type path1.

```
prompt> create_qtm_constraint_arc -hold -edge rise -from CLK -to IN1 -  
path_type path1 -path_factor 2
```

The following command creates a hold arc from the positive edge of constraining pin

CLK to IN1 (constrained pin) with a delay of 3 time units.

```
prompt> create_qtm_constraint_arc -hold -edge rise -from CLK -to IN1 -value  
3.0
```

SEE ALSO

```
create_qtm_delay_arc(2)  
create_qtm_model(2)  
create_qtm_path_type(2)  
save_qtm_model(2)  
report_qtm_model(2)
```

create_qtm_delay_arc

Creates a delay arc for a quick timing model (QTM).

SYNTAX

```
string create_qtm_delay_arc
[-name arc_name]
-from port_spec
-to port_spec
[-edge triggering_edge]
[-path_type path_type]
[-path_factor multiplication_factor]
[-value delay_value]
[-total_value total_value]
[-input_transition_fall ftrans]
[-input_transition_rise rtrans]
```

Data Types

<i>arc_name</i>	string
<i>port_spec</i>	list
<i>path_type</i>	string
<i>multiplication_factor</i>	float
<i>delay_value</i>	float
<i>total_value</i>	float

ARGUMENTS

-name *arc_name*
Specifies the name of the delay arc.

-from *port_spec*
Specifies the QTM port name or a collection of QTM ports, which is the start point of the delay arc. For an edge triggering arc this must be a clock.

-to *port_spec*
Specifies the QTM port name or a collection of QTM ports. This port must be output/inout type.

-edge *triggering_edge*
Specifies the triggering edge (rise or fall)

-path_type *path_type*
Specifies the path type you want to use to set the delay.

-path_factor *multiplication_factor*
Specifies the multiplication factor for the path type.

-value *delay_value*
Specifies the delay value in terms of the time units you use for the model. By default, the drive arc delay is added on top of the QTM delay arcs created to the output port and therefore increase the actual delay.

`-total_value total_value`

Specifies the total value in terms of the time units you use for the model. This option indicates that the drive arc delay computed with max load is to be subtracted from the total delay budgeted to this output port.

DESCRIPTION

This command allows you to create a delay arc in a QTM from a QTM port or collection of QTM ports to a port or collection of QTM ports. If you specify a list of ports for the **from** and **to** ports, a delay arc is created from each start port to each end port.

To create a edge triggering arc, use the **-edge** option. For an edge triggering arc, the **clk_to_output** delay (specified as a global parameter) is added to the delay value specified.

You can use a **-path_type** to specify the delay value or you can specify the delay as a value in terms of the time units used in the design. Use either the **-path_type** or the **-value** option; you cannot use both options simultaneously.

When delay arc is defined for an output port, by default, the actual delay to the output port will be the delay arc defined by this command for the port plus the delay computed for the drive arc at the output port defined by **create_qtm_port_drive**. If in your QTM creation, the delay defined by **create_qtm_delay_arc** for the port is actually the maximum delay budgeted for the output and you do not want the drive arc to add extra delay on top of that budget, you can optionally use **-total_value**, instead of **-value** to keep the delay with maximum load at port unchanged. But with smaller load at the output port, the delay may actually be smaller than the budgeted maximum delay.

To show the information about the current QTM model, use **report_qtm_model** command.

For a basic description about QTM, please refer to **create_qtm_model** man page. For a more detailed description about QTM, please refer to the *ICC Design Planning User Guide*.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following command creates a delay arc from input ports {A, B, C} to output port D with a delay value of 2.0 time units.

```
prompt> create_qtm_delay_arc -from {A, B, C} -to D -value 2.0
```

The following command creates an arc from clock CLK to output port OUT with a delay equivalent to 3 times that of path type 'path1'.

```
prompt> create_qtm_delay_arc -from CLK -to OUT -path_type path1 -path_factor
3
```

create_qtm_delay_arc

The following example uses wildcard to match all ports matching "CL*".

```
prompt> create_qtm_delay_arc -from CL* -to OUT -path_type path1 -path_factor  
3
```

The following example uses a collection for the QTM port 'CLK'.

```
prompt> create_qtm_delay_arc -from [get_qtm_ports CLK] -to OUT -path_type  
path1 -path_factor 3
```

SEE ALSO

```
create_qtm_constraint_arc(2)  
create_qtm_model(2)  
create_qtm_path_type(2)  
report_qtm_model(2)  
save_qtm_model(2)
```

create_qtm_drive_type

Creates a drive type in a quick timing model (QTM).

SYNTAX

```
string create_qtm_drive_type  
-lib_cell lib_cell_name  
[-input_pin pin_name]  
[-output_pin pin_name]  
[-input_transition_rise rtrans]  
[-input_transition_fall ftrans]  
drive_type_name  
[-lib_cell_input_transition lib_cell_input_transition]
```

Data Types

<i>lib_cell_name</i>	string
<i>rtrans</i>	float
<i>ftrans</i>	float
<i>drive_type_name</i>	string

ARGUMENTS

-lib_cell *lib_cell_name*
Specifies the library cell name in the technology library.

-input_pin *pin_name*
Specifies the input pin in the lib_cell is the start point of the arc and ends in the output pin.

-output_pin *pin_name*
Specifies the output pin in the lib_cell that specifies the drive.

-input_transition_rise *rtrans*
Specifies the input rising transition time associated with the **-input_pin** to compute the delay for the drive arc. If you do not include this option, the delay table for rising input of the drive arc will be loaded from library as is.
Use the **-input_transition_rise** and **-input_transition_fall** options to capture the transition time associated with the *input_pin*. This can obtain more accurate information on the transition time and delay time for the defined drive arc.

-input_transition_fall *ftrans*
Specifies the input falling transition time associated with the *input_pin* to compute the delay for the drive arc. If you do not include this option, the delay table for falling input of the drive arc will be loaded from library as is.

drive_type_name
Specifies the name given to the defined drive type.

create_qtm_drive_type

DESCRIPTION

This command can be used to create a drive type. The defined drive type can be used to set the drives on output ports. The drive type is specified by referring to a `lib_cell` in the technology library. The `lib_cell` is specified by using the **-lib_cell** option. This `lib_cell` must be present in the technology library.

The output pin which drives the port is specified by using the **-output_pin** option. In addition, you can specify the input pin by using the **-input_pin** option. This option selects an arc that drives the output pin of the `lib_cell`.

If neither the input pin nor the output pin is specified, an arbitrary arc is chosen to define the drive type. If the input pin is specified and output pin is not specified, an arbitrary arc starting from the input pin defines the drive type. If the output pin is specified and input pin is not specified, the arbitrary delay arc coming into the output pin defines the drive type. If both input and output pins are specified, the `lib_cell` must have a combinational arc from the input pin to the output pin.

To use this command you must read in the technology library and then specify it by using the **set_qtm_technology** command.

To show the information about the current QTM model, use **report_qtm_model** command.

For a basic description QTM, see the **create_qtm_model** man page. For a more detailed description, see Chapter 12 of the *ICC Design Planning User Guide*.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

In the following examples it is assumed that you have loaded the technology library, and the QTM technology is set. Do this by using the following the sequence of commands as a guide, substituting the name of your technology library.

```
prompt>read_db my_technology_library.db
```

```
prompt>set_qtm_technology -library my_technology_library
```

The following command creates a drive type equivalent to the `lib_cell` **BUF1**.

```
prompt> create_qtm_drive_type -lib_cell BUF1 drive1
```

The following command creates a drive type **drive2** equivalent to `lib_cell` **BUF2** and specifies the output pin to use.

```
prompt> create_qtm_drive_type -lib_cell BUF2 -output_pin Y drive2
```

SEE ALSO

`create_qtm_load_type(2)`

```
create_qtm_model(2)
create_qtm_path_type(2)
report_qtm_model(2)
save_qtm_model(2)
set_qtm_port_drive(2)
set_qtm_technology(2)
```

create_qtm_generated_clock

Creates a generated clock for a quick timing model (QTM).

SYNTAX

```
string create_qtm_generated_clock  
-source master_clock_name  
[-divide_by divide_factor  
 | -multiply_by multiply_factor]  
[-invert]  
generated_clock_name
```

Data Types

<i>master_clock_name</i>	string
<i>divide_factor</i>	int
<i>multiply_factor</i>	int
<i>generated_clock_name</i>	string

ARGUMENTS

-source *master_clock_name*
Specifies the name of the clock defined as master source of the generated clock. The master source must be defined as a clock, or a generated clock, prior to the *generated_clock_name* definition.

-divide_by *divide_factor*
Specifies the frequency division factor. If the *divide_factor* value is 2, the generated clock period is twice as long as the master clock period.

-multiply_by *multiply_factor*
Specifies the frequency multiplication factor. If the *multiply_factor* value is 3, the generated clock period is one-third as long as the master clock period.

-invert
Inverts the generated clock signal (in the case of frequency multiplication and division).

generated_clock_name
Specifies the name of the generated clock. If the name has been used in defining a port or internal pin, then the *generated_clock* will be defined on the existing port or internal pin. Otherwise, a new same-named internal pin will be created to be the source of the generated clock.

DESCRIPTION

This command creates QTM generated clock. A generated clock can be defined on the external port of the QTM, as well as internal pin in the QTM. After definition of the generated clock, the source port/pin can be used in other QTM commands the same way as those port/pins defined by **create_qtm_port**, i.e. delay arcs from or to the

generated clock source pin can be defined with **create_qtm_delay_arc**; constraint arcs related to the generated clock can be defined with **create_qtm_constraint_arc**.

For a basic description about QTM, please refer to **create_qtm_model** man page. For a more detailed description about QTM, please refer to the *ICC Design Planning User Guide*.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example creates a divide-by 2 generated clock on an internal pin, assume there is no port defined with name "GCLK_int", with master clock CLK.

```
prompt> create_qtm_generated_clock GCLK_int -source CLK -multiply_by 2
```

The following example creates a generated clock on port GCLK, assume GCLK has already been defined as a port with command **create_qtm_port**.

```
prompt> create_qtm_generated_clock GCLK -source CLK -divide_by 2 -invert
```

SEE ALSO

```
create_qtm_model(2)  
report_qtm_model(2)  
save_qtm_model(2)  
create_qtm_delay_arc(2)  
create_qtm_constraint_arc(2)
```

create_qtm_insertion_delay

Specifies the insertion delay on the clock port of a quick timing model (QTM).

SYNTAX

```
string create_qtm_insertion_delay  
[-max]  
[-min]  
[-value insertion_delay]  
port_list
```

Data Types

<i>insertion_delay</i>	float
<i>port_list</i>	list

ARGUMENTS

-max
Specifies whether to generate insertion delay for Maximum operating condition.

-value *insertion_delay*
Specifies the insertion delay in terms of the time units you use for the model.

port_list
Specifies a list of clock ports on which to set internal clock latency. Each element in the list is either a collection of QTM ports or a pattern that matches QTM ports.

DESCRIPTION

This command creates the insertion delay for a clock port on the QTM model. The insertion delay is the network latency from the clock port to a (hypothetical) FlipFlop inside the QTM model.

If the CTS has been done on the design, then there would be network latency for all the clock pins of the FlipFlops. If some of the hierarchical blocks in the design are replaced by blackboxes, then the user can generate QTM models for the blackboxes to represent their timing model.

If the user generates the QTM models for the blackboxes, then *create_qtm_insertion_delay* command can be called to set the network delay from clock port of the hierarchical block to FlipFlop inside the hierarchical block, as the insertion delay.

To show the information about the current QTM model, use the **report_qtm_model** command.

For a basic description about QTM, see the **create_qtm_model** man page. For a more

detailed description about QTM, see the design planning chapter of the *IC Compiler User Guide*.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following command sets the insertion delay on CLK to be 2.0 for Max mode.

```
prompt> create_qtm_insertion_delay -max -value 2.0 CLK
```

The following command sets the insertion on CK to be 4.5 for both Max and Min modes.

```
prompt> create_qtm_insertion_delay -value 4.5 CK
```

SEE ALSO

```
create_qtm_model(2)  
create_qtm_port(2)  
report_qtm_model(2)  
save_qtm_model(2)  
create_qtm_clock(2)
```

create_qtm_load_type

Creates a load type for a quick timing model (QTM) description.

SYNTAX

```
string create_qtm_load_type  
-lib_cell name  
[-input_pin pin_name]  
load_type name
```

ARGUMENTS

-lib_cell *name*
Specifies the name of the library cell in the technology library.

-input_pin *pin_name*
Specifies the input pin in the lib_cell to specify capacitance.

load_type name
Specifies the name given to the defined load type.

DESCRIPTION

This command is used to create a load type. A load type can be used to define the load on a QTM input port. The load type is specified by referring to a lib_cell in the technology library. The **lib_cell** is specified using the **-lib_cell** option. **lib_cell** must be present in the specified technology library.

The input pin can be specified by using the **-input_pin** option. If the input pin is not specified, an arbitrary input pin is chosen to define the load type.

To use this command you must first read in the technology library, then specify the technology library by using the **set_qtm_technology** command.

To show the information about the current QTM model, use **report_qtm_model** command.

For a basic description about QTM, please refer to **create_qtm_model** man page. For a more detailed description about QTM, please refer to Chapter 12 of the *ICC Design Planning User Guide*.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

In the following examples the technology library is loaded by you and the QTM technology is set. This is done by the following sequence of commands:

```
prompt>read_db my_technology_library.db
```

```
prompt>set_qtm_technology -library my_technology_library
```

The following command creates a load type 'load1' using cell OR1. Since the input pin name is not specified, QTM selects an arbitrary input pin.

```
prompt> create_qtm_load_type -lib_cell OR1 load1
```

The following command creates a load type 'load2' using cell OR1, input pin A.

```
prompt> create_qtm_load_type -lib_cell OR1 -input_pin A load2
```

SEE ALSO

```
create_qtm_drive_type(2)  
create_qtm_model(2)  
create_qtm_path_type(2)  
report_qtm_model(2)  
save_qtm_model(2)  
set_qtm_port_load(2)
```


create_qtm_model

Begins the definition of a quick timing model (QTM) description.

SYNTAX

```
string create_qtm_model  
model_name
```

Data Types

```
model_name      string
```

ARGUMENTS

```
model_name  
    Specifies the name of the generated model.
```

DESCRIPTION

Specifies the name of a new quick timing model (QTM) to be created by ICC Design Planning. QTMs are temporary timing models that can be created quickly for a block using ICC Design Planning commands. The purpose of these commands is to provide timing information without the necessity of writing a detailed timing model.

QTMs are intended to be used early in the design cycle to describe rough initial timing of a block. These models will eventually be replaced, either by some other detailed timing model or by the netlist of the block, to obtain more accurate timing. For a more detailed description of QTMs, see the *ICC Design Planning User Guide*.

A QTM model can be saved as a Synopsys DB file, which can be instantiated in a design in the same way library cells or ITS models are instantiated. Both Design Compiler and JupiterXT accept designs with instantiated QTMs. To save a QTM model, use the **save_qtm_model** command.

Other commands in the QTM command set must be placed between a **create_qtm_model** and **save_qtm_model** command. Many QTM commands exist for specifying, configuring and examining QTMs. The following is not an exhaustive list, but gives examples of tasks you can perform using QTM commands:

- To create a QTM port, use the **create_qtm_port** command.
- To create constraint arcs and delay arcs, use the **create_qtm_constraint_arc** and the **create_qtm_delay_arc** commands, respectively.
- To set the drive of a QTM output port, use the **set_qtm_port_drive** command.

- To set the load of a QTM input port, use **set_qtm_port_load** command.
- To show the information about the current QTM model, use the **report_qtm_model** command.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following command creates a new QTM model with the name adder.

```
prompt> create_qtm_model adder
```

SEE ALSO

```
create_qtm_constraint_arc(2)
create_qtm_delay_arc(2)
create_qtm_drive_type(2)
create_qtm_load_type(2)
create_qtm_path_type(2)
create_qtm_port(2)
report_qtm_model(2)
save_qtm_model(2)
set_qtm_global_parameter(2)
set_qtm_port_drive(2)
set_qtm_port_load(2)
set_qtm_technology(2)
```

create_qtm_path_type

Creates a path type in a quick timing model (QTM) description.

SYNTAX

```
string create_qtm_path_type  
-lib_cell name  
[-input_pin pin_name]  
[-output_pin pin_name]  
[-fanout count]  
path_type name
```

Data Types

<i>name</i>	string
<i>pin_name</i>	string
<i>count</i>	integer-range

ARGUMENTS

-lib_cell *name*
Specifies the name of the library cell in the technology library.

-input_pin *pin_name*
Specifies the input pin in the lib_cell, which is the start point of the arc, to define the path type.

-output_pin *pin_name*
Specifies the output pin in the lib_cell, which is the end point of the arc, to define the path type.

-fanout *count*
Specifies the average fanout (number of pins) to consider while computing the delay of the path type. The fanout count can range from 1 to 1000. By default this is 1.

path_type name
Specifies the name given to the defined path type.

DESCRIPTION

This command is used to create a path type. A path type mimics an arc in a library cell. An arc in the QTM model is defined in terms of the delays of the path type. The path type is specified by referring to a lib_cell in the technology library. The lib_cell is specified using the **-lib_cell** option. This lib_cell must be present in the technology library specified.

You can specify the average fanout count the arc fans out to while defining the path type. It is assumed the arc fans out to the first input pin of the same library cell, while calculating the delays. The input pin is specified by using the **-input_pin** option. The output pin is specified by using the **-output_pin** option.

If neither input pin nor the output pin is specified, an arbitrary delay arc in the `lib_cell` is chosen to define the path type.

If the input pin is specified and output pin is not specified, an arbitrary delay arc starting from the input pin defines the path type.

If the output pin is specified and input pin is not specified, an arbitrary delay arc coming into the output pin defines the path type.

If both input and output pins are specified, the `lib_cell` must have an arc from the input pin to the output pin.

To use this command you must read in the technology library then specify the technology library by using the **`set_qtm_technology`** command.

To show the information about the current QTM model, use **`report_qtm_model`** command.

For a basic description about QTM, please refer to **`create_qtm_model`** man page. For a more detailed description about QTM, please refer to Chapter 12 of the *ICC Design Planning User Guide*.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

In the following examples the technology library is loaded by the user and the qtm technology is set. This is done by the following sequence of commands:

```
prompt>read_db my_technology_library.db
prompt>set_qtm_technology -library my_technology_library
```

The following command creates a path type `path1` by using `lib_cell AN2`, with an average fanout count of 2 (uses default input, output pins).

```
prompt> create_qtm_path_type -lib_cell AN2 -fanout 2 path1
```

The following command creates a path type `path2` by using cell `AN2`, with an average fanout count of 2, and using pin 'A' as the starting point for the path type.

```
prompt> create_qtm_path_type -lib_cell AN2 -input_pin A -fanout 2 path2
```

The following command creates a path type `path3` by using cell `AN2`, with an average fanout count of 2, using 'A' as the start point of the arc, and pin 'Z' as the end point of the arc.

```
prompt> create_qtm_path_type -lib_cell AN2 -input_pin A -output_pin Z -fanout
2 path3
```

SEE ALSO

`create_qtm_constraint_arc(2)`

`create_qtm_path_type`

```
create_qtm_delay_arc(2)
create_qtm_drive_type(2)
create_qtm_load_type(2)
create_qtm_model(2)
report_qtm_model(2)
save_qtm_model(2)
```

create_qtm_port

Creates a quick timing model (QTM) port.

SYNTAX

```
string create_qtm_port  
-type port_type  
port_list
```

Data Types

```
port_type    string  
port_list    list
```

ARGUMENTS

```
-type port_type  
    Specifies the type of port. The port can be one of the following types: input,  
    output, inout, internal or clock. If you want a port to be a clock, define  
    it as a clock port.  
  
port_list  
    Specifies the list of QTM ports you want created.
```

DESCRIPTION

This command creates QTM port. You can create a single port or a list of ports with this command. The ports can be input, output, inout, internal or clock. Any port that constrains another port or has a edge triggered launch arc originating from it, has to be defined to be of the type 'clock'.

Bused ports are also defined by giving the start and end index (A[0:5]). Bused ports cannot be internal.

To show the information about the current QTM model, use **report_qtm_model** command.

For a basic description about QTM, please refer to **create_qtm_model** man page. For a more detailed description about QTM, please refer to the *ICC Design Planning User Guide*.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example creates an input bused QTM port A[0:5].

```
prompt> create_qtm_port A[0:5] -type input
```

The following example creates a clocked QTM port CLK.

```
prompt> create_qtm_port CLK -type clock
```

The following example creates QTM output ports A, B, C, and D.

```
prompt> create_qtm_port {A B C D} -type output
```

The following example creates QTM internal pin D_int.

```
prompt> create_qtm_port D_int -type internal
```

SEE ALSO

```
create_qtm_model(2)  
report_qtm_model(2)  
save_qtm_model(2)  
set_qtm_port_drive(2)  
set_qtm_port_load(2)
```

create_rail_setup

Creates a setup file and writes out necessary files for performing rail analysis in PrimeRail.

SYNTAX

```
status create_rail_setup
[-sdc sdc_file]
[-spef spef_file]
[-verilog verilog_file]
[-no_rc_extract]
[-hierarchy]
[-directory dir_name]
[-parasitic_corner min | max]
[-no_save]
[-upf string]
```

Data Types

<i>sdc_file</i>	string
<i>spef_file</i>	string
<i>verilog_file</i>	string
<i>dir_name</i>	string

ARGUMENTS

-sdc *sdc_file*

Specifies an existing SDC file to use.

If you do not specify both this option and the **-hierarchy**, the command generates an SDC file for the current Milkyway design.

-spef *spef_file*

Specifies an existing SPEF file to use.

If you do not specify both this option and the **-hierarchy**, the command performs RC extraction and generates an SPEF file for the maximum corner of the current Milkyway design.

If RC extraction has already been done, use the **-no_rc_extract** option to skip RC extraction.

To generate an SPEF file for the minimum corner instead of the maximum corner, use the **-parasitic_corner min** option.

-verilog *verilog_file*

Specifies an existing Verilog file to use.

If you do not specify this option, the command generates a Verilog file for the current Milkyway design.

-no_rc_extract

Skips RC extraction.

Use this option to avoid rerunning extraction if you have already run the **extract_rc** command and the parasitics are in the memory.

-hierarchy

Generates only the Verilog, UPF, and PrimeRail setup files for the top-level design in a hierarchical flow. This option is used for the full-chip sign-off flow. You must provide the signal parasitics and the SDC file.

-directory *dir_name*

Specifies the output directory where the files required for rail analysis are output.

If you do not specify this option, the command puts the files in a directory called `synopsys_rail_setup`.

-parasitic_corner min | max

Specifies the parasitic corner used for generating the SPEF file.

If you do not specify this option, the command uses the maximum corner to generate the SPEF file.

-no_save

Do not save the Milkyway design.

By default, the command saves the design in a cell called *design_name_pr*.

DESCRIPTION

The **create_rail_setup** command creates the files necessary for running PrimeRail. By default, the command writes the following files to the specified output directory (the default directory is `synopsys_rail_setup`):

```
synopsys_pr_setup.e
design_name.v
design_name.spef
design_name.sdc
design_name.upf
```

Note that the UPF file is generated only if the tool is running in UPF mode.

The *design_name* prefix used in the file names is the return value of the *current_design* command in `icc_shell`. For example, if *current_design* returns "top" in `icc_shell`, the Verilog output file name is "top.v".

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLE

The following example runs the **create_rail_setup** command with no options:

```
prompt> create_rail_setup
```

The following example skips RC extraction.

```
prompt> create_rail_setup -no_rc_extract
```

SEE ALSO

`extract_rc(2)`

create_rectangular_rings

Creates rectangular rings in the design.

SYNTAX

```
integer create_rectangular_rings
[-
around core | specified | specified_as_group | all_macros_except_specified | rectan
gle]
[-cells {collection_of_cells}]
[-within {{llx lly} {urx ury}}]
-nets {collection_of_nets}
[-skip_left_side]
[-skip_right_side]
[-skip_bottom_side]
[-skip_top_side]
[-left_segment_layer number_or_name]
[-right_segment_layer number_or_name]
[-bottom_segment_layer number_or_name]
[-top_segment_layer number_or_name]
[-left_segment_width distance]
[-right_segment_width distance]
[-bottom_segment_width distance]
[-top_segment_width distance]
[-offsets adjusted | absolute]
[-left_offset distance]
[-right_offset distance]
[-bottom_offset distance]
[-top_offset distance]
[-extend_ll]
[-extend_lh]
[-extend_rl]
[-extend_rh]
[-extend_bl]
[-extend_bh]
[-extend_tl]
[-extend_th]
[-create_innermost_core_ring_conservatively]
[-ignore_parallel_targets]
[-advanced_via_rules]
[-extend_for_multiple_connections]
[-extension_gap distance]
-undo
```

Data Types

<i>collection_of_cells</i>	collection
<i>collection_of_nets</i>	collection
<i>number_or_name</i>	string
<i>distance</i>	distance

ARGUMENTS

`-around core | specified | specified_as_group | all_macros_except_specified | rectangle`

Defines the area for which to specify rings. The valid values are defined as follows:

core creates rings around the predefined core area.

specified creates rings around instances specified by using the **-cells** option.

specified_as_group creates rings around instances specified by using the **-cells** option, treating all the instances as one group.

all_macros_except_specified creates rings around all macros except those specified by using the **-cells** option. If you do not use the **-cells** option, the **all_macros_except_specified** option creates rings around all macros in the design.

rectangle creates rings around the rectangle specified by using the **-within** option.

You must select only one of these values. The default is **core**.

`-cells {collection_of_cells}`

Specifies a collection of instances for the **-around** option when it is used with the **specified**, **specified_as_group**, or **all_macros_except_specified** value.

`-within {{llx lly} {urx ury}}`

Specifies the rectangular area within which to create rings. A rectangle is a collection of two points that specifies the lower-left and upper-right corners. A *point* is a collection of X and Y coordinates. The value to substitute for *rectangle* is in the form `{{llx lly} {urx ury}}`, where *llx* is the X coordinate of the lower-left corner and *lly* is its Y coordinate, and *urx* is the X coordinate of the upper-right corner and *ury* is its Y coordinate. By default, the top cell boundary is used.

`-nets {collection_of_nets}`

Specifies nets to create rectangular rings. If you have more than one net in collection, separate each one by a space.

`-skip_left_side`

Specifies not to create the left side of the ring.

`-skip_right_side`

Specifies not to create the right side of the ring.

`-skip_bottom_side`

Specifies not to create the bottom side of the ring.

`-skip_top_side`

Specifies not to create the top side of the ring.

`-left_segment_layer number_or_name`

Specifies the layer on which you want the left section of the ring. The value can be specified either as *layerNumber* or *maskName* from the technology file. By default, the first vertical layer is used.

`-right_segment_layer number_or_name`
 Specifies the layer on which you want the right section of the ring. The value can be specified either as `layerNumber` or `maskName` from the technology file. By default, the first vertical layer is used.

`-bottom_segment_layer number_or_name`
 Specifies the layer on which you want the bottom section of the ring. The value can be specified either as `layerNumber` or `maskName` from the technology file. By default, the first horizontal layer is used.

`-top_segment_layer number_or_name`
 Specifies the layer on which you want the top section of the ring. The value can be specified either as `layerNumber` or `maskName` from the technology file. By default, the first horizontal layer is used.

`-left_segment_width distance`
 Specifies the width of the left section of the ring. By default, the **minWidth** for layer is used.

`-right_segment_width distance`
 Specifies the width of the right section of the ring. By default, the **minWidth** for layer is used.

`-bottom_segment_width distance`
 Specifies the width of the bottom section of the ring. By default, the **minWidth** for layer is used.

`-top_segment_width distance`
 Specifies the width of the top section of the ring. By default, the **minWidth** for layer is used.

`-offsets adjusted | absolute`
 Selects a setting for the ring offsets. Valid arguments are **adjusted** and **absolute**; select only one. The **absolute** value prevents automatic adjustment to correct violations. As a result, there may be a design rule checking (DRC) violation, and the section of the ring causing a violation is not created. The **adjusted** value allows automatic adjustment of the ring position to avoid DRC violations. By default, **adjusted** is used.

`-left_offset distance`
 Specifies the offset on the left side of the ring. By default the offset is **0.0**.

`-right_offset distance`
 Specifies the offset on the right side of the ring. By default the offset is **0.0**.

`-bottom_offset distance`
 Specifies the offset on the bottom side of the ring. By default the offset is **0.0**.

`-top_offset distance`
 Specifies the offset on the top side of the ring. By default the offset is **0.0**.

-extend_ll
 Extends rings to the top cell boundary or to the first target on the same net. Use this option to extend the left segment of the ring low. See the **DIRECTIONS** section for details on all of the directions in which you can extend rings. By default, this option is turned off.

-extend_lh
 Extends rings to the top cell boundary or to the first target on the same net. Use this option to extend the left segment of the ring high. See the **DIRECTIONS** section for details on all of the directions in which you can extend rings. By default, this option is turned off.

-extend_rl
 Extends rings to the top cell boundary or to the first target on the same net. Use this option to extend the right segment of the ring low. See the **DIRECTIONS** section for details on all of the directions in which you can extend rings. By default, this option is turned off.

-extend_rh
 Extends rings to the top cell boundary or to the first target on the same net. Use this option to extend the right segment of the ring high. See the **DIRECTIONS** section for details on all of the directions in which you can extend rings. By default, this option is turned off.

-extend_bl
 Extends rings to the top cell boundary or to the first target on the same net. Use this option to extend the bottom segment of the ring low (left). See the **DIRECTIONS** section for details on all of the directions in which you can extend rings. By default, this option is turned off.

-extend_bh
 Extends rings to the top cell boundary or to the first target on the same net. Use this option to extend the bottom segment of the ring high (right). See the **DIRECTIONS** section for details on all of the directions in which you can extend rings. By default, this option is turned off.

-extend_tl
 Extends rings to the top cell boundary or to the first target on the same net. Use this option to extend the top segment of the ring low (left). See the **DIRECTIONS** section for details on all of the directions in which you can extend rings. By default, this option is turned off.

-extend_th
 Extends rings to the top cell boundary or to the first target on the same net. Use this option to extend the top segment of the ring high (right). See the **DIRECTIONS** section for details on all of the directions in which you can extend rings. By default, this option is turned off.

-create_innermost_core_ring_conservatively
 Places the innermost core ring far enough away from the core boundary to allow cell placement without DRC violations. This option works only if you also specify the **-around** option with the **core** value. You might want to use this option when you are creating core rings before cell placement. The tool scans all the standard cells to determine the maximum distance any cell could extend from the core boundary after placement. That maximum distance is placed in

between the innermost core ring and the core boundary. This space is a conservative estimate and could be more than is actually necessary. Note that macros are ignored in the calculation.

-ignore_parallel_targets

Keeps a ring segment from connecting passing targets if the targets are parallel to the segment. By default, each ring segment connects all passing targets.

-advanced_via_rules

Uses advanced via rules that have been previously set by the command **set_preroute_advanced_via_rule**. By default, this option is turned off.

-extend_for_multiple_connections

Extends ring segments to reach more targets on the same net. The extension continues until either there are no more targets in the respective directions or the next target exceeds the spacing threshold set by the *distance* value of the **extension_gap** option.

-extension_gap *distance*

Specifies the space threshold for multiple connections. Adjacent targets closer than the *distance* value are connected. The default is **0.0**.

-undo

Removes rings created by the last **create_rectangular_rings** run.

DESCRIPTION

This command creates rectangular rings in the design. If a blockage does not allow the tool to place the ring in the area you specified, the tool places the ring in the closest legal position. You have the choice of preventing the tool from automatically adjusting the ring position; in which case, if the ring you specify creates a design rule checking (DRC) error, the tool does not create the segment of the ring causing the violation. You can change the internal application of the DRC rules by using the **set_preroute_drc_strategy** command. Specify at least one net.

As a prerequisite for using the **create_rectangular_rings** command, use the **connect_pg_nets** command to specify port-to-net connections for power and ground nets that are not in the netlist.

You can extend a ring in any or all of several directions to the top cell boundary or to the first target on the same net. Use the following options to extend a ring in the direction described:

- extend_ll** extends the left segment of the ring low.
- extend_lh** extends the left segment of the ring high.
- extend_rl** extends the right segment of the ring low.
- extend_rh** extends the right segment of the ring high.
- extend_bl** extends the bottom segment of the ring low (left).
- extend_bh** extends the bottom segment of the ring high (right).
- extend_tl** extends the top segment of the ring low (left).
- extend_th** extends the top segment of the ring high (right).

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates a rectangular ring with a net named *VDD*, within the boundaries of a rectangle with coordinates *(10.0 10.0) (1400.0 1400.0)*, with left and right sides on layer *M2*, width *0.42*, bottom and top sides on layer *M3*, and width of *0.2*.

```
prompt> create_rectangular_rings -nets VDD \  
-within {{10.0 10.0} {1400.0 1400.0}} \  
-left_segment_layer M2 -right_segment_layer M2 \  
-bottom_segment_layer M3 -top_segment_layer M3 \  
-left_segment_width 0.42 -right_segment_width 0.42 \  
-bottom_segment_width 0.2 -top_segment_width 0.2
```

SEE ALSO

```
create_pad_rings(2)  
create_power_straps(2)  
create_preroute_vias(2)  
preroute_instances(2)  
preroute_standard_cells(2)  
report_preroute_drc_strategy(2)  
set_preroute_drc_strategy(2)  
set_preroute_advanced_via_rule(2)  
verify_pg_nets(2)
```


create_rectilinear_rings

Creates rectilinear rings in the design.

SYNTAX

```
status create_rectilinear_rings
[-nets {collection_of_nets}]
[-around core | macros | all_macros_except_specified]
[-macro_cells {collection_of_macro_cells}]
[-offset {x_offset y_offset}]
[-layers {h_segment_layer_name v_segment_layer_name}]
[-width {width_of_h_segments width_of_v_segments}]
[-space {space_between_h_segments space_between_v_segments}]
[-exclude_instances {collection_of_macro_cells}]
[-extension_of_excluded_instances {x_extension y_extension}]
[-max_deviation distance]
[-create_bridges {offset space}]
[-ignore_parallel_targets]
[-undo]
```

Data Types

<i>collection_of_nets</i>	collection
<i>collection_of_macro_cells</i>	collection
<i>x_offset</i>	distance
<i>y_offset</i>	distance
<i>h_segment_layer_name</i>	string
<i>v_segment_layer_name</i>	string
<i>width_of_h_segments</i>	distance
<i>width_of_v_segments</i>	distance
<i>space_between_h_segments</i>	distance
<i>space_between_v_segments</i>	distance
<i>x_extension</i>	distance
<i>y_extension</i>	distance
<i>distance</i>	space
<i>offset</i>	distance
<i>space</i>	distance

ARGUMENTS

- nets {collection_of_nets}**
Specifies the power and ground nets for which to create rectilinear rings. Rings are created for each net specified in this option. The order of rings corresponds to the order of nets in the collection: the outermost ring is assigned to the first net and the innermost inner ring is assigned to the last net. If you need to create more than one ring for the same net, include that net in the collection as many times as you need. This option is mandatory unless you use the **-undo** option.
- around core | macros | all_macros_except_specified**
Defines the areas around which rings are created. Valid arguments are core, macros, and all_macros_except_specified. You can specify only one of these

arguments. Use **-around core** to create rings around the predefined core area. Use **-around macros** to create rings around the macro cell instances specified in the **-macro_cells** option. Use **-around all_macros_except_specified** to create rings around each macro cell instance in the design, except those specified in the **-macro_cells** option.

The default value is **-around core**.

-macro_cells {*collection_of_macro_cells*}

Specifies a collection of macro cell instances. The meaning of this option depends on the value of the **-around** option:

- If you specify **-around core**, you cannot specify the **-macro_cells** option.
- If you specify **-around macros**, the **-macro_cells** option is required and specifies the macro cell instances around which the rings are created.
- If you specify **-around all_macros_except_specified**, the tool creates rings around all macro cell instances in the design, except those specified in the **-macro_cells** option. If you specify **-around all_macros_except_specified** and do not specify **-macro_cells**, rings are created around all macro cell instances in the design.

-offset {*x_offset y_offset*}

Specifies the offset of the inner boundary of the ring from the boundary of the area around which the rings are created. *x_offset* specifies the offset for vertical segments. *y_offset* specifies the offset for horizontal segments. A negative offset value shifts the rings inside the area, while a positive offset value shifts the rings outside the area.

By default, both the x- and y-offset is 0.

-layers {*h_segment_layer_name v_segment_layer_name*}

Specifies the metal layers on which horizontal and vertical segments of rings are created. By default, the segments are assigned to the lowest metal layers in the design having the corresponding preferred direction.

-width {*width_of_h_segments width_of_v_segments*}

Specifies the width of the horizontal and vertical ring segments. The minimum and maximum widths specified in the technology file override these settings. If the specified width is less than the minimum width specified in the technology file, the tool uses the minimum width from the technology file. If the specified width is greater than the maximum width specified in the technology file, the tool uses the maximum width from the technology file. By default, the width is set to the minimum width of the metal layer, as specified in the technology file.

-space {*space_between_h_segments space_between_v_segments*}

Specifies the minimum spacing between horizontal ring segments and the minimum spacing between vertical ring segments. If the specified spacing is less than the minimum spacing defined in the technology file, the tool uses the minimum spacing from the technology file.

By default, the values are set to the corresponding minimum spacing defined in the technology file.

-exclude_instances {*collection_of_macro_cells*}

Specifies the macro cell instances to be excluded from the rings. If an instance boundary intersects at least one ring when extended by the values of the **-extension_of_excluded_instances** option, the rings are detoured to leave that instance outside the ring. By default, no instances are excluded.

-extension_of_excluded_instances {*x_extension y_extension*}

Specifies the values by which the boundaries of the instances specified in the **-exclude_instances** option are extended. By default, the instance boundaries are not extended.

-max_deviation *distance*

Specifies maximum deviation of rings location from their "ideal" location (the location as specified by all other options). The deviations can be necessary to avoid DRC violations.

-create_bridges {*offset space*}

Controls the creation of bridges between neighboring rings of the same net. A bridge is a single segment connecting neighboring segments of rings. It is located on the same layer and has the same width as the connected segments. In general, bridges are required if it is impossible to create a single ring because of maximum metal width restrictions defined in the technology file. If the bridges connect vertical segments of rings, the *offset* value specifies the offset of the bottom edge of the first bridge from the bottom edge of a segment of the inner ring. If the bridges connect horizontal segments of rings, the *offset* value specifies an offset of the left edge of the first bridge from the left edge of a segment of the inner ring. The *space* value specifies the distance between bridges. If this value does not correspond to requirements of the technology file, it is adjusted automatically. By default, no bridges are created.

-ignore_parallel_targets

Keeps a ring segment from connecting passing targets if the targets are parallel to the segment. A target refers to an object of the same net located under or over the segment. By default, each ring segment connects all passing targets.

-undo

Removes the rings created by the last **create_rectilinear_rings** run. This option is mutually exclusive with all other options.

DESCRIPTION

This command creates rectilinear rings in the design. You can create rings either around the core area or around macro cell instances. By using the options, you can control the location of the rings and the width and layers of their segments. The main difference between this command and the **create_rectangular_rings** command is that you can specify a list of macro cell instances that are excluded from the ring.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates rectilinear rings for the VSS and VDD nets. Rings are created around a core area with offsets of X = 2.08, Y = 3.56. All segments of the ring are located on the PM layer. All segments have a width of 30.0. The spacing

between rings is selected automatically. The analog_01 macro cell is excluded from the rings. The maximum allowed deviation of the rings location is 12. No vias are dropped from the rings to parallel targets.

```
prompt> create_rectilinear_rings \  
-nets {VSS VDD} \  
-offset {2.08 3.56} \  
-layers {PM PM} \  
-width {30.0 30.0} \  
-exclude_instances analog_01 \  
-extension_of_excluded_instances {2 3.01} \  
-max_deviation 12 \  
-ignore_parallel_targets
```

The following example creates seven rectilinear rings for the VSS net. Rings are created around a core area with offsets of X = 17.08, Y = 18.56. All segments of the ring are located on the M7 layer. All segments have a width of 4.5. The spacing between rings is 1.5. Bridges between rings are created without an offset and spacing between them of 6.0. The analog_01 macro cell is excluded from the rings. The maximum allowed deviation of the ring location is 12. No vias are dropped from the rings to parallel targets.

```
prompt> create_rectilinear_rings \  
-nets {VSS VSS VSS VSS VSS VSS VSS} \  
-offset {17.08 18.56} \  
-layers {M7 M7} \  
-width {4.5 4.5} \  
-space {1.5 1.5} \  
-create_bridges {0 6} \  
-exclude_instances {analog_01} \  
-extension_of_excluded_instances {4.0 4.0} \  
-max_deviation 12 \  
-ignore_parallel_targets
```

SEE ALSO

```
create_rectangular_rings(2)  
create_pad_rings(2)  
report_preroute_drc_strategy(2)  
set_preroute_drc_strategy(2)
```

create_route_guide

Creates a new route guide.

SYNTAX

```
collection create_route_guide
-coordinate rect
[-no_signal_layers layers]
[-no_preroute_layers layers]
[-preferred_direction_only_layers layers]
[-zero_min_spacing]
[-repair_as_single_sbox]
[-horizontal_track_utilization percentage]
[-vertical_track_utilization percentage]
[-switch_preferred_direction]
[-name guide_name]
```

Data Types

<i>rect</i>	list of points
<i>layers</i>	list
<i>percentage</i>	integer
<i>guide_name</i>	string

ARGUMENTS

-coordinate *rect*
Specifies the coordinates of the bounding box of the route guide.
This is a required option.

-no_signal_layers *layers*
Specifies the layers that cannot contain signals. You specify the layers by using the layer names from the technology file.

-no_preroute_layers *layers*
Specifies the layers that cannot contain preroutes. You specify the layers by using the layer names from the technology file.

-preferred_direction_only_layers *layers*
Specifies the layers that must be routed in the preferred direction. You specify the layers by using the layer names from the technology file.

-zero_min_spacing
Specifies that the router can route wires within the keepout margin of the route guide.

-repair_as_single_sbox
Specifies that this route guide should be repaired as a single sbox when there is a difficult violation on a prerouted wire or inside a large macro.

-horizontal_track_utilization *percentage*
Specifies the horizontal track utilization for the route guide.

You must specify an integer value between 0 and 100 for the *percentage* argument. If you do not specify this option or you specify a value greater than 100, it is set to 100.

-vertical_track_utilization *percentage*

Specifies the vertical track utilization for the route guide.

You must specify an integer value between 0 and 100 for the *percentage* argument. If you do not specify this option or you specify a value greater than 100, it is set to 100.

-switch_preferred_direction

Specifies that preferred direction should be switched for the route guide.

-name *guide_name*

Specifies the name of the created route guide.

If you do not specify this option, the tool assigns a name to the route guide using the naming convention RG#x, where x is an integer value.

DESCRIPTION

This command creates a new route guide and returns a collection containing the created route guide.

The tool uses the global snap settings to snap the bounding box of the route guide.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates a route guide.

```
prompt> create_route_guide -coordinate {0.0 0.0 100.0 100.0} \  
-no_signal_layers M1 -no_preroute_layers M2 \  
-preferred_direction_only_layers M4 \  
-horizontal_track_utilization 50 -vertical_track_utilization 30
```

SEE ALSO

```
create_net_shape(2)  
create_user_shape(2)  
create_placement_blockage(2)  
create_via(2)  
create_terminal(2)  
create_text(2)  
get_route_guides(2)  
remove_route_guide(2)
```

create_routing_blockage

Creates a new routing blockage on metal or via routing blockage layers.

SYNTAX

```
collection create_routing_blockage  
-layers layer_list  
-bbox | -boundary {polygon_boundary_points}
```

Data Types

<i>layer_list</i>	list of layers
<i>bounding_box</i>	lower left and upper right coordinates
<i>polygon_boundary_points</i>	list of coordinates

ARGUMENTS

-layers *layer_list*
Specifies the layers in which the routing blockage is located. If you specify more than one layer, the tool creates multiple routing blockages simultaneously.
The valid value are: metal1Blockage-metal15Blockage, via1Blockage-via14Blockage, polyBlockage, and polyContBlockage.
This is a required option.

-bbox {*bounding_box*}
Specifies the the bounding box of a rectangular routing blockage. The format of the bounding box specification is {{*llx lly*} {*urx ury*}}, which specifies the lower-left and upper-right corners of the rectangle.
The unit for the coordinates is the unit specified in the technology file. This option and the **-boundary** option are mutually exclusive. You must specify one of these options.

-boundary {*polygon_boundary_points*}
Specifies the boundary points of a rectilinear routing blockage. The format of the boundary specification is {{*x1 y1*} {*x2 y2*} ... {*xn yn*}}.
The unit for the coordinates is the unit specified in the technology file. This option and the **-bbox** option are mutually exclusive. You must specify one of these options.

DESCRIPTION

The **create_routing_blockage** command creates metal or via routing blockages on the metal, via, or poly blockage layers. Unlike route guides, which direct the router to change routing information on nets that go through a route guide, routing blockages direct the router to avoid routing through these areas.

You can create either rectangular routing blockages by using the **-bbox** option or rectilinear routing blockages by using the **-boundary** option.

When the tool creates routing blockages, it automatically assigns a name of

RB_object_id to each routing blockage.

EXAMPLES

The following example creates a rectangular via routing blockage in the current design.

```
prompt> create_routing_blockage -layers via5Blockage \  
-bbox {{0 0} {100 100}}  
{RB_29440}
```

The following example creates two rectilinear metal routing blockages in the current design.

```
prompt> create_routing_blockage -layers {metal13Blockage via4Blockage} \  
-boundary {{0 0} {50 0} {50 50} {100 50} {100 100} {0 100}}  
{RB_29696 RB_29697}
```

The following example creates a rectangular metal routing blockage in the current design on each metal blockage layer in the design library: prompt>

```
create_routing_blockage \ -layers [get_layers -include_system -filter  
{name=~metal*Blockage}] \ -bbox {{0 0} {50 50}} {RB_29441 RB_29442 RB_29443 RB_29444  
RB_29445 RB_29446 RB_29447 RB_29448 RB_29449 RB_29450 RB_29451 RB_29452 RB_29453  
RB_29454 RB_29455}
```

SEE ALSO

```
get_layers(2)  
get_routing_blockages(2)  
remove_routing_blockage(2)
```


create_rp_group

Creates relative placement groups.

SYNTAX

```
collection create_rp_group
group_list
[-design design_name]
[-columns num_cols]
[-rows num_rows]
[-alignment bottom-left | bottom-pin | bottom-right]
[-pin_align_name pin_name]
[-utilization percentage]
[-ignore]
[-x_offset float]
[-y_offset float]
[-compress]
[-cts_option fixed_placement | size_only]
[-route_opt_option fixed_placement | in_place_size_only]
[-psynopt_option fixed_placement | size_only]
[-move_effort low | medium | high]
[-allow_keepout_over_tapcell false | true]
```

Data Types

<i>group_list</i>	list
<i>design_name</i>	design
<i>num_cols</i>	integer
<i>num_rows</i>	integer
<i>pin_name</i>	string
<i>percentage</i>	float
<i>float</i>	percentage

ARGUMENTS

group_list
Specifies the names for the relative placement (RP) groups you want to create in the specified design. Within a design, each group name must be unique. The tool generates a warning if you try to create a group that does not have a unique name.
The **create_rp_group** command creates one relative placement group for each item in the *group_list*.

-design design_name
Specifies the name of the design in which to create the new groups. If you do not specify a value for this option, the groups are created in the current design.

-columns num_cols
Specifies the number of columns into which objects can subsequently be placed, relative to each other.
If you do not specify a value for this option, the group is created with 1

column.

-rows *num_rows*

Specifies the number of rows into which objects can be subsequently placed, relative to each other.

If you do not specify a value for this option, the group is created with 1 row.

-alignment *bottom-left | bottom-pin | bottom-right*

Specifies the default alignment method to use when placing leaf cells and relative placement groups in the specified relative placement groups. If you do not specify this option, the tool uses bottom-left alignment.

You can override the default alignment method for a specific leaf cell or relative placement group when you add it to a relative placement group (by using the **add_to_rp_group** command).

-pin_align_name *pin_name*

Specifies the default alignment pin for the created relative placement groups. During placement, the tool uses the alignment pin's location to align the cells within a column.

You can override the alignment pin for a specific leaf cell when you add the cell to the group (by using the **add_to_rp_group** command).

-utilization *percentage*

Specifies the utilization percentage to use for placement of this relative placement group. Utilization is represented as a floating-point number between 0.0 (representing 0%) and 1.0 (the default, representing 100%).

-ignore

Indicates that the tool is to ignore this relative placement group during placement and not treat it as a relative placement group. Instead, the tool places all of the group's parts individually, as it would normally do when there is no relative placement.

-origin

-x_offset *float*

Specifies a left coordinate (anchor) for the group, in microns, relative to the lower-left corner in the core area, for top level relative placement group. It is ignored for a sub-level relative placement group. Specifying only an x-offset allows the group to slide in the y-direction.

-y_offset *float*

Specifies a lower coordinate (anchor) for the group, in microns, relative to the lower-left corner in the core area, for top level relative placement group. It is ignored for a sub-level relative placement group. Specifying only a y-offset allows the group to slide in the x-direction.

-compress

Applies compression in the horizontal direction to a relative placement group during placement. Setting this option places each row of a relative placement group without any gaps between leaf cells, lower-level hierarchical relative placement groups, or keepouts. Column alignment is not maintained when you use **-compress**.

If you use **-alignment bottom-right** or **-alignment bottom-pin** and specify **-compress**, the tool generates an error and does not create relative placement

groups.

If both **-utilization** and **-compress** are specified, the utilization constraints are observed with gaps between leaf elements in a relative placement row.

The **-compress** option does not propagate from a parent group to child groups.

-cts_option fixed_placement | size_only

Specifies how to treat the cells in the relative placement group during clock tree synthesis and clock tree optimization.

If you specify fixed_placement, the cells in the relative placement group are treated as fixed during **compile_clock_tree**, **optimize_clock_tree**, and **clock_opt** actions and cannot be sized or moved.

If you specify size_only, the cells in the relative placement group can only be sized. This option is applicable to the **compile_clock_tree**, **optimize_clock_tree**, and **clock_opt** commands.

The default value of the option is fixed_placement.

This option applies only to top-level relative placement groups and is ignored for hierarchical relative placement groups. The top-level value is propagated to the hierarchical relative placement groups.

-route_opt_option fixed_placement | in_place_size_only

Specifies how to treat the cells in the relative placement group during **route_opt** and **psynopt on_route** activity.

If you specify fixed_placement, the cells in the relative placement group are treated as fixed during **route_opt** and **psynopt on_route** activity and cannot be sized or moved.

If you specify in_place_size_only, sizing changes are constrained for minimal engineering change order (ECO) placement changes. For example, a cell is sized to improve timing or design rule costs only if the newly-sized cell can fit into any available space adjacent to the original cell location. The resulting transformation is verified to ensure it is legal.

The default value of the option is in_place_size_only.

This option applies only to top-level relative placement groups and is ignored for hierarchical relative placement groups. The top-level value is propagated to the hierarchical relative placement groups.

-psynopt_option fixed_placement | size_only

Specifies how to treat the cells in the relative placement group during **psynopt** and **place_opt**.

If you specify fixed_placement, the cells in the relative placement group are treated as fixed during **psynopt** and cannot be sized or moved. This value is not recommended for **place_opt**.

If you specify size_only, the cells in the relative placement group can only be sized.

The default value of the option is size_only.

This option applies only to top-level relative placement groups and is ignored for hierarchical relative placement groups. The top-level value is propagated to the hierarchical relative placement groups.

-move_effort low | medium | high

Controls the extent to which a relative placement group can be moved from its initial location to preserve the relative placement without violating relative placement constraints. The size of the region searched for placement of a relative placement group is progressively reduced as you reduce the the effort from high to medium to low.

Coarse placement estimates an initial location for every top-level relative

placement group. If those groups are placed at the same locations returned by coarse placement, the relative placement groups might need to be broken, which would violate the relative placement constraints for those groups. Specifying high effort allows the higher movement of a relative placement group from its initial location in search of a location compared to low and medium effort, to maintain the relative placement constraints and potentially avoid breaking the relative placement group. The tradeoff for maintaining the relative placement group is a potential loss of QoR, because the group is moved from the location initially determined by coarse placement. Specifying low effort indicates that the relative placement group is placed as close as possible to the initial location returned by coarse placement. The tradeoff for maintaining the location is a higher potential for breaking the relative placement groups and violating the relative placement constraints for some relative placement groups. By default, the tool uses medium effort.

`-allow_keepout_over_tapcell false | true`

Specifies that the hard keepout in the relative placement group should be overlapped with tap cells if needed. By default, the value is false.

`-compression`

DESCRIPTION

The **create_rp_group** command creates new relative placement groups. The new groups are created empty and contain no items. Add items to a group using the **add_to_rp_group** command. Remove groups using the **remove_rp_groups** command.

Persistently store and restore relative placement groups with the **write_milkyway** command and the **read_milkyway** command.

This command returns a collection containing the relative placement groups that are created. If no objects are created, the empty string is returned.

Relative placement groups provide for explicit user control of placement for a group of cells. A relative placement group is placed as a whole while maintaining the relative placement (or tiling) of the cells within the group as specified with the **add_to_rp_group** command. Relative placement is also known as "structured placement."

When placing a relative placement group, the placer automatically chooses the orientation for the group. For example, if the placer chooses the north orientation for a relative placement group, the first column of that relative placement group is placed at the left and subsequent columns are placed towards the right. If the placer chooses the flip-north orientation for a relative placement group, the first column of that relative placement group is placed at the right and subsequent columns are placed towards the left.

Relative placement usage is available with Design Compiler Ultra. Relative placement constraints are respected in `create_fp_placement`.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses **create_rp_group** to create a relative placement group:

```
prompt> create_rp_group grp_ripple -design ripple -rows 8  
{ripple::grp_ripple}
```

```
prompt> get_rp_groups grp_ripple  
{ripple::grp_ripple}
```

```
prompt> add_to_rp_group *ripple -leaf carry_in_1 -row 2  
{ripple::grp_ripple}
```

SEE ALSO

```
add_to_rp_group(2)  
all_rp_groups(2)  
create_fp_placement(2)  
remove_rp_groups(2)  
report_rp_group_options(2)  
set_rp_group_options(2)  
write_rp_groups(2)
```

create_scenario

Creates a scenario in memory.

SYNTAX

```
status create_scenario  
scenario_name
```

Data Types

```
scenario_name      string
```

ARGUMENTS

```
scenario_name  
    Specifies the name of the scenario created. A scenario_name must be unique  
    within the current session.
```

DESCRIPTION

Creates a new scenario in memory. The newly created scenario has no scenario-specific constraints.

Multicorner-Multimode Support

This command supports an option to specify which active scenarios it should work with.

EXAMPLES

The following example uses **create_scenario** to create a scenario.

```
prompt> create_scenario MODE1  
prompt> all_scenarios
```

SEE ALSO

```
all_scenarios(2)  
all_active_scenarios(2)  
set_active_scenarios(2)  
current_scenario(2)  
remove_scenario(2)  
create_scenario(2)
```

create_short_drc_error

Creates an error record for an error type in the short error type class.

SYNTAX

```
collection create_short_drc_error
-error_view mw_error_view
-type error_type
-bbox {llx lly urx ury}
[-status error | fixed | ignored | waived]
[-info description]
[-net1 net]
[-net2 net]
[-layer layer]
```

Data Types

<i>mw_error_view</i>	list or collection of one item
<i>error_type</i>	string
<i>llx</i>	float
<i>lly</i>	float
<i>urx</i>	float
<i>ury</i>	float
<i>description</i>	string
<i>net</i>	list or collection of one item
<i>layer</i>	list or collection of one item

ARGUMENTS

-error_view *mw_error_view*
Specifies the error view in which to create the error record. You may specify only a single error view; specifying more than one error view causes an error. This is a required option.

-type *error_type*
Specifies the error type for the new error object. You can specify the error type by using the id or name of the error type. The specified error type must be in the short error type class. This is a required option.

-bbox {*llx lly urx ury*}
Specifies the bounding box of the shorted area. This is a required option.

-status *error* | *fixed* | *ignored* | *waived*
Sets the current status of this error.
If you do not specify this option, the status is set to **error**.

-info *description*
Provides a full description of this error.
If you do not specify this option, the description is left empty.

-net1 net

Specifies the first shorted net. You can specify the net by providing a collection containing the net or by specifying the net name. You cannot use wildcards in the net name.

-net2 net

Specifies the second shorted net. You can specify the net by providing a collection containing the net or by specifying the net name. You cannot use wildcards in the net name.

-layer layer

Specifies the associated layer. You can specify the layer by using the layer name from the technology file or by using the **get_layers** command.

DESCRIPTION

This command creates a new error object of the specified error type, which must be associated with the short error type class. Every error object must be associated with one and only one error type. The error type association is made when the error object is created and cannot be changed. Therefore, you must create an error type before you can create error objects of that type.

The short error type class is for violations where routed shapes overlap in the same area in a layer. At a minimum, an error object associated with the short error type class must define the bounding box of the shorted area. To fully describe the violation, you can also provide a description, the two shorted nets, and the layer where the short occurs.

If successful, the command returns a collection containing the new error object.

EXAMPLES

The following example opens an error view named `mydesign_mydrc.err`, creates an error type named `Short`, and then creates an error record of the `Short` type:

```
prompt> set cellId [open_mw_cel -not_as_current mydesign_mydrc.err]
{mydesign_mydrc}
prompt> set typeId [create_drc_error_type -name Short \
-class short -error_view $cellId
1024
prompt> create_drc_error -type $typeId -error_view $cellId \
-bbox {574.0700 430.3600 578.9700 433.4400} \
-net1 [get_nets myNetA] -net2 [get_nets myNetB] \
-layer [get_layers layerA]
{1280}
```

SEE ALSO

`create_drc_error_type(2)`
`list_drc_error_types(2)`
`report_drc_error_type(2)`
`get_drc_errors(2)`
`remove_drc_error(2)`

create_site_row

Creates a row of sites.

SYNTAX

```
collection create_site_row
-coordinate {X Y}
[-name row_name]
-kind site_type
-space space
-count site_count
[-orient orientation]
[-dir direction]
```

Data Types

<i>row_name</i>	string
<i>site_type</i>	string
<i>space</i>	string
<i>site_count</i>	integer
<i>direction</i>	string

ARGUMENTS

-coordinate {X Y}
Specifies the lower-left corner of the row, regardless of the orientation. The values are specified in microns relative to the chip origin.

-name *row_name*
Specifies the name of the site row, such as ROW1, ROW2, and so forth. If a row name is not specified, the tool names the newly created site row.

-kind *site_type*
Specifies the type of site being defined.

-space *space*
Specifies the space for each site, from the lower-left corner of the site to the lower-left corner of the next site. The value is specified in microns.

-count *site_count*
Specifies the number of sites in the row.

-orient *orientation*
Specifies the orientation of sites. The following values are allowed:

- 0
- 90
- 180
- 270
- 0-mirror
- 90-mirror
- 180-mirror

270-mirror

Alternatively, you can use the following values:

N
W
S
E
FN
FW
FS
FE

The default value is either **0** or **N**.

`-dir direction`

Specifies the direction of the row. The following values are allowed:

h
v
horizontal
vertical
VERTICAL
HORIZONTAL
V
H

The default value is **horizontal**.

DESCRIPTION

The **create_site_row** command creates a row of sites at the specified location.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command creates a horizontal row of 3 sites spaced by 5 units, and originating at {10,10}:

```
prompt> create_site_row -coordinate {10,10} -kind CORE -space 5 -count 3  
{SITE_ROW#12345}
```

SEE ALSO

get_site_rows(2)
ignore_site_row(2)
remove_row_type(2)
remove_site_row(2)
set_row_type(2)

create_spacing_drc_error

Creates an error record for an error type in the spacing error type class.

SYNTAX

```
collection create_spacing_drc_error
-error_view mw_error_view
-type error_type
-bbox {llx lly urx ury}
[-status error | fixed | ignored | waived]
[-info description]
[-direction horizontal | vertical | llToUr | ulToLr]
[-net1 net]
[-net2 net]
[-layer layer]
```

Data Types

<i>mw_error_view</i>	list or collection of one item
<i>error_type</i>	string
<i>llx</i>	float
<i>lly</i>	float
<i>urx</i>	float
<i>ury</i>	float
<i>description</i>	string
<i>net</i>	list or collection of one item
<i>layer</i>	list or collection of one item

ARGUMENTS

-error_view *mw_error_view*
Specifies the error view in which to create the error record. You may specify only a single error view; specifying more than one error view causes an error. This is a required option.

-type *error_type*
Specifies the error type for the new error object. You can specify the error type by using the id or name of the error type. The specified error type must be in the spacing error type class. This is a required option.

-bbox {*llx lly urx ury*}
Specifies the bounding box that describes the spacing error. This is a required option.

-status *error* | *fixed* | *ignored* | *waived*
Sets the current status of this error.
If you do not specify this option, the status is set to **error**.

-info *description*
Provides a full description of this error.
If you do not specify this option, the description is left empty.

`-direction horizontal | vertical | llToUr | ulToLr`

Specifies the direction of the spacing violation. The direction determines the way in which the violation is rendered graphically. The following list describes the rendering behavior for each direction keyword:

- **horizontal**

Draws parallel vertical lines with a double-ended horizontal arrow between them.

- **vertical**

Draws parallel horizontal lines with a double-ended vertical arrow between them.

- **llToUr**

Draws a double-ended arrow from the lower-left corner to the upper-right corner.

- **ulToLr**

Draws a double-ended arrow from the upper-left corner to the lower-right corner.

If you do not specify this option, the tool draws a bounding box for the spacing violation.

`-net1 net`

Specifies the first net involved in the spacing violation. You can specify the net by providing a collection containing the net or by specifying the net name. You cannot use wildcards in the net name.

`-net2 net`

Specifies the second net involved in the spacing violation. You can specify the net by providing a collection containing the net or by specifying the net name. You cannot use wildcards in the net name.

`-layer layer`

Specifies the associated layer. You can specify the layer by using the layer name from the technology file or by using the **get_layers** command.

DESCRIPTION

This command creates a new error object of the specified error type, which must be associated with the spacing error type class. Every error object must be associated with one and only one error type. The error type association is made when the error object is created and cannot be changed. Therefore, you must create an error type before you can create error objects of that type.

The spacing error type class is for violations where two routed shapes in the same layer violate a spacing rule. At a minimum, an error object associated with the spacing error type class must include the bounding box that describes the distance that violated the spacing rule. To fully describe the violation, you can also provide a description, the two nets with the spacing violation, and the layer where the violation occurs. You can also specify the rendering direction.

If successful, the command returns a collection containing the new error object.

EXAMPLES

The following example opens an error view named `mydesign_mydrc.err`, creates an error type named "Via Spacing", and then creates an error record of the "Via Spacing" type:

```
prompt> set cellId [open_mw_cel -not_as_current mydesign_mydrc.err]
{mydesign_mydrc}
prompt> set typeId [create_drc_error_type -name \
    "Via Spacing" -class Spacing \
    -info "minimum spacing = 0.22 um" -error_view $cellId]
1024
prompt> create_spacing_drc_error -type $typeId -error_view $cellId \
    -bbox {574.0700 430.3600 578.9700 433.4400} \
    -direction horizontal
{1280}
```

SEE ALSO

```
create_drc_error_type(2)
list_drc_error_types(2)
report_drc_error_type(2)
get_drc_errors(2)
remove_drc_error(2)
```

create_stack_via_on_pad_pin

Creates stacked vias from the pad layer to the top layer for flip-chip drivers.

SYNTAX

```
status create_stack_via_on_pad_pin
-from_metal layer
-to_metal layer
[-remove_existing_stack_via true | false]
[-route_type user_enter | signal_route]
[-nets collection_of_nets
  | -nets_in_file file_name]
[-all_driver_pins true | false]
[-terminal_names list_of_terminal_names]
```

Data Types

<i>layer</i>	collection of one layer
<i>collection_of_nets</i>	collection
<i>file_name</i>	string
<i>list_of_terminal_names</i>	list

DATA TYPES

<i>layer</i>	collection of one layer
<i>collection_of_nets</i>	collection
<i>file_name</i>	string
<i>list_of_terminal_names</i>	list

ARGUMENTS

-from_metal layer
Specifies the pad layer. You can specify the layer as mx or Mx, where x is the layer number or you can use the layer number from the technology file. This is a required option.

-to_metal layer
Specifies the top layer. You can specify the layer as mx or Mx, where x is the layer number or you can use the layer number from the technology file. This is a required option.

-remove_existing_stack_via true | false
Controls whether the tool removes existing stacked vias between the specified layers.
By default, this option is false and existing stacked vias are not removed.

-route_type user_enter | signal_route
Specifies the route type for the created stacked via. Valid values are **user_enter** or **signal_route**.
If you do not specify this option, the stacked via is created with a route

type of **signal_route**.

-nets_collection_of_nets

Specifies the flip-chip nets for which to create stacked vias.

This option and the **-nets_in_file** option are mutually exclusive; you can specify only one. If you do not specify either option, the tool uses the nets specified in the **set_route_flip_chip_options** command.

-nets_in_file file_name

Specifies the name of the file that contains the list of flip-chip nets for which to create stacked vias.

This option and the **-nets_in_file** option are mutually exclusive; you can specify only one. If you do not specify either option, the tool uses the nets specified in the **set_route_flip_chip_options** command.

-all_driver_pins true | false

Controls whether stacked vias are created on all pins of driver cells.

By default (false), the flip-chip router creates a stacked via on one pin of the driver cell.

-terminal_names list_of_terminal_names

Use this option for nets with multiple driver pins, also known as multi-terminal nets, or for electrical equivalent (EEQ) pins. Sometimes a flip chip net can be connected to several driver pins or EEQ pins. The **-terminal_names** option allows you to selectively create stacked vias on the specified driver pins or EEQ pins. The syntax for the terminal names is *instance_name/pin_name* for multi-terminal nets and *instance_name/port_name:pin_name* for EEQ pins.

DESCRIPTION

This command creates stacked vias on flip-chip drivers from the pad pin to the top metal layer. If stacked vias already exist on the flip-chip drivers, the tool does not remove the existing stacked vias unless you set the **-remove_existing_stack_via** option to true.

This command behaves as follows:

- Follows the var-rule width and spacing settings to create square vias or via arrays.
- If the vias cross the pad pin bounding box or design boundary, cut it with pad pin bounding box or design boundary. In this case, the vias might become rectangular vias.
- If there are still DRC violations due to stacked vias crossing the design boundary, the tool issues warning messages and creates the stacked vias.

EXAMPLES

The following example creates stacked vias with a width of 10 microns and spacing of 2 microns.

```
prompt> set_route_flip_chip_options \
      -rule_name rdlRule -target_net_file bumpnet \
      -layer_spacing {m8 2, m7 2, m6 2, m5 2, m4 2} \
      -layer_width {m8 10, m7 10, m6 10, m5 10, m4 10}
prompt> create_stack_via_on_pad_pin -from_metal m4 -to_metal M8
```

The following command uses the layer number from the technology file to identify the pad layer and the top layer.

```
prompt> create_stack_via_on_pad_pin -from_metal 65 -to_metal 57
```

The following command specifies the terminal_names to create stacked vias on EEQ pins "VSS", "VSS 1", and "VSS 5" of instance I_1.

```
prompt> create_stack_via_on_pad_pin -from_metal m5 -to_metal m9 -nets VSS
      -terminal_names {"I_1/VSS:VSS" "I_1/VSS:VSS 1" "I_1/VSS:VSS 5"}
```

SEE ALSO

```
set_route_flip_chip_options(2)
route_flip_chip(2)
write_flip_chip_nets(2)
push_flip_chip_route(2)
display_flip_chip_route_flylines(2)
optimize_flip_chip_route(2)
```


create_supply_net

Creates a supply net for the specified power domain. The supply net is created in the logic hierarchy at the same scope as the specified power domain. This command is supported only in UPF mode.

SYNTAX

```
string create_supply_net  
supply_net_name  
-domain domain_name  
[-reuse]  
[-resolve unresolved | parallel]
```

Data Types

<i>supply_net_name</i>	string
<i>domain_name</i>	string

ARGUMENTS

supply_net_name
Specifies the name of the supply net to be created. The name should be a simple (non-hierarchical) name.
In the scope of specified power domain, if there is a supply net, logical hierarchical net, or logical port with the same name, the supply net cannot be created. An exception to this rule is when you use the **-reuse** option. In this case the net name must be same as an existing supply net in the same scope; name; check is not performed.
This argument is required.

-domain *domain_name*
Specifies the power domain for which this supply net is defined. The power domain must exist.
This argument is required.

-reuse
Reuses an existing supply net instead of creating a new one. One supply net can be associated with several power domains. If this option is used, the specified supply net must already exist.

-resolve unresolved | parallel
Specifies how the state and voltage of the supply net are resolved when the supply net is driven by a power switch or multiple power switches. Specifying *unresolved* means that this supply net allows only a single driver. Specifying *parallel* means that this supply net can be driven by multiple power switches. When the supply net is connected to the output of multiple power switches, any number of outputs may be ON at the same time, as long as the voltage value is same.
The default is **unresolved**.
You cannot use both **-resolve** and **-reuse** options in the same **create_supply_net** command.

DESCRIPTION

The **create_supply_net** command enables you to create a supply net defined in the specified power domain. A supply net presents the intention of the power supply in the power domains. A supply net connects supply ports and, or pins.

Each power domain has a primary power supply net and a primary ground supply net, and could have several other supply nets. If the command succeeds, a supply net is created in the scope of the power domain. The new supply net is neither a primary power net nor a primary ground net of the power domain. Use the **set_domain_supply_net** command to set the supply net as the primary power or ground net.

On success this command returns the full name of the supply net (from the current scope). On failure, it returns a null string.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates a supply net named A_VDD in the power domain PD1, and recreates supply net A_DD in the power domain PD2 using the **-reuse** option:

```
prompt> create_power_domain PD1 -element INST1
PD1

prompt> create_power_domain PD2 -element INST2
PD2

prompt> create_supply_net A_VDD -domain PD1
A_VDD

prompt> create_supply_net A_VDD -domain PD2 -reuse
A_VDD
```

SEE ALSO

report_supply_net(2)

create_supply_port

Creates a supply port in the specified power domain or in the current scope if no power domain is specified. This command is supported only in UPF mode.

SYNTAX

```
string create_supply_port  
supply_port_name  
[-domain domain_name]  
[-direction in | out]
```

Data Types

<i>supply_port_name</i>	string
<i>domain_name</i>	string

ARGUMENTS

supply_port_name
Specifies the name of the supply port to be created. The name should be either a simple (non-hierarchical) name if **-domain** is specified, or a full (hierarchical) name if **-domain** is not specified.
In the scope of the specified power domain, if there is a supply port, logical port, or logical hierarchical net with the same name, the supply port cannot be created.
This argument is required.

-domain *domain_name*
Specifies the power domain where this port defines a supply net connection point. If the specified power domain does not exist in the current scope, the command fails.

-direction in | out
Defines how state information is propagated through the supply network as it is connected to the port. If the port is an input port, the state information of the external supply net connected to the port shall be propagated into the domain. Likewise, for an output port, the state information of the internal supply net connected to the port shall be propagated outside of the domain. The default value for this option is in.

DESCRIPTION

The *create_supply_port* command enables you to create a supply port at the scope of the specified power_domain or at the current scope. A supply port provides a connection point for the supply net.

This command returns the full name of the supply port (from the current scope) upon success. The command returns a null string if it fails.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates the supply port VN1 at the scope of power domain PD1:

```
prompt> create_power_domain PD1 -element INST1
PD1
```

```
prompt> create_supply_port VN1 -domain PD1
VN1
```

SEE ALSO

```
remove_supply_port(2)
report_supply_port(2)
```

create_terminal

Creates a new terminal for a logical port.

SYNTAX

```
collection create_terminal  
-bbox rect | -boundary boundary  
-layer layer  
-port string  
[-direction {pin_directions}]  
[-name string]
```

Data Types

<i>rect</i>	list of two points
<i>boundary</i>	list of n points
<i>layer</i>	collection

ARGUMENTS

- bbox *rect***
Specifies the bounding box of the terminal.
This option and the **-boundary** option are mutually exclusive; you must specify one.
- boundary *boundary***
Specifies the rectilinear boundary of the terminal.
This option and the **-bbox** option are mutually exclusive; you must specify one.
- layer *layer***
Specifies the layer of the terminal.
This is a required option.
- port *string***
Specifies the name of the associated logical port of the terminal.
This is a required option.
- direction {*pin_directions*}**
Specifies a list of allowable access directions for the terminal. You can specify one or more of the following values: **left**, **right**, **up**, or **down**.
If you do not specify this option, no access direction is assumed.
- name *string***
Specifies the name of the created terminal.
If you do not specify this option, the tool automatically generates a terminal name from the port name.
Note: In most cases, you should not specify this option and allow the tool to automatically generate the terminal name, because some operations require that the terminal name conforms to the terminal naming conventions. For information about the terminal naming conventions, see the DESCRIPTION section.
If the specified terminal name does not match terminal naming conventions, a

warning is issued.

DESCRIPTION

This command creates a new terminal for a logical port and returns a collection containing the created terminal.

Terminals have a layer, a set of access directions, and a boundary that can be either a simple rectangle or a rectilinear shape.

By convention, the terminal name is the same as the name of the associated port; however, this is not a strict requirement. For electrically equivalent (EEQ) pins, where there are multiple terminals associated with the same port, the first terminal has the same name as the port and subsequent terminal names have the format '`<port_name> <number>`' where '`<port_name>`' is the port name and '`<number>`' is a unique number which, when automatically generated, starts from one and increments for each new terminal on that port.

Notes

Snapping of the bounding box or boundary is done automatically using the global snap settings.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates a terminal for port A1 on the M3 layer:

```
prompt> create_terminal -bbox {0 0 10 10} -port A1 -layer M3
```

SEE ALSO

```
create_net_shape(2)
create_user_shape(2)
create_placement_blockage(2)
create_route_guide(2)
create_via(2)
create_text(2)
```

create_text

Creates a new text.

SYNTAX

```
object create_text
-origin {x_y}
[-height height]
[-layer layer_name]
[-orient orient_type]
[-anchor text_anchor]
text
```

Data Types

<i>height</i>	real
<i>layer_name</i>	layer
<i>orient_type</i>	string
<i>text_anchor</i>	string
<i>text</i>	string

ARGUMENTS

-origin {x_y}
Specifies text origin.
Replace x and y with the coordinates of the point of origin.

-height *height*
Specifies text height.
By default, the *height* value is taken from the **mw_text_height** global variable.

-layer *layer_name*
Specifies the text layer.
By default, the *layer_name* value is taken from the **mw_text_layer** global variable.

-orient *orient_type*
Specifies the orientation of the text using either Design Exchange Format (DEF) or Floorplan Compiler notation.
The valid values are as follows:
N, W, S, E, FN, FS, FE, FW
NW, NE, EN, ES, SE, SW, WN, WS
0, 90, 180, 270, 0-mirror, 90-mirror,
180-mirror, 270-mirror
The following groups of values are synonymous:
N, NE, 0
W, WN, 90
S, SW, 180
E, ES, 270

FN, NW, 0-mirror

FS, SE, 180-mirror
FE, EN, 270-mirror
FW, WS, 90-mirror
The default is **N**.

`-anchor text_anchor`
Specifies the anchor position for text from the origin.
The valid values and their descriptions are as follows:
lb - Left Bottom
cb - Center Bottom
rb - Right Bottom
lc - Left Center
c - Center Center
rc - Right Center
lt - Left Top
ct - Center Top
rt - Right Top
The default is **lb**.

`text`
Specifies the text string to create and display.

RETURNS

`object`
The created object

DESCRIPTION

This command creates a new text.

Snapping of the origin is automatically done by using global snap settings.

Note: The object returned by the command is the object created.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates a text string at origin coordinates {10 10}.

```
prompt> create_text -origin {10 10} -text Hello
```

SEE ALSO

`create_net_shape(2)`
`create_placement_blockage(2)`
`create_route_guide(2)`
`create_terminal(2)`


```
create_user_shape(2)
create_via(2)
```

create_track

Creates the tracks for a routing layer or a poly layer.

SYNTAX

```
int create_track  
-layer layer  
[-space track_pitch]  
[-count number_of_tracks]  
[-coord start_x_or_y]  
[-dir X | Y]  
[-bounding_box track_boundary_box]
```

Data Types

<i>layer</i>	string
<i>track_pitch</i>	float
<i>number_of_tracks</i>	integer
<i>start_x_or_y</i>	float
<i>track_boundary_box</i>	rectangle

ARGUMENTS

- layer *layer***
Specifies the layer to use the tracks. You can use layer name, layer number, or a collection containing one layer object.
- space *track_pitch***
Specifies the pitch distance between each track. The pitch is the center-to-center distance between two routing wires (the routing wire route on top of each track). By default, the distance is the same as the routing pitch from the physical library. The pitch distance's unit is specified in technology file (usually it is micron).
- count *number_of_tracks***
Specifies how many tracks to create. By default, the tracks use the entire die area. If not specify this option, the value will be automatically calculated and decided by die area and space.
- coord *start_x_or_y***
Specifies the location of the first track. The X and Y coordinates depend on the values specified with the **-dir** option. By default, the track is located at half of the space distance inside of the die area. The coordinates, Å unit is specified in technology file (usually it is micron).
- dir X | Y**
Specifies the direction how tracks are placed. The valid values are **X** and **Y**; specify either. By default, the direction is the routing direction of the layer, which is specified in the physical library.
- bounding_box *track_boundary_box***
Specifies the lower left and upper right coordinates of the boundary box which

includes the entire tracks. By default, the boundary box just fits the entire tracks. The coordinates, Å unit is specified in technology file (usually it is micron).

DESCRIPTION

This command creates a group of tracks on the floorplan so the router can use them to perform detail routing or command **insert_metal_filler** can use them to fill the poly layer at chip finishing step. You must specify either a poly layer or a routing layer for the tracks. The option of creating the track on poly layer is not intended to support routing on poly layer. The tracks can be saved in the Design Exchange Format (DEF) file or the Synopsys database format (Milkyway) file.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates routing tracks for a routing layer named *m3* on the floorplan and reports the newly created routing tracks.

```
prompt> create_track -layer m3
Warning: Space is not specified. Using pitch. (MWUI-124)
Warning: Direction is not specified. Using the layer preferred direction.
(MWUI-125)
Warning: Coordinate value is not specified. Defaulting to the left(bottom)
coordinate of the track's bounding box. (MWUI-126)
Warning: Count value is not specified. Covering the bounding box of track,
depending on its space value. (MWUI-127)
1
prompt> report_track -layer m3
*****
Report track
Design : design
Version: Y-2006.06-ICC-SP2
Date   : Thu Jul  6 00:45:35 2006
*****
Layer          Direction      Start          Tracks        Pitch          Attr
-----
Attributes :
    usr : User defined
    rt  : Route66 defined
    def : DEF defined

m3              Y              0.280          3696          0.560          usr
1
```

SEE ALSO

remove_track(2)
report_track(2)

create_user_shape

Creates a new user shape. A user shape is a metal shape that is not associated with a net.

SYNTAX

```
collection create_user_shape
[-type wire | path | trap | rect | poly]
-origin point
    | -points list_of_points
    | -bbox rect
    | -boundary boundary
[-length real]
[-width real]
[-path_type square | round | extend_half_width | octagon]
-layer layer
[-vertical]
[-route_type route_type]
[-datatype int]
[-avoid_short_segment]
```

Data Types

<i>point</i>	point
<i>list_of_points</i>	list of points
<i>rect</i>	rectangle
<i>boundary</i>	polygon
<i>real</i>	real
<i>layer</i>	collection
<i>route_type</i>	string
<i>int</i>	integer

ARGUMENTS

-type wire | path | trap | rect | poly

Specifies the type of the user shape. Valid values are wire, path, trap (trapezoid), rect (rectangle), and poly (polygon).

The default is **poly**.

-origin *point*

Specifies the origin of the user shape for wires.

When you specify this option, you must also specify the **-length** and **-width** options.

The **-origin**, **-points**, **-bbox**, and **-boundary** options are mutually exclusive. You must specify one of these options.

-points *list_of_points*

Specifies the points of the user shape for paths.

When you specify this option, you must also specify the **-width** option.

The **-origin**, **-points**, **-bbox**, and **-boundary** options are mutually exclusive. You must specify one of these options.

-bbox *rect*

Specifies the bounding box of the user shape. You must specify the lower-left and upper-right corners of the bounding box by using the following syntax: {{llx lly} {urx ury}}.

The **-origin**, **-points**, **-bbox**, and **-boundary** options are mutually exclusive. You must specify one of these options.

-boundary *boundary*

Specifies the boundary of the user shape. You must specify at least three nonoverlapping points.

The **-origin**, **-points**, **-bbox**, and **-boundary** options are mutually exclusive. You must specify one of these options.

-length *real*

Specifies the length of the user shape for wires in user units.

-width *real*

Specifies the width of the user shape for wires in user units.

This option is required when you specify the **-origin** or **-points** option and ignored otherwise.

-path_type *square | round | extend_half_width | octagon*

Specifies the alignment type of a wire or path end.

Possible values are

square	- square, no extension
round	- round, half width extension
extend_half_width	- square, half width extension
octagon	- octagon, half width extension

The default is **square**.

-layer *layer*

Specifies the layer for the user shape. You can specify the layer by using the layer name from the technology file or by using the **get_layers** command. This is a required option.

-vertical

Indicates that the wire is vertical.

By default, the user shape is horizontal.

-route_type *route_type*

Specifies the route type of the user shape.

Use one of the following route types:

user_enter	User entered
signal_route, signal_route_detail	Detail routing
signal_route_global	Global routing
pg_ring	Power or ground (PG) ring
pg_strap	PG strap
pg_macro_io_pin_conn	PG net that connects to the PG pin of an I/O pad cell or a macro cell
pg_std_cell_pin_conn	PG net that connects to the PG pin of a standard cell
clk_ring	Clock ring
clk_strap	Clock strap
clk_zero_skew_route	Clock zero skew route
bus	Bus

shield, shield_fixed	Fixed shield
shield_dynamic	Dynamic shield
fill_track, clk_fill_track	Fill track

By default, the route type is **signal_route**.

-datatype *int*

Specifies the data type number.

By default, the data type is 0.

-avoid_short_segment

Specifies that segments shorter than a half width should be avoided for paths.

DESCRIPTION

This command creates a shape object that is not attached to a net and returns a collection containing the created object.

The valid shape types are

Wire (horizontal or vertical)

Path

Trapezoid

Rectangle

Polygon

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates a wire user shape.

```
prompt> create_user_shape -type wire \
-origin {0 0} -length 10 -width 2 -layer M1
```

SEE ALSO

create_net_shape(2)

create_placement_blockage(2)

create_route_guide(2)

create_via(2)

create_terminal(2)

create_text(2)

create_via

Creates a new via.

SYNTAX

```
object create_via
-at point
[-name string]
-master string | -auto
[-net string | -no_net]
[-
route_type route_type | signal_route | signal_route_global | signal_route_detail |
pg_ring | pg_strap | pg_macro_io_pin_conn | pg_std_cell_pin_conn | clk_ring | clk_s
trap | clk_zero_skew_route | bus | shield | shield_dynamic | clk_fill_track]
[-
orient orient | FN | FS | FE | FW | NW | NE | EN | ES | SE | SW | WN | WS | 0 | 90
| 180 | 270 | 0-mirror | 90-mirror | 180-mirror | 270-mirror]
[-type via | via_array | via_cell]
[-row int]
[-col int]
[-x_pitch real]
[-y_pitch real]
[-allow_multiple]
```

ARGUMENTS

-at *point*

Specifies the center of the created via or via array.
This is a required option.

-name *string*

Specifies the optional name of the created via or via array.

-master *string*

Specifies the master name of the via.
This option and the **-auto** option are mutually exclusive; you must specify one of these options.

-auto

Automatically creates a via based on the nearest wire intersection at the specified creation coordinate.
This option automatically sets the **-col**, **-row**, **-x_pitch**, **-y_pitch**, **-master**, and **-net** options.
If you specify values for **-col**, **-row**, **-x_pitch**, or **-y_pitch**-col, the specified values override the automatic values.
This option and the **-master** option are mutually exclusive; you must specify one of these options.

-net *string*

Specifies the name of the net to connect to the via.
If you use the **-auto** option, you cannot specify this option. If you do not use the **-auto** option, you must specify either this option or the **-no_net**

option.

-no_net

Specifies that no net should be connected to the via.

If you use the **-auto** option, you cannot specify this option. If you do not use the **-auto** option, you must specify either this option or the **-net** option.

-route_type *route_type* | *signal_route* | *signal_route_global* | *signal_route_detail* | *pg_ring* | *pg_strap* | *pg_macro_io_pin_conn* | *pg_std_cell_pin_conn* | *clk_ring* | *clk_strap* | *clk_zero_skew_route* | *bus* | *shield* | *shield_dynamic* | *clk_fill_track*

The route type of the via. One of:

<i>user_enter</i>	User entered
<i>signal_route</i> , <i>signal_route_detail</i>	Detail routing
<i>signal_route_global</i>	Global routing
<i>pg_ring</i>	Power or ground (PG) ring
<i>pg_strap</i>	PG strap
<i>pg_macro_io_pin_conn</i>	PG net that connects to the PG pin of an I/O pad cell or a macro cell
<i>pg_std_cell_pin_conn</i>	PG net that connects to the PG pin of a standard cell
<i>clk_ring</i>	Clock ring
<i>clk_strap</i>	Clock strap
<i>clk_zero_skew_route</i>	Clock zero skew route
<i>bus</i>	Bus
<i>shield</i> , <i>shield_fixed</i>	Fixed shield
<i>shield_dynamic</i>	Dynamic shield
<i>fill_track</i> , <i>clk_fill_track</i>	Fill track

By default, the route type is **signal_route**.

-orient *orient* | FN | FS | FE | FW | NW | NE | EN | ES | SE | SW | WN | WS | 0 | 90 | 180 | 270 | 0-mirror | 90-mirror | 180-mirror | 270-mirror

Specifies the orientation of the via or via array using either DEF or IC Compiler notation.

The following values are allowed:

N, W, S, E, FN, FS, FE, FW

NW, NE, EN, ES, SE, SW, WN, WS

0, 90, 180, 270, 0-mirror, 90-mirror, 180-mirror, 270-mirror

The following values are synonymous:

N , NE, 0
W , WN, 90
S , SW, 180
E , ES, 270
FN, NW, 0-mirror
FS, SE, 180-mirror
FE, EN, 270-mirror
FW, WS, 90-mirror

By default, the orientation is **N**.

-type *via* | *via_array* | *via_cell*

Specifies which type of via to create. One of:

create_via


```
via          Via
via_array Via array
via_cell    Via cell
```

By default, a via is created, unless you specify another command option that is specific to via arrays.

```
-row int
    Specifies the number of via array rows.

-col int
    Specifies the number of via array columns.

-x_pitch real
    Specifies the x-distance between cuts in the via array.

-y_pitch real
    Specifies the y-distance between cuts in the via array.

-allow_multiple
    Allows multiple vias to be created when you use the -auto option.
```

DESCRIPTION

This command creates a new via, via array, or via cell and returns the created object.

Snapping is done automatically by using the global snap settings.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates a via.

```
prompt> create_via -at {100 100} -route_type clk_ring\
-master VIA12 -no_net
```

SEE ALSO

```
create_net_shape(2)
create_user_shape(2)
create_placement_blockage(2)
create_route_guide(2)
create_terminal(2)
create_text(2)
```

create_voltage_area

Creates a voltage area at the specified region for providing placement constraints of cells associated with the region.

SYNTAX

```
int create_voltage_area
modules -name voltage_area_name
    | -power_domain power_domain_name
-coordinate llx1_lly1_urx1_ury1_...
-polygons {{{x1_y1}_..._{xN_yN}_{x1_y1}}_...}
[-guard_band_x guard_band_width]
[-guard_band_y guard_band_width]
[-is_fixed]
[-target_utilization utilization]
[-color string]
[-cycle_color]
```

Data Types

<i>modules</i>	list
<i>voltage_area_name</i>	string
<i>power_domain_name</i>	string
<i>guard_band_width</i>	int
<i>utilization</i>	float
<i>string</i>	power_domain_name

ARGUMENTS

modules

Specifies a list of hierarchical cells that are contained in the voltage area. Multiple hierarchies are permitted, and one cell cannot be in two voltage areas. Leaf cells (or standard cells) cannot be specified in this option. You must specify this option when you specify the **-name** option. You cannot specify this option when you specify the **-power_domain** option.

-name *voltage_area_name*

Specifies the name of the voltage area to be created. If there is a voltage with the same name, the voltage cannot be created. The name *DEFAULT_VA* is reserved and cannot be used for user-defined voltage area. When you specify this option, you must also specify the *modules* argument. This option and the **-power_domain** option are mutually exclusive; specify only one.

-power_domain *power_domain_name*

Creates a voltage area from an existing power domain. The name of the voltage area is identical to the name of the power domain. All hierarchical cells associated with the power domain are included in the voltage area. After the voltage area is successfully created, the power domain contains the corresponding boundary information. You cannot create a voltage area from a top-level power domain.

This option and the **-name** option are mutually exclusive. You must specify

create_voltage_area

either this option or the **-name** option along with the *modules* argument.

-coordinate *llx1_lly1_urx1_ury1_...*

Specifies a list of lower-left and upper-right coordinates of a voltage area. Each set of {llx lly urx ury} defines a target rectangular placement area. The voltage area geometry can be a rectangle or a rectilinear polygon. The unit of value is micron.

If this option is not specified, the voltage area will be created outside the chip. This option is used for automatic voltage area shaping.

-polygons {{{x1_y1}_..._{xN_yN}_{x1_y1}}_...}

Specifies a list of polygons of a voltage area. Each set of {x1 y1 ... xN yN x1 y1} defines the points list of one polygon. The voltage area geometry can be a rectangle (5 points) or a rectilinear polygon. The unit of value is micron.

If this option is not specified, the voltage area will be created outside the chip. This option is used for automatic voltage area shaping.

-guard_band_x *guard_band_width*

Specifies the guard width in the horizontal direction. The guardband is the spacing along the boundary of the voltage area where cells cannot be placed because of the lack of power supply rails.

The value specified with this option must be a positive integer.

-guard_band_y *guard_band_width*

Specifies the guard width in the vertical direction. The guardband is the spacing along the boundary of the voltage area where cells cannot be placed because of the lack of power supply rails.

The value specified with this option must be a positive integer.

-is_fixed

Specifies that the voltage area is in a fixed location, and the shaping will ignore it. The default value of this option is false.

-target_utilization *utilization*

Specifies the utilization required for this voltage area during shaping. The default is the design utilization. The value specified must be between 0.1 and 1.0.

-color *string*

Specifies the color for the voltage area and its leaf cells. This is used in the GUI. The value can be either a color string such as red, blue, and so on, or it can be a color index which is a number between 0 to 63. Each of these numbers represents a different color for displaying in the GUI, the voltage area and its leaf cells.

This option and the **-cycle_color** option are mutually exclusive; if you specify both, no color is assigned to the voltage area.

-cycle_color

Automatically chooses the next color in the color table, and assigns the color to the voltage area and its leaf cells.

This option and the **-color** option are mutually exclusive; if you specify both, no color is assigned.

DESCRIPTION

The **create_voltage_area** command enables you to create a voltage area of specific geometry on the core area of the chip. The voltage area is associated with hierarchical cells. The placer assumes the voltage area to be an exclusive, hard move bound and tries to place all the cells associated with the voltage area within the geometrical area of the voltage area, as well as place all the cells not associated with the voltage area outside.

Voltage areas can physically be completely nested: one voltage area lies completely inside another voltage area. When considering the area or utilization of outside voltage area, the area of inner voltage area is excluded.

Voltage areas can also be logically nested: one hierarchical cell belongs to one voltage area, while one of its descents belongs to another voltage area. When considering the utilization, all descents belonging to other voltage areas will be excluded.

Voltage areas cannot be partially overlapped. The creation will fail if you try to create a voltage area partially overlapping with an existing voltage area.

Move bounds can only be completely nested inside voltage areas. Voltage areas cannot be partially overlapping with move bounds, nor it can be completely nested inside move bounds.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example constrains the instance INST_1 to lie within the voltage area whose coordinates are lower-left corner (100 100) and upper-right corner (200 200):

```
prompt> create_voltage_area -name foo \  
        -coordinate {100 100 200 200} INST_1
```

The following example creates a voltage area from an existing power domain:

```
prompt> create_voltage_area -power_domain INST \  
        -coordinate {215 215 350 350}
```

The following example creates a voltage area by -polygons option:

```
prompt> create_voltage_area -power_domain INST \  
        -  
        polygons {{100 100} {300 100} {300 200} {200 200} {200 300} {100 300} {100 100}}
```

The following example creates a voltage area with disjoint polygons by -polygons option:

```
prompt> create_voltage_area -power_domain INST \  
        -
```

```
polygons {{{100 100} {300 100} {300 200} {200 200} {200 300} {100 300} {100 100}} \
          {{400 400} {500 400} {500 500} {400 500} {400 400}}}
```

SEE ALSO

```
remove_voltage_area(2)
report_voltage_area(2)
update_voltage_area(2)
create_bounds(2)
```

create_zrt_shield

Performs automatic shield routing.

SYNTAX

```
status create_zrt_shield
[-mode new | unshield | reshield]
[-nets collection_of_nets]
[-with_ground net_name]
[-ignore_shielding_net_pins true | false]
[-ignore_shielding_net_rails true | false]
[-coaxial_below true | false]
[-coaxial_above true | false]
[-coaxial_below_skip_tracks number_of_tracks]
[-coaxial_above_skip_tracks number_of_tracks]
[-pg_via_tie_effort_level low | medium | high]
```

Data Types

<i>collection_of_nets</i>	collection
<i>net_name</i>	string
<i>number_of_tracks</i>	int

ARGUMENTS

-mode new | unshield | reshield
Controls which action to perform during automatic shielding.
The definition for each mode is

- * **new** (default)
Creates new shielding for unshielded nets. If a net is previously shielded, even partially, no new shielding is created for that net.
- * **unshield**
Removes existing shielding wires that are associated with shielded nets. All tie-off connections from deleted shielding to the power and ground network are also deleted to prevent dangling power or ground (PG) nets. The remaining shielding is reconnected to the power and ground network, if needed.

This command can remove only those shielding wires that were created by using the **create_zrt_shield** command. Shielding wires created by other means, such as a third-party tool, might not have full association information and must be removed by using the **remove_route_by_type -shield** command.

If the tool can't find any shielding wires associated with shielded nets, no action is performed.

In unshield mode, only the **-nets** option is considered; all other options are ignored.

* **reshield**

Creates new shielding after unshielding the nets. The shielding is first removed by using the process described for the **unshield** mode. After moving any existing shielding, new shielding is added for the specified nets that do not have shielding.

If the design does not contain existing shielding, it is more efficient to use new mode, rather than resshield mode, because in resshield mode the tool first checks for existing shielding and then creates new shielding.

-nets *collection_of_nets*

Specifies the nets to be shielded, unshielded, or resshielded, depending on the specified mode.

When creating shielding, if the specified nets do not have shielding rules defined, the tool uses the default width and minimum spacing for the layer as defined in the technology file.

By default, this command works on all nets with defined shielding rules. To define the shielding rules, use the **define_routing_rule** and **set_net_routing_rule** commands.

-with_ground *net_name*

Specifies which power or ground net to tie the shielding wires to.

By default, the shielding wires are tied to the ground net. If the design contains multiple ground nets, you must use this option to specify the ground net.

-ignore_shielding_net_pins true | false

Controls whether the shielding wires are connected to the standard cell power or ground (PG) pins.

By default (false), this command connects the shielding wires to the standard cell PG pins.

-ignore_shielding_net_rails true | false

Controls whether the shielding wires are connected to the standard cell rails.

By default (false), this command connects the shielding wires to the standard cell rails.

-coaxial_below true | false

Controls whether coaxial shielding is created below the shielded net segment layer.

By default (false), this command does not perform coaxial shielding below the shielded net segment layer.

-coaxial_above true | false

Controls whether coaxial shielding is created above the shielded net segment layer.

By default (false), this command does not perform coaxial shielding above the shielded net segment layer.

-coaxial_below_skip_tracks *number_of_tracks*

Controls the number of tracks to be left open between coaxial shielding segments created below the shielded net segment layer.

You can specify an integer value between 0 and 7. The default is 1, which spaces the coaxial shielding segments such that they leave at least one signal routing resource on the layer. To disallow signal routing below the shielded net segment layer, specify 0.

This option is used only when **-coaxial_below** is true.

-coaxial_above_skip_tracks *number_of_tracks*

Controls the number of tracks to be left open between coaxial shielding segments created above the shielded net segment layer.

You can specify an integer value between 0 and 7. The default is 1, which spaces the coaxial shielding segments such that they leave at least one signal routing resource on the layer. To disallow signal routing above the shielded net segment layer, specify 0.

This option is used only when **-coaxial_above** is true.

-pg_via_tie_effort_level low | medium | high

Specifies the effort level used to tie the shielding wires to the power and ground network with vias. This option affects the via connections from the shielding wires to PG pins and wires (straps, rails, and rings) that cross shielding wires on adjacent layers.

Valid values for this option are:

- * low

Creates a single via or via array connection regardless of the number of intersections between the shielding structure and a PG net on an adjacent layer. Use low effort to speed up the command.

- * medium (default)

Creates a via or via array for each intersection between an unconnected wire in the shielding structure and a PG net on an adjacent layer; however, it avoids creating vias or via arrays in the immediate vicinity of already connected shielding wires.

- * high

Creates a via or via array connection for each intersection between a shielding wire and a PG net on an adjacent layer. There is a runtime cost associated with using high effort.

DESCRIPTION

The **create_zrt_shield** command shields nets based on the associated shielding rules.

By default, the command performs same-layer shielding on all nets with predefined shielding rules. The router routes shielding wires based on the shielding widths and spacing defined in the shielding rules. The shielding wires are tied to the ground net, standard cell ground pins, and standard cell rails.

To explicitly specify the nets on which to perform shielding, use the **-nets** option. To tie the shielding wires to a named power or ground net, use the **-with_ground** option. To prevent connections to the standard cell PG pins, use the **-ignore_shielding_net_pins** option. To prevent connections to the standard cell rails, use the **-ignore_shielding_net_rails** option.

By default, the command does not perform coaxial shielding. To perform coaxial shielding above the shielded net segment layer, use the **-coaxial_above true** option. To perform coaxial shielding below the shielded net segment layer, use the **-coaxial_below true** option. When the tool performs coaxial shielding, the coaxial shielding segments

- * Are routed in the preferred layer direction
- * Use the defaultWidth layer attribute from the technology file.
- * Are spaced as specified by the **-coaxial_above_skip_tracks** option for coaxial shielding above the shielded net segment layer and as specified by the **-coaxial_below_skip_tracks** option for coaxial shielding below the shielded net segment layer

To remove existing shielding on specified nets, use the **-mode unshield** option. To remove existing shielding on the whole design, you should use the **remove_route_by_type -shield** command.

PREREQUISITES

Before you run this command, you must first define the shielding rules with the **define_routing_rule** command and assign these rules to the nets to be shielded with the **set_net_routing_rule** command.

This requirement applies when you do not specify the **-nets** option, you want to use a nondefault width or minimum spacing, you want to snap the shielding segments to the grid, or you want to reserve space for future postroute shielding during signal routing.

LIMITATIONS

There may be a discrepancy between DRC violations, reported by **create_zrt_shield** and **verify_zrt_route** commands. Please be advised that **create_zrt_shield** command only works with shielding shapes, thus the reported violations are detected only for shielding shapes, and not for other signal net routing. In order to get an accurate DRC violations report for all the nets, please run **verify_zrt_route** after **create_zrt_shield**.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example defines the shielding rules; assigns these shielding rules to all clock nets; and then shields the CLK_B1 and CLK_B5 clock nets with the VSS ground net, the CLK_B2 clock net with the GND ground net, and the CLK_B3 and CLK_B4 clock nets with the VDD power net. The shielding wires are not connected to the standard cell PG pins. The CLK_B5 net also has coaxial shielding above and below the shielded net segment layers and no signal routing resources are left between the

coaxial shielding segments.

```
prompt> define_routing_rule clock_net_shielding \  
-default_reference_rule -taper_level 0 -snap_to_track \  
-widths {METAL1 0.16 METAL2 0.2 METAL3 0.2 METAL4 0.2} \  
-spacing {METAL1 0.18 METAL2 0.21 METAL3 0.21 METAL4 0.21} \  
-shield_width {METAL1 0.16 METAL2 0.2 METAL3 0.2 METAL4 0.2} \  
-shield_spacing {METAL1 0.18 METAL2 0.21 METAL3 0.21 METAL4 0.21} \  
-via_cuts {via1 1X1 via2 1X1 via3 1X1}  
  
prompt> set_clock_nets [get_nets -filter "net_type == Clock" -hier]  
  
prompt> set_net_routing_rule -rule clock_net_shielding $clock_nets  
  
prompt> create_zrt_shield -with_ground VSS -nets {CLK_B1} \  
-ignore_shielding_net_pins true  
  
prompt> create_zrt_shield -with_ground GND -nets {CLK_B2} \  
-ignore_shielding_net_pins true  
  
prompt> create_zrt_shield -with_ground VDD -nets {CLK_B3 CLK_B4} \  
-ignore_shielding_net_pins true  
  
prompt> create_zrt_shield -with_ground VSS -nets {CLK_B5} \  
-coaxial_below true -coaxial_below_skip_tracks 0 \  
-coaxial_above true -coaxial_above_skip_tracks 0 \  
-ignore_shielding_net_pins true
```

SEE ALSO

```
define_routing_rule(2)  
remove_route_by_type(2)  
set_net_routing_rule(2)
```

current_design

Sets the working design.

SYNTAX

```
string current_design  
[design]
```

Data Types

design string

ARGUMENTS

design

Specifies the working or focal design for many dc_shell commands. If *design* is not specified or a period "." is specified, dc_shell returns to the current working design. If *design* refers to a design that cannot be found, an error is issued and the working design remains unchanged.

DESCRIPTION

The **current_design** command sets the working design for many dc_shell commands. Without arguments, **current_design** returns the name of the current working design.

To display designs currently available in dc_shell, use the **list_designs** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses **current_design** to show the current context and to change the context from one design to another:

```
prompt> current_design .  
Current design is 'TOP'.  
  
prompt> list_designs  
ADDER                FULL_SUBTRACTOR HALF_SUBTRACTOR TOP (*)  
FULL_ADDER          HALF_ADDER        SUBTRACTOR  
  
prompt> current_design ADDER  
Current design is 'ADDER'.  
  
prompt> current_design  
Current design is 'ADDER'.
```

The **current_design** command is a parameter of other dc_shell commands. In the following example, **remove_design** is used to delete the working design from dc_shell.

```
prompt> current_design
Current design is 'TOP'.

prompt> remove_design current_design()
Removing design 'TOP'

prompt> current_design
Error: Current design is not defined. (UID-4)
```

SEE ALSO

```
current_instance(2)
list_designs(2)
list_instances(2)
```

current_design_name

Returns the current design name.

SYNTAX

string **current_design_name**

DESCRIPTION

This command returns the string of the current design name.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses **current_design_name** to save the current design name to a variable.

```
prompt> current_design  
Current design is 'TOP'.  
{TOP}
```

```
prompt> set design_name [current_design_name]  
Information: Defining new variable 'design_name'. (CMD-041)  
TOP  
Current design is 'TOP'.
```

SEE ALSO

current_design(2)

current_instance

Sets the working instance object and enables other commands to be used on a specific cell in the design hierarchy.

SYNTAX

```
string current_instance  
[instance]
```

Data Types

instance string

ARGUMENTS

instance

Specifies the working cell in `dc_shell`. If *instance* is not specified, the focus returns to the top level of the current design. If *instance* is ".", `dc_shell` returns to the working instance. If *instance* is "..", the context is moved up one level in the instance hierarchy. If *instance* begins with "/", `dc_shell` returns to the working instance of the design whose name is after the "/". More complex examples of *instance* arguments are described below in **EXAMPLES**.

DESCRIPTION

Sets the working instance in `dc_shell`. An instance is a cell embedded in the hierarchy of a design; normally, you define an instance to set or get attributes on a cell.

To display the instances available at the current level of design hierarchy, use **list_instances**.

The **current_design** command changes the working design, setting the current instance to the top level of the new current design.

The **current_instance** command traverses the design hierarchy similar to the way the UNIX **cd** command traverses the file hierarchy. The **current_instance** command operates with a variety of *instance* arguments.

- If no *instance* argument is specified, the focus of `dc_shell` is returned to the top level of the hierarchy.
- If *instance* is ".", the current instance is returned and no change is made.
- If *instance* is "..", the current instance is moved up one level in the design hierarchy.

- If *instance* is a valid cell at the current level of hierarchy, the current instance is moved down to that level of the design hierarchy.

- Multiple levels of hierarchy can be traversed in a single call to **current_instance** by separating multiple cell names with slashes.

For example, **current_instance U1/U2** sets the current instance down two levels of hierarchy if both cells exist at the current levels in the design hierarchy.

- The **".."** directive can also be nested in complex *instance* arguments.

For example, the command **current_instance "../..MY_INST"** attempts to move the context up two levels of hierarchy, then down one level to the "MY_INST" cell.

NOTE: The **current_instance** command does not work on leaf cells. If you attempt to run **current_instance** on a leaf cell, an error will occur.

By default, the dc_shell prompt is "prompt>". The prompt can be set to show the working instance. If you embed the string "\$CI" in the value of the variable *shell_prompt*, the working instance is shown as part of the dc_shell prompt. The command to set *shell_prompt* in this case is:

```
shell_prompt = "dc_shell$CI> "
```

For the *shell_prompt* assignment to take effect from the beginning of a dc_shell session, define this variable in your ".synopsys_dc.setup" file.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses the **current_instance** command to move up and down the design hierarchy and the **list_instances** command to show the available instances at each point in the hierarchy.

```
prompt> current_design .  
Current design is 'TOP'.
```

```
prompt> list_instances  
U1 (ADDER)          U2 (SUBTRACTOR)
```

```
prompt> current_instance U1  
Current instance is 'TOP/U1'.
```

```
prompt> current_instance ".."  
Current instance is 'TOP/U1'.
```

```
prompt> list_instances  
U1 (FULL_ADDER) U2 (FULL_ADDER) U3 (FULL_ADDER) U4 (FULL_ADDER)
```

```
prompt> current_instance U3  
Current instance is 'TOP/U1/U3'.
```

```
prompt> current_instance "../U4"  
Current instance is 'TOP/U1/U4'.
```

```
prompt> current_instance  
Current instance is the top-level of design 'TOP'.
```

In the following example, changing the current design resets the **current_instance** to the top level of the new design hierarchy.

```
prompt> current_design  
Current design is 'TOP'.
```

```
prompt> current_instance "U2/U1"  
Current instance is 'TOP/U2/U1'.
```

```
prompt> current_design ADDER  
Current design is 'ADDER'.
```

```
prompt> current_instance .  
Current instance is the top-level of design 'ADDER'.
```

The next example shows how to modify the variable `shell_prompt` to show the current context in the `dc_shell` prompt. Set the `shell_prompt` in your `".synopsys_dc.setup"` file.

```
prompt> list shell_prompt  
shell_prompt = "prompt> "
```

```
prompt> current_instance .  
Current instance is 'EXAMPLE/U77/BLAT'.
```

```
prompt> shell_prompt = "icc_shell$CI> "  
"icc_shell$CI> "
```

```
icc_shell EXAMPLE/U77/BLAT> current_design TOP  
Current design is 'TOP'.
```

```
icc_shell TOP>
```

The following example uses **current_instance** to go to an instance of another design. The current design is set by the new design whose name is the name after the first slash of the given instance name.

```
prompt> current_design .  
Current design is 'TOP'.
```



```
prompt> current_instance U1  
Current instance is 'TOP/U1'.
```

```
prompt> current_instance "/TOP/U2"  
Current instance is 'TOP/U2'.
```

```
prompt> current_instance "/ALARM_BLOCK/U6"  
Current instance is 'ALARM_BLOCK/U6'.
```

SEE ALSO

```
current_design(2)  
list_designs(2)  
list_instances(2)
```

current_mw_cel

Gets (or sets) the working Milkyway design in the tool.

SYNTAX

```
collection current_mw_cel  
[mw_cel]
```

Data Types

mw_cel cel

ARGUMENTS

mw_cel

Specifies a Milkyway design to be set as the current Milkyway design. You can specify a Milkyway design by name, name pattern, or the Milkyway design collection's name. For example, *top* matches a Milkyway design named *top* in the current library and *top** matches all Milkyway designs that have names beginning with *top*. This option must specify only one Milkyway design. Multiple Milkyway designs cannot be set as the current Milkyway design. By default, the command uses the current working Milkyway design.

DESCRIPTION

This command gets (or sets) the working Milkyway design for many other commands. If you do not specify an *mw_cel* value and the *current_mw_cel* was previously set, the command returns the value of the current Milkyway design.

If you specify an *mw_cel* value, the command first determines if the specified Milkyway design is an open Milkyway design. If the Milkyway design is not open, it instructs you to open the Milkyway design first. If it is an open Milkyway design, the command sets the current Milkyway design from the specified Milkyway design.

The following three commands change the value of the current Milkyway design:

- The **open_mw_cel** command automatically sets the opened Milkyway design as the current Milkyway design.
- The **close_mw_cel** command clears the current Milkyway design if the closed Milkyway design is the current Milkyway design.
- The **close_mw_lib** command clears the current Milkyway design if the current Milkyway design belongs to that closed library.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses the **current_mw_cel** command to show the current context and changes the context from one Milkyway design to another:

```
prompt> current_mw_cel  
{"top"}
```

```
prompt> current_mw_cel [get_mw_cel ADDER]  
{"ADDER"}
```

```
prompt> current_mw_cel  
{"ADDER"}
```

The following example uses the **close_mw_cel** command to close the working Milkyway design from the tool:

```
prompt> current_mw_cel  
{"TOP"}
```

```
prompt> close_mw_cel  
1
```

```
prompt> current_mw_cel  
Error: No mw cel is open
```

SEE ALSO

```
close_mw_cel(2)  
copy_mw_cel(2)  
create_mw_cel(2)  
get_mw_cels(2)  
mw_cel_collection(2)  
open_mw_cel(2)  
remove_mw_cel(2)  
rename_mw_cel(2)  
save_mw_cel(2)
```

current_mw_lib

Gets the current Milkyway library.

SYNTAX

collection **current_mw_lib**

ARGUMENTS

None.

DESCRIPTION

This command gets the current Milkyway library. The **current_mw_lib** command returns a collection of the current Milkyway library, if one exists.

Many Milkyway library related commands use the current Milkyway library by default. The current Milkyway library is automatically set when the **open_mw_lib** command opens a Milkyway library to use.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example returns the current Milkyway library in the current session:

```
prompt> current_mw_lib
{"design"}
```

SEE ALSO

```
close_mw_lib(2)
copy_mw_lib(2)
create_mw_lib(2)
open_mw_lib(2)
rebuild_mw_lib(2)
rename_mw_lib(2)
report_mw_lib(2)
```

current_scenario

Sets the current scenario.

SYNTAX

```
string current_scenario  
[scenario_name]
```

ARGUMENTS

scenario_name
Specifies the name of the current scenario. If *scenario_name* is not specified, the tool returns to the current scenario.

DESCRIPTION

Sets the current scenario for many commands. Without arguments, **current_scenario** returns the name of the current scenario.

Multicorner-Multimode Support

This command uses information from active scenarios only.

EXAMPLES

The following example uses **current_scenario** to set the current scenario.

```
prompt> create_scenario MODE1  
Warning: Discarding all scenario specific information previously defined in t  
his session. (UID-1008)  
Current scenario is: MODE1  
1  
prompt> create_scenario MODE2  
Current scenario is: MODE2  
1  
prompt> create_scenario MODE3  
Current scenario is: MODE3  
1  
prompt> current_scenario MODE2  
Current scenario is: MODE2  
MODE2
```

SEE ALSO

```
all_scenarios(2)  
current_scenario(2)  
remove_scenario(2)
```

cut_objects

Cuts from or adds to the boundary of one or more geometric objects.

SYNTAX

```
new_objects cut_objects
{-bbox rect | -boundary boundary
  | -by collection}
[-channel extra_channel_spacing]
[-invert]
[-keep_placement]
[-keep_pad_to_core_distance]
objects
```

Data Types

<i>rect</i>	x1 y1 x2 y2
<i>collection</i>	new_objects
<i>extra_channel_spacing</i>	real
<i>objects</i>	collection

ARGUMENTS

-bbox *rect*

Specifies the bounding box by which to cut the objects boundary or to append to the boundary.

-boundary *boundary*

Specifies the list of points by which to cut the objects boundary or to append to the boundary.

The boundary must be rectilinear and must have at least four points.

-by *collection*}

Specifies the list of objects by which to cut the specified objects boundary or to append to the specified objects boundary.

These objects must be geometric (that is, have at least a bounding box) and can be rectilinear.

See the man page for the **get_edit_property** command for details on which objects support these properties.

-channel *extra_channel_spacing*

Specifies extra channel spacing by which to cut.

By default, this option is off.

-invert

Specifies to invert the cut so that instead of cutting from the objects boundary, the overlapping shape is added to the boundary.

By default, this option is off.

-keep_placement

This option is effective only when applied to the core or the die. It will be ignored if applied on any other object.

cut_objects

Cells in the core and IO pad cells are re-placed if necessary. The relative positions of the cells are not guaranteed to be preserved, although an attempt will be made to preserve them as much as possible. Furthermore, the output of the re-placement is not legalized. The user needs to explicitly legalize the placement using `legalize_placement`. The default is **false**.

`-keep_pad_to_core_distance`

This option is effective only when applied to the core or the die. It will be ignored if applied on any other object.

This maintains the distance between the core and the pad cells and by implication the distance between the core and the die, thus forcing changes to the core and the die to be made in tandem with each other. If pad cells are absent then the core-to-die distance will still be maintained.

The default is **false**.

`objects`

Specifies the list of objects which will be processed.

These objects must either be resizable or support a rectilinear boundary depending on the final result of the cut.

See the man page for the **`get_edit_property`** command for details on which objects support these properties.

By default, this option is off.

RETURNS

`new_objects`

Contains a list of the new split objects.

DESCRIPTION

This command cuts from or adds to the boundary of a set of objects by using an explicit bounding box, an explicit list of points, or the boundary of one or more specified objects.

When multiple objects are specified to be cut each object is treated independently.

When multiple objects are specified by which to cut, the result is the cumulative result of all cuts.

Note: Cutting takes place only if the bounding box, list of points, or object boundary by which to cut, overlaps with the boundary of the object being cut.

Snapping is done automatically using global snap settings.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example cuts the boundary of the selected objects by a bounding box

defined by the points {0 0} {100 100}.

```
prompt> cut_objects -bbox {0 0 100 100} [get_selection]
```

The following example cuts the boundary of the selected objects by a list of points which represent the bounding box {0 0} {100 100}.

```
prompt> cut_objects -boundary {{0 0} {100 0} {100 100} {0 100}} \  
[get_selection]
```

The following example cuts the boundary of all the non-fixed selected objects by the fixed selected objects:

```
prompt> set unfixed [filter [get_selection] "is_fixed==false"]  
prompt> set fixed [filter [get_selection] "is_fixed==true"]  
prompt> cut_objects -by $fixed $unfixed
```

SEE ALSO

`set_object_boundary(2)`
`set_object_shape(2)`

cut_row

Cuts rows from the current design.

SYNTAX

```
status cut_row
[-all]
[-area {{ll_x ll_y} {ur_x ur_y}}]
```

ARGUMENTS

-all
Specifies to cut all rows from the current design.
The option and **-area {{ll_x ll_y} {ur_x ur_y}}** are mutually exclusive.

-area {{ll_x ll_y} {ur_x ur_y}}
Specifies to cut rows in the specified area.
The option and **-all** are mutually exclusive.

DESCRIPTION

This command cuts rows from the *current design*. If the **-all** option is specified, all rows are cut in the design. If an area is given, the rows in the specified area are cut. The command returns a value of **1** if it succeeds, or **0** if the command fails.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example cuts rows in the specified rectangle:

```
prompt> cut_row -area {250 1710 1800 1730}
1
```

SEE ALSO

`add_row(2)`

date

Returns a string containing the current date and time.

SYNTAX

string **date**

DESCRIPTION

The **date** command generates a string containing the current date and time, and returns that string as the result of the command. The format is fixed as follows:

```
ddd mmm nn hh:mm:ss yyyy
```

Where:

ddd is an abbreviation for the day
mmm is an abbreviation for the month
nn is the day number
hh is the hour number (24 hour system)
mm is the minute number
ss is the second number
yyyy is the year

The **date** command is useful because it is native. It does not fork a process. On some operating systems, when the process becomes large, no further processes can be forked from it. With it, there is no need to call the operating system with **exec** to ask for the date and time.

EXAMPLES

The following command prints the date.

```
prompt> echo "Date and time: [date]"  
Date and time: Thu Dec  9 17:29:51 1999
```

SEE ALSO

define_antenna_accumulation_mode

Defines an antenna accumulation mode route rule.

SYNTAX

```
status define_antenna_accumulation_mode  
[mw_lib]  
[-cut_to_metal]  
[-metal_to_cut]
```

Data Types

mw_lib list

ARGUMENTS

mw_lib
 Specifies the Milkyway library to be updated. The value of *mw_lib* can be a library name or a one-element collection of a library. The *mw_lib* option is optional. The default is to use the current Milkyway library.

-cut_to_metal
 When -cut_to_metal is on, via ratios will be accumulated to metal ratios. Default is off.

-metal_to_cut
 When -cut_to_metal is on, metal ratios will be accumulated to cut ratios. Default is off.

DESCRIPTION

This command defines an antenna accumulation mode route rule and stores it in the library. The command returns a status indicating success or failure.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

```
prompt> define_antenna_accumulation_mode  
      -cut_to_metal -metal_to_cut  
1
```

SEE ALSO

report_antenna_rules(2)
remove_antenna_rules(2)

define_antenna_layer_ratio_scale

Creates an antenna layer ratio route rule.

SYNTAX

```
status define_antenna_layer_ratio_scale
[mw_lib]
-layer layer_name
-layer_scale layer_scale
-accumulate_scale accumulate_scale
```

Data Types

<i>mw_lib</i>	list
<i>layer_name</i>	string
<i>layer_scale</i>	float
<i>accumulate_scale</i>	float

ARGUMENTS

mw_lib
Specifies the Milkyway library to be updated. The value of *mw_lib* can be a library name or a one-element collection of a library. The *mw_lib* option is optional. The default is to use the current Milkyway library.

-layer *layer_name*
Name of the layer.

-layer_scale *layer_scale*
Layer scale factor.

-accumulate_scale *accumulate_scale*
Accumulation scale factor.

DESCRIPTION

This command creates an antenna layer ratio route rule. The command returns a status indicating success or failure.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

```
prompt> define_antenna_layer_ratio_scale -layer metal1
-layer_scale 1000.00 -accumulate_scale 2.00
```

SEE ALSO

`define_antenna_rule(2)`
`define_antenna_layer_rule(2)`
`report_antenna_rules(2)`
`remove_antenna_rules(2)`

define_antenna_layer_rule

Defines an advanced antenna rule for the specified layer and stores it in the library.

SYNTAX

```
status define_antenna_layer_rule
[mw_lib]
-mode mode
-layer layer_name
[-ratio ratio]
[-pratio pratio]
[-nratio nratio]
-diode_ratio diode_ratio
[-scale_factor scale_factor]
```

Data Types

<i>mw_lib</i>	list
<i>mode</i>	int
<i>layer_name</i>	string
<i>ratio</i>	float
<i>pratio</i>	float
<i>nratio</i>	float
<i>diode_ratio</i>	list
<i>scale_factor</i>	list

ARGUMENTS

mw_lib
Specifies the Milkyway library to be updated. The value of *mw_lib* can be a library name or a one-element collection of a library. By default, the command uses the current Milkyway library.

-mode mode
Defines the way antenna areas are computed. The valid values and their descriptions are as follows:

- 1 -Uses polygon area, ignoring all lower-layer segments.
- 2 -Uses polygon area, including all lower-layer segments to the input pins.
- 3 -Uses polygon area, including all lower-layer segments.
- 4 -Uses side-wall area, ignoring all lower-layer segments.
- 5 -Uses side-wall area, including all lower-layer segments to the input pins.
- 6 -Uses side-wall area, including all lower-layer segments.

Note that only one rule can be defined for each mode. If two commands contain the same mode number, the second command overwrites the first one.

-layer layer_name
Specifies the name of the valid metal layer or cut layer in the library.

-ratio ratio
Specifies the maximum allowable ratio of the antenna area to the gate area if the antenna is not protected by any diode. Any number is a valid value for

define_antenna_layer_rule

ratio.

-pratio *pratio*

Specifies the maximum allowable ratio of the antenna area to the p-gate area.

-nratio *nratio*

Specifies the maximum allowable ratio of the antenna area to the n-gate area.

-diode_ratio *diode_ratio*

Specifies the allowable ratio of the antenna area to the gate area if the antenna is protected by a diode. The valid values for *diode_ratio* are **{v0 v1 v2 v3 [v4]}**. If the output pin protection value specified in the Cell Library Format (CLF) file is **dp**, the allowable ratio for *diode_ratio* is the following:

$((dp + v1) * v2 + v3)$, if $(dp) > (v0)$
layer_max_ratio, if $(dp) \leq (v0)$

-scale_factor *scale_factor*

Specifies the scale factor of the metal area to the gate area. This option is only valid for diode mode 11 and diode mode 12.

DESCRIPTION

This command defines an advanced antenna rule for the specified layer and stores it in the library. The command returns a status indicating success or failure.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example defines an advanced antenna rule using a mode with a side-wall area, ignoring all lower-layer segments, for a metal layer named *metal1* with a diode ratio as shown.

```
prompt> define_antenna_layer_rule -mode 4 -layer metal1 -ratio 400 -  
diode_ratio {0.336 -0.5 400 2400}
```

SEE ALSO

`define_antenna_rule(2)`
`report_antenna_rules(2)`
`remove_antenna_rules(2)`

define_antenna_rule

Defines an advanced antenna rule for the specified mode and stores it in the library.

SYNTAX

```
status define_antenna_rule
[mw_lib]
-mode mode
-diode_mode diode_mode
[-metal_ratio metal_ratio]
[-cut_ratio cut_ratio]
[-metal_pratio metal_pratio]
[-metal_nratio metal_nratio]
[-cut_pratio cut_pratio]
[-cut_nratio cut_nratio]
[-protected_metal_scale metal_scale]
[-protected_cut_scale cut_scale]
```

Data Types

<i>mw_lib</i>	list
<i>mode</i>	integer
<i>diode_mode</i>	integer
<i>metal_ratio</i>	float
<i>cut_ratio</i>	float
<i>metal_pratio</i>	float
<i>metal_nratio</i>	float
<i>cut_pratio</i>	float
<i>cut_nratio</i>	float
<i>metal_scale</i>	float
<i>cut_scale</i>	float

ARGUMENTS

mw_lib Specifies the Milkyway library to be updated. The value of *mw_lib* can be a library name or a one-element collection of a library. The default is to use the current Milkyway library.

-mode mode Defines the way antenna areas are computed. The valid values and their descriptions are as follows:

- 1 -Uses a polygon area, ignoring all lower-layer segments.
- 2 -Uses a polygon area, including all lower-layer segments to the input pins.
- 3 -Uses a polygon area, including all lower-layer segments.
- 4 -Uses a side-wall area, ignoring all lower-layer segments.
- 5 -Uses a side-wall area, including all lower-layer segments to the input pins.
- 6 -Uses a side-wall area, including all lower-layer segments.

Note that only one rule can be defined for every mode. If two commands contain the same mode number, the second command overwrites the first one.

define_antenna_rule

`-diode_mode diode_mode`

Defines the protection capability of the diode. By default, all output pins are considered to be a diode. The valid values and their descriptions are as follows:

- 0** -Output pin cannot protect antenna.
- 1** -Any diode can provide unlimited protection.
- 2** -Diode protection is limited; if more than one diode is connected, the largest value of max-antenna-ratio for all diodes is used.
- 3** -Diode protection is limited; if more than one diode is connected, the sum of max-antenna-ratio for all diodes is used.
- 4** -Diode protection is limited; if more than one diode is connected, the sum of all diode-protection values for all diodes is used to compute the max-antenna-ratio.
- 5** -Diode protection is limited; the maximum diode-protection value for all diodes is used to calculate the equivalent gate area.
- 6** -Diode protection is limited; the sum of all diode-protection values for all diodes is used to calculate the equivalent gate area.
- 7** -Diode protection is limited; the maximum diode-protection value for all diodes is used to calculate the equivalent metal area.
- 8** -Diode protection is limited; the sum of all diode-protection values for all diodes are used to calculate the equivalent metal area.
- 9 through 12** -Reserved for various particular antenna modes.

`-metal_ratio metal_ratio`

Specifies the maximum allowable ratio for metal area to gate size if the metal layer is not defined by using the **define_antenna_layer_rule** command. If this value is zero, the ratio is ignored. The valid value is any non-negative number.

`-cut_ratio cut_ratio`

Specifies the maximum allowable ratio for cut area to gate size if the cut layer is not defined by using the **define_antenna_layer_rule** command. If this value is zero, the ratio is ignored. The valid value is any non-negative number.

`-metal_pratio metal_pratio`

Specifies the maximum allowable ratio for antenna area (metal) to p-gate area.

`-metal_nratio metal_nratio`

Specifies the maximum allowable ratio for antenna area (metal) to n-gate area.

`-cut_pratio cut_pratio`

Specifies the maximum allowable ratio for antenna area (cut) to p-gate area.

`-cut_nratio cut_nratio`

Specifies the maximum allowable ratio for antenna area (cut) to n-gate area.

`-protected_metal_scale metal_scale`

Specifies the value used to scale the area of the metal layer that is protected by the diode. The option is used only when the mode is 2 or 5. If this value is zero, the scale is ignored. The valid value is any non-negative number. By default, the value is set to 1.0.

`-protected_cut_scale cut_scale`

Specifies the value used to scale the area of the cut layer that is protected by a diode. The option is used when mode is 2 or 5 only. If this value is zero, the scale is ignored. The valid value is any non-negative number. By default, the value is set to 1.0.

DESCRIPTION

This command defines an advanced antenna rule for the specified mode and stores it in the library.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows the use of the command specifying a polygon area (ignoring all lower-layer segments), limited diode protection, maximum allowable ratio for metal area to gate size of 1000, and a maximum allowable ratio for cut area to gate size of 0.

```
prompt> define_antenna_rule -mode 1 -diode_mode 2\  
-metal_ratio 1000 -cut_ratio 0
```

SEE ALSO

```
define_antenna_layer_ratio_scale(2)  
define_antenna_layer_rule(2)  
remove_antenna_rules(2)  
report_antenna_rules(2)
```

define_bus

Creates a port bus or a net bus.

SYNTAX

```
status define_bus  
-type type_name  
-name full_bus_name  
-range {start_index end_bit}
```

Data Types

<i>type_name</i>	string
<i>full_bus_name</i>	string
<i>start_index</i>	integer
<i>end_bit</i>	integer

ARGUMENTS

-type *type_name*
Specifies the type of the object whose bus attribute is to be defined. Valid types are port and net.
This argument is required.

-name *full_bus_name*
Specifies the full path and the name of the bus. The specified name cannot be the same as any other bus. In other words, port bus names must be different from the names of ports, and net bus names must be different from the names of nets.
This argument is required.

-range {*start_index end_bit*}
Specifies the start index and the end index for the bus.
This argument is required.

DESCRIPTION

The **define_bus** command creates a bus object of the type port or net. It collects a list of ports or nets and put them into a bus.

Buses appear as multibit ports or multiwire nets in a design. This command groups any number of ports or nets into a bus object.

EXAMPLES

This example groups the A[0], A[1], A[2], and A[3] ports into a bus named A, with the bus starting from index 1 and ending by index 3. The ports must already exist in the design.

```
prompt> define_bus -type port -name A -range {0 3}
prompt> report_bus [get_ports A]
```

This example groups the B[6], B[5], and B[4] nets into a bus named B, with the bus starting from index 6 and ending by index 4. The nets must already exist in the design.

```
prompt> define_bus -type net -name B -range {6 4}
prompt> report_bus [get_nets B]
```

This example groups the ports U1/A[1], U1/A[2], and U1/A[3] in the MID subdesign into a bus named U1/A. U1 is a unique instantiation of MID.

```
prompt> define_bus -type port -name U1/A -range {1 3}
prompt> report_bus [get_pins U1/A]
```

This example groups the nets U1/NET_BUS[1] and U1/NET_BUS[0] in the MID subdesign into a bus named NET_BUS. U1 is a unique instantiation of MID.

```
prompt> define_bus -type net U1/NET_BUS -range {1 0}
prompt> report_bus [get_nets U1/NET_BUS]
```

SEE ALSO

```
get_nets(2)
get_ports(2)
get_pins(2)
undefine_bus(2)
report_bus(2)
```

define_io_antenna_area

Defines external antenna properties (antenna area) for a pin or terminal.

SYNTAX

```
status define_io_antenna_area  
-pin_name name  
-values {{layer area_value true | false} ...}
```

Data Types

name string

DATA TYPES

name string
layer string
area_value float

ARGUMENTS

-pin_name *name*
 Specifies a pin to set the new external antenna properties (antenna area).

-values {{layer area_value true | false} ...}
 Specifies a list of external antenna properties (antenna area) of the specified pin for each routing layer. Each entry in the list consists of a routing layer name, its external antenna area, and the reference area. Use the true keyword for sidewall area, and false for polygon area.

DESCRIPTION

This command sets external antenna properties (antenna area). These settings are persistent in the Milkyway design. To permanently save these properties, you must save cell upon exit.

EXAMPLES

The following example sets the external antenna properties (antenna area) of the pin CLK_in.

```
prompt> define_io_antenna_area -pin_name CLK_in -values \  
          {{metal1 100.5 true} {metal1 150 false} {metal2 200 true} \  
          {via1 10 false} {via2 20 false}}
```

SEE ALSO

define_io_gate_size(2)

```
define_io_diode_protection(2)
report_io_antenna_properties(2)
remove_io_antenna_properties(2)
extract_hier_antenna_property(2)
verify_zrt_route(2)
report_antenna_ratio(2)
```

define_io_diode_protection

Defines external antenna properties (diode protection) for a pin or terminal.

SYNTAX

```
status define_io_diode_protection  
-pin_name name  
-values {metal1_protection_value metal2_protection_value ...}
```

Data Types

metal1_protection_value float

Data Types

name string
metal1_protection_value float

ARGUMENTS

```
-pin_name name  
          Specifies a pin to set the external antenna properties (diode protection).  
  
-values {metal1_protection_value metal2_protection_value ...}  
          Specifies a list of external antenna properties (diode protection) for the  
          specified pin for each routing layer.
```

DESCRIPTION

This command sets external antenna properties (diode protection). These settings are persistent in the Milkyway design. To permanently save the settings, you must save cell upon exit.

EXAMPLES

The following two examples set the external antenna properties (diode protection) of the pin CLK_in. The first example sets metal1_diode protection: 1.2, metal2_diode protection: 1.3, metal3_diode protection: 1.5, and other metal4~15_diode protection: 1.5. prompt> **define_io_diode_protection -pin_name CLK_in -values {1.2 1.3 1.5}**

This example sets all metal_diode protection : 1.2.

```
prompt> define_io_diode_protection -pin_name CLK_in -values {1.2}
```

SEE ALSO

define_io_gate_size(2)
define_io_antenna_area(2)

```
report_io_antenna_properties(2)
remove_io_antenna_properties(2)
extract_hier_antenna_property(2)
verify_zrt_route(2)
report_antenna_ratio(2)
```


define_io_gate_size

Defines external antenna properties (gate size) for pins.

SYNTAX

```
status define_io_gate_size  
-pin_name name  
-values {metal1_gateSize metal2_gateSize ...}
```

Data Types

<i>name</i>	string
<i>metal1_gateSize</i>	float
<i>metal2_gateSize</i>	float

ARGUMENTS

```
-pin_name name  
    Specifies the pin to set external antenna properties (gate size).  
  
-values {metal1_gateSize metal2_gateSize ...}  
    Specifies a list of external antenna properties (gate size) for the specified  
    pin for each routing layer.
```

DESCRIPTION

This command sets external antenna properties (gate size). These settings are persistent in the Milkyway database. To save the settings, you must save cell upon exit.

EXAMPLES

The following two examples set the external antenna properties (gate size) of the pin CLK_in. This example sets metal1_gateSize: 1.2, metal2_gateSize: 1.3, metal3_gateSize: 1.5, other metal4~15_gateSize: 1.5

```
prompt>define_io_gate_size -pin_name CLK_in -values {1.2 1.3 1.5}
```

The example sets all metal_gateSize : 1.2.

```
prompt>define_io_gate_size -pin_name CLK_in -values {1.2}
```

SEE ALSO

```
define_io_diode_protection(2)  
define_io_antenna_area(2)  
report_io_antenna_properties(2)  
remove_io_antenna_properties(2)  
extract_hier_antenna_property(2)  
verify_zrt_route(2)
```

```
report_antenna_ratio(2)
```

define_name_rules

Defines a set of name rules for designs.

SYNTAX

```
status define_name_rules
name_rules
[-max_length length]
[-target_bus_naming_style bus_naming_style]
[-allowed allowed_chars]
[-restricted restricted_chars]
[-first_restricted first_chars]
[-last_restricted last_chars]
[-reserved_words reserves]
[-replacement_char char]
[-remove_chars]
[-equal_ports_nets]
[-inout_ports_equal_nets]
[-collapse_name_space]
[-case_insensitive]
[-special output_format]
[-prefix prefix_name]
[-map map_string]
[-type object_type] [-reset]
[-remove_internal_net_bus]
[-remove_port_bus]
[-check_bus_indexing]
[-check_bus_indexing_use_type_info]
[-rename_three_state_port_net]
[-check_internal_net_name]
[-remove_irregular_port_bus]
[-remove_irregular_net_bus]
[-flatten_multi_dimension_busses]
[-dont_change_bus_members]
[-dont_change_ports]
[-add_dummy_nets]
[-dummy_net_prefix dummy_nets_format]
[-dir_inout_as_in]
```

Data Types

<i>name_rules</i>	string
<i>length</i>	integer
<i>bus_naming_style</i>	string
<i>allowed_chars</i>	string
<i>restricted_chars</i>	string
<i>first_chars</i>	string
<i>last_chars</i>	string
<i>reserves</i>	list
<i>char</i>	string
<i>output_format</i>	string
<i>prefix_name</i>	string
<i>map_string</i>	string

object_type string
dummy_nets_format string

ARGUMENTS

name_rules

Specifies the name of the rules being defined. If named rules called *name_rules* do not exist, they are created.

-max_length length

Specifies the maximum length of a name, which must be 8 or more characters, except that a value of 0 resets the maximum length to its default (any length).

-target_bus_naming_style bus_naming_style

Specifies the target bus naming style for all object types, to which the current bus naming style is to be changed. For this option to succeed, set the **bus_naming_style** variable with the current bus naming style. For an example, see the section entitled "Changing the Bus Naming Style". By default, the target bus naming style is set to be the same as the current bus naming style. If the target bus naming style is different from the current bus naming style, after the rule is applied through the **change_names** command, the bus naming style becomes the target bus naming style, but this does not occur automatically. If more **change_names** commands need to be issued later, be careful that the bus naming style is in transition from one rule to another rule and needs to be set manually to reflect the real bus style inside the .db file during the transition.

-allowed allowed_chars

Specifies the set of characters allowed in names, which must be 10 or more characters. The format of the *allowed_chars* string is provided in the "DESCRIPTION" section. By default, any printable character is allowed in a name.

-restricted restricted_chars

Specifies the set of characters not allowed in names. The format of the *chars* string is provided in the "DESCRIPTION" section. By default, any printable character is allowed in a name.

-first_restricted first_chars

Specifies a set of characters that are not allowed as the first character in a name. The format of the *chars* string is provided in the "DESCRIPTION" section. By default, any printable character is allowed as the first character of a name.

-last_restricted last_chars

Specifies a set of characters not allowed as the last character in a name. The format of the *chars* string is described in the "DESCRIPTION" section. By default, any printable character is allowed as the last character of a name.

-reserved_words reserves

Specifies a list of words that cannot be used as a name. The *reserves* value contains words that are considered reserved in your target system. By default, there are no reserved words.

define_name_rules

-replacement_char *char*

Specifies the character used to replace characters not allowed in names, as specified by the **-allowed** or **-restricted** option. By default, the replacement character is the underscore character (`_`).

-remove_chars

Specifies that characters not allowed in names, as specified by the **-allowed** or **-restricted** option, are removed rather than replaced. By default, illegal characters are replaced by the character specified with **-replacement_char**.

-equal_ports_nets

Specifies that nets connected to non-inout ports must have the same name as their connecting port. If a net is connected to more than one port, the net's name is not changed and a warning is issued. By default, nets are not required to match the names of the non-inout ports to which they are connected.

-inout_ports_equal_nets

Specifies that nets connected to inout ports must have the same name as their connecting port. If a net is connected to more than one port, the net's name is not changed and a warning is issued. By default, nets are not required to match the names of the inout ports to which they are connected.

-collapse_name_space

Specifies that the name space of ports, cells, and nets is the same. No port, cell, or net in a design can have the same name. By default, ports, cells, and nets each have their own name space.

-case_insensitive

Specifies that the case of characters is not significant when comparing names. For example, the name *example* (lowercase) is equivalent to *EXAMPLE* (uppercase). If two names are equivalent within a name space, one of the names must be changed. By default, the case of a name is significant.

-special *output_format*

Specifies to use special rules pertaining to a specific target system when changing names. By default, no special rules apply. The possible values for *output_format* are as follows:

verilog
vhdl
sge
vhdl93
sge_vhdl
siff

-prefix *prefix_name*

Specifies a prefix used when changing a name. By default, the prefixes are **P** for ports, **U** for cells, and **N** for nets.

Use this option only when the **change_names** command needs to create a completely new name to ensure the name's uniqueness. See Step 3 of the "Naming Rules" section for more information.

-map *map_string*

Provides a way to change objects in addition to the the previously specified name rules. This option specifies the name mapping and replacement rules as

defined in the *map_string*. See "Mapping Rules" under the "Naming Rules" section for details.

-type *object_type*

Specifies that the rules being defined apply only to the given type of objects. Allowed values for *object_type* are **port**, **cell**, or **net**. By default, the defined rules apply to all object types.

-reset

Resets all rules to their default values. When all rules are set to the defaults, no restrictions are imposed on the names.

-remove_internal_net_bus

Bit-blasts all internal net buses.

-remove_port_bus

Bit-blasts all port buses. Net buses that connect to ports are also bit-blasted.

-check_bus_indexing

Checks the indices of buses and bit-blasts the incomplete buses.

-check_bus_indexing_use_type_info

Checks bus indexing based on the bus type information and bit-blast for incomplete port buses.

-rename_three_state_port_net

Renames three-state port nets so they have different names from the ports on the inout port connections.

-check_internal_net_name

Checks and renames nets that have the same name as some ports, but the nets are not connected to these ports.

-remove_irregular_port_bus

Bit-blasts any port bus that contains irregular members.

-remove_irregular_net_bus

Bit-blasts any net bus that contains irregular members.

-flatten_multi_dimension_busses

Flattens multi-dimension ports, net buses, and arrays so writing out is easier.

-dont_change_bus_members

Specifies not to change the elements that are in either a port bus or a net bus. This option overrides the previous four options (**-remove_irregular_port_bus**, **-remove_irregular_net_bus**, **-flatten_multi_dimension_busses**, and **-dont_change_bus_members**). If this option is set, these four options are reset to the default value of **false**. In most cases, this option should remain **false**.

-dont_change_ports

Specifies not to change elements that are set by the **-type** option to **port**.

define_name_rules

-add_dummy_nets
 Adds dummy nets for unconnected pins. The **-dummy_net_prefix** option specifies the format to name dummy nets. The format defaults to `SYNOPSIS_UNCONNECTED_%d`.

-dummy_net_prefix *dummy_nets_format*
 Sets the prefix for the dummy net naming convention. The *dummy_nets_format* value specifies the format to name the dummy nets.

-dir_inout_as_in
 Instructs the tool to treat all instances with the INOUT direction as if they were set to IN. The default behavior is to treat them as direction OUT.

DESCRIPTION

The **define_name_rules** command defines a set of rules for naming design objects. Name rules are used by the **change_names** and **report_names** commands. The **report_name_rules** command displays a listing of the name rules currently defined in the shell.

Name rules can be defined in multiple calls to the **define_name_rules** command. For an example of multiple calls, see the "EXAMPLES" section.

The **-type** option enables you to define rules that apply to a specific object type, such as port, cell, or net). Each call to **define_name_rules** is additive or overrides previous calls for a specific name rules.

Naming Rules

This section describes the effects of specific options when defining name rules with **define_name_rules**.

Maximum Length Rules

The maximum number of characters in a name can be restricted with the **-max_length** option.

Names shorter than the specified maximum length remain unchanged. Names longer than the specified maximum length are modified so that their lengths are less than or equal to the maximum.

Fixed names are guaranteed to be unique within the design. The following steps are applied in succession to fix a name to meet the uniqueness criteria:

- Step 1

Truncation - Names are truncated to meet the maximum length. The truncated name is accepted if it is unique within the design. For example, if the name is `THIS_NAME_IS_TOO_LONG_TO_BE_ACCEPTABLE` and the maximum number of characters is 16, the resulting name is `THIS_NAME_IS_TOO`.

- Step 2

Truncation with index - Names are truncated. The truncated name is appended with an index in an attempt to make it unique. If a unique name is found, it is accepted. If the name from the previous example, `THIS_NAME_IS_TOO`, is not unique, this the tool tries `THIS_NAME_IS_T1`, `THIS_NAME_IS_T2`, and continues up to, `THIS_NAME_IS_T99`. The first unique name is accepted.

- Step 3

Basic names - If both previous steps fail, the original name is deleted and names of the form *<prefix><index>* are generated until a unique name is found. The *<prefix>* is defined using the **-prefix** option. Examples are N1, N2, and so on.

If all three steps fail to generate a unique name, the original name is retained and a warning message is issued.

Character Restriction Rules

Specify characters that make up names with the **-allowed**, **-restricted**, **-first_restricted**, and **-last_restricted** options.

To specify the character set allowed, use **-allowed** or **-restricted**. Either of these options overrides the previous values of **-allowed**, **-restricted**, **-first_restricted**, and **-last_restricted**.

In conjunction with **-allowed** or **-restricted**, use **-first_restricted** and **-last_restricted** to further restrict the available characters for the first and last character of a name. The **-first_restricted** and **-last_restricted** options depend on **-allowed** and **-restricted**, and modify the set of available characters for their special cases. The set of characters available for first and last characters cannot exceed that defined for all other characters.

The *allowed_chars*, *restricted_chars*, *first_chars*, and *last_chars* strings contain characters that can be used (they are **-allowed**), or cannot be used (they are **-restricted**) in design object names. The dash (-) character indicates a range of ASCII characters. To include a literal dash character in a *chars* parameter, precede it with 2 backslashes (\-). Blank spaces are disregarded. A minimum of 10 characters must be allowed or an error is reported. The order of the characters in the *chars* string parameters is not significant.

The following example specifies a character set consisting of uppercase characters and an underscore.

```
prompt> define_name_rules -allowed "A-Z _"
```

In the following example, some punctuation characters are restricted. The dash (-) character is restricted with the character sequence \-. The double quotation marks (") are restricted with the sequence First characters cannot be lowercase.

```
prompt> define_name_rules -restricted "!@#$$%^&*()\-\" \
      -first_restricted "a-z"
```

The following example attempts to restrict the character set to fewer than 10 characters. An error is reported.

```
prompt> define_name_rules -allowed "ABCDE"
```

If a design object name contains a character that is restricted for its object type, the following steps are taken to fix the name:

- Step 1

Remove the character - If **-remove_chars** is specified in the name rules, the offending character is removed.

- Step 2

Change the case - The case of characters is changed to meet character restrictions. The change is accepted if the new character is allowed. For example, if names are restricted to uppercase characters, the name *sum_port* is changed to *SUM_PORT*.

- Step 3

Replace the character - A restricted character is changed to the value of **-replacement_char**. For example, if punctuation characters are not allowed and the replacement character is an underscore, the name U\$1 is changed to U_1.

- Step 4

Uniquify the name - If the name created in either Step 1 or Step 2 is not unique, an index is appended to the new name.

Reserved Word Rules

Prohibit specific names from becoming design object names by using the **-reserved_words**. The parameter to this option is a list of names that are not allowed as design object names.

The following example uses **-reserved_words**. The words DESIGN, MODULE, START, and END are designated as reserved. The case of the names in this list is significant.

```
prompt> define_name_rules \  
-reserved_words {"DESIGN", "MODULE", "START", "END"}
```

Case Insensitive Rules

If **-case_insensitive** is specified, the case of the characters in a name is not significant when comparing names.

If **-case_insensitive** is specified, the case of characters in reserved words is not significant. For example, if **-case_insensitive** is used and the string MODULE is specified as a reserved word, the name module is also considered a reserved word and changed.

Port and Net Rules

Make the name of a net connected to a port equal to the name of that port with **-equal_ports_nets**. If this substitution causes a conflict, no names are changed and a warning message is issued. If a net is connected to more than one port, no names are changed and a warning message is issued.

Name Space Rules

By default, name spaces of ports, cells, and nets are separate. Names of ports and cells must be unique within a design. A port, a cell, and a net within a design can share the same name.

Name spaces for ports, cells, and nets can be shared using the **-collapse_name_space** option. Ports, cells, and nets within a design must be unique across all objects. Conflicting names are modified by appending digits to their name. The **-collapse_name_space** option cannot be specified with **-equal_ports_nets**, because the two options conflict.

Type-Specific Rules

Name rules can be tailored for ports, cells, and nets object types with the **-type** option. The options to **define_name_rules** that can be specified by object type are as follows:

- max_length**
- allowed**
- restricted**
- first_restricted**
- last_restricted**
- replacement_char**

-remove_chars
-prefix

To specify the maximum length restriction on ports different than the maximum length for cells, make multiple calls to **define_name_rules**.

In the following example, the maximum length of ports is set to 12. The maximum length of cells is set to 8. The maximum length of nets is unrestricted.

```
prompt> define_name_rules EXAMPLE -max_length 12 -type port
```

```
prompt> define_name_rules EXAMPLE -max_length 8 -type cell
```

Special Rules

Define name rules that are for a specific target system and cannot be modeled with the general **define_name_rules** options. The systems supported by the **-special** option are *sge*, *vhdl*, *vhdl93*, *sge_vhdl*, and *siff*.

Specifying **-special vhdl** adds one rule to make names conform to the rules of VHDL. This option does not allow consecutive underscores in a name.

Specifying **-special vhdl93** adds one rule to make names conform to the rule of VHDL93. This option adds support for an extended identifier. An extended identifier is a sequence of characters that are defined by the VHDL93 character set and are written between two backslashes (\ \).

Specifying **-special sge** adds one rule to make names conform to the rules of the SGE system. This option does not allow net names that start with the characters BBBB_ and NNNN_. These names are reserved by the SGE system.

Specifying **-special sge_vhdl** is a combination of the previous two rules. The **change_names** command enforces **-collapse_name_space** and **-equal_ports_nets**, but because these two rules conflict with each other, the **sge_vhdl** rule applies under the following conditions:

- The **-collapse_name_space** rule first checks the object class type. If the object is a port, it checks to see if the name is used by any reference names. If the object is a cell or a design, it verifies that the name is not used previously by any reference or port. If the object is a net, it checks the reference, port, and cell name spaces. If there is a name conflict, it changes the name so that it is unique. Refer to the **-collapse_name_space** rule description for details.
- The **-equal_ports_nets** rule is applied to a scalar net or a net object whose owning bus is not of a single width. Refer to the **-equal_ports_nets** rule description for details.
- Specifying **-special siff** adds one rule to make names conform to the rules for the "Synopsys Integrator For Falcon Framework."

All of the special rules previously given share the same name rules in addition to their own name rules:

- The names of bused ports or nets follow the *bus_naming_style*, except that *sge_vhdl* always uses "%s(%d)" instead of *bus_naming_style*.
- The names of bused ports or nets owned by the same bus have the same base name. The bus index order follows the index part defined in the owning bus.

- If a net does not belong to a busbag and it contains a separator character defined in the **bus_naming_style** variable, this separator character changes to **_** (underscore) and the ending separator is removed. For example, the net name `sample[24][3]` changes to the name `sample_24_3`.

Specifying **-special** does not create all of the rules necessary for VHDL, SGE, or SIFF compliance. Only the non-general rules previously described are created. To **define_name_rules** to constrain names to fully meet the requirements of SGE and VHDL, use the other options in conjunction with **-special**.

To avoid conflict, do not merge special name rules with name rules that you defined. When defined name rules conflict, **change_names** could generate names that are not compliant to your name rule. When there is a conflict among name rules, define each one of those name rules using a different name in **define_name_rules** and apply each of them using **change_names**. The last rule applied generates names that override names generated by previous rules.

Mapping Rules

The **-map** option specifies the name mapping and replacement rules as defined in the *map_string*. The mapping string has the following format:

```
{{"pattern", "replacement"} [, {"pattern", "replacement"}]*
```

The mapping string is grouped into pairs. There can be any number of pairs in a mapping string. Each pair is enclosed by a **{}** (curly braces) and separated by a **,** (comma). The **change_names** and **report_names** commands apply all of them according to the order specified in the *map_string*.

The first member of each pair is the *pattern* string, which is used to match against the object name. The second member is the *replacement* string, which is used to replace the matched portion of the object name. Both the *pattern* and *replacement* string must begin and end **"** (double quotation marks). There is a **,** (comma) between *pattern* and *replacement* string.

The following example has the first occurrence of any portion of a object name that contains **_reg** replaced by an empty string and the first occurrence of any portion of a object name that contains **A** replaced by **a** in the new name.

```
prompt> define_name_rules my_rule -map {"_reg", ""}, {"A", "a"}
```

The pattern string also provides limited regular expression capability. The following 4 special characters are supported to make the mapping strings more specific.

- The **\$** (dollar sign) indicates the end of string.
- The **^** (caret) indicates the beginning of string.
- The ***** (asterisk) indicates a wildcard matching 0 or more characters.
- The **?** (question mark) indicates a wildcard matching 1 character.

The following example defines a rule that any name ending with **_reg** is replaced by **in** at the end string:

```
prompt> define_name_rules my_rule0 -map {"_reg$", "in"}
```

The following example defines a rule that any name beginning with **_reg** is replaced by **in** at the beginning of the string:

```
prompt> define_name_rules my_rule1 -map {"^_reg", "in"}
```

The following example defines a rule that the first occurrence of any substring of a name that begins with **_r** and ends with **g** is replaced by the **in** string:

```
prompt> define_name_rules my_rule2 -map {"_r*g", "in"}}
```

The following example defines a rule that the first occurrence of any substring of a name that begins with **_r** followed any one character, then ending with **g** is replaced by **in**:

```
prompt> define_name_rules my_rule3 -map {"_r?g", "in"}}
```

Once a mapping rule is defined, you can only use **-reset** to reset the mapping rule. You cannot redefine the same matching pattern within the same rule. For example, assume you define the following rule:

```
prompt> define_name_rules my_rule4 -map {"A", "X"}}
```

If you then decide to replace all occurrences of **A** with **Z**, you must first execute a reset and then redefine the name rules as follows:

```
prompt> define_name_rules my_rule4 -reset
```

```
prompt> define_name_rules my_rule4 -map {"A", "Z"}}
```

To insert escape characters into names, you can also use the **-map** option. This is useful prior to generating reports, if the report output is processed by a program that requires certain characters to be escaped. For example, if names contain the **/** (slash) character, the tool cannot differentiate between a **/** used in a name and a **/** that denotes a change in hierarchy. Defining a rule with the following command replaces any **/** in a name with **\/** (backslash slash).

```
prompt> define_name_rules -map {"/", "\\/"}}
```

The name **HAS/SLASH** inside hierarchy **A** writes out as **A/HAS\/SLASH**.

Use **{}** around the *map_string* to reserve the content of the *map_string* as shown in the example:

```
prompt> define_name_rules my_rule -map {{{"_reg", ""}, {"A", "a"}}}
```

Default Rules Values

Once defined, you can reset name rules to their default values with the **-reset** option. Name rules with all default values impose no restrictions on design object names.

Default values for specific rules are shown below. A set of name rules called **ALL_DEFAULTS** is defined with no options. The **report_name_rules** command displays the default value of each specific rule.

```
prompt> define_name_rules ALL_DEFAULTS
```

```
prompt> report_name_rules ALL_DEFAULTS
```

```
*****
Report : name_rules
Name Rules : ALL_DEFAULTS
Version: v3.0
Date    : Fri Sep 27 09:36:52 1991
*****
```

```

Rules Name: ALL_DEFAULTS
  Equal port and net names: false
  Collapse name space: false
  Case insensitive: false
  Special rules: none
  Reserved words: none

```

	Max	Repl	Rem				
Rules	Type	Len	Char	Chrs	Prefix	Allowed	Chars
Port Rules	none	'_'	no	P		No restrictions	
Cell Rules	none	'_'	no	U		No restrictions	
Net Rules	none	'_'	no	N		No restrictions	

Many rules can be reset individually by specifying empty values for the option. The following table shows how to specify the default value for these fields. Options not listed in the table must be reset with the **-reset** option.

Option	Default Value
-max_length	0
-restricted	" "
-allowed	" "
-first_restricted	" "
-last_restricted	" "
-reserved_words	{}
-replacement_char	" "
-prefix	" "

Applying Name Rules

When names are changed using the **change_names** or **report_names** command, name rules are applied sequentially even though they are specified concurrently in the **define_name_rules** command. Name rules are applied in order, so rules applied later might overwrite previous names rules and there will be cases where names after **change_names** or **report_names** do not conform to all rules specified.

This section describes the order in which name rules are applied, and the results you can expect.

The name rules are applied in the following order:

1. Map rule: rules that are applied to an object name string. The rule in this category is **-map_rule**.
2. Character rules: rules that are applied to object name characters. Rules in this category are **-allowed**, **-restricted**, **-first_restricted**, and **-last_restricted**. To support the character rules, use **-remove_char** and **-replacement_chars**.
3. Design rules: rules that are applied to object names in terms of whole design naming methodology. Rules in this category are **-equal_ports_nets** (for net objects only), **-special**, **-max_length**, and **-reserve_words**.
4. Meta rules: rules that extend beyond one of the previous single categories. Rules in this category are **-collapse_name_space**, **-case_insensitive**, **-prefix**, and **-type**. Since rules applied earlier might be overwritten by later rules, apply the "must have" rule last. The name rules are applied to object names according to the order shown above. That is, the string rules are applied first, then the character rules, and the design rules are last. Meta rules are used for supporting those rules during

name changes. Rules within the same category are applied following the same order as the prior list. Because later rules might override previous ones, to preserve the priority it is better to use those rules that cause conflict separately. Among these rules, use mapping primarily.

During name changes, the objects are categorized into three types: port, cell, and net. Port objects are changed first, then cell objects, then net objects. When there is a name conflict among a port name, a cell name, and a net name, the port name has highest priority (it is not changed). The net and port names must be changed to unique names.

Changing the Bus Naming Style

When **define_name_rules** changes the bus naming style, use the **bus_naming_style** variable to identify the bus naming style used by the current database.

For example, the following script changes the bus naming style from [] to < >:

```
prompt> define_name_rules change_bus_naming_style \
        -target_bus_naming_style "%s<%d>" \
        bus_naming_style="%s[%d]"

prompt> change_names -rule change_bus_naming_style -hierarchy
```

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example defines a name rule called *SIMPLE* that requires design names in uppercase. The *SIMPLE* name rule is used as the default name rule by **change_names**.

```
prompt> define_name_rules SIMPLE -allowed "A-Z _"

prompt> default_name_rules = SIMPLE

prompt> change_names
Information: Using name rules 'SIMPLE'.
Information: 153 names changed in design 'MY_DESIGN'.
```

The following example defines name rules called *MY_RULES* using multiple calls to **define_name_rules**. The length of names is restricted to 16 characters. The example defines the allowed character set and a replacement character:

```
prompt> define_name_rules MY_RULES -max_length 16

prompt> define_name_rules MY_RULES -replacement_char "X"
```

```
prompt> define_name_rules MY_RULES -allowed "A-Z _"
```

In the following example, multiple calls to **define_name_rules** use most of the options. The **report_name_rules** command displays the effect of these calls.

```
prompt> define_name_rules SMORG -collapse_name_space
```

```
prompt> define_name_rules SMORG -case_insensitive
```

```
prompt> define_name_rules SMORG -reserved_words {IN,OUT,INOUT}
```

```
prompt> define_name_rules SMORG -case_insensitive
```

```
prompt> define_name_rules SMORG -type port -max_length 16 \  
-allowed "A-Z a-z 0-9_*" \  
-replacement_char ""
```

```
prompt> define_name_rules SMORG -type cell -prefix "CELL" \  
-restrict "()[\{\}]"
```

```
prompt> define_name_rules SMORG -type net -allowed "A-Za-z_*" \  
-first_restrict "0-9" -last_restrict "0-9" \  
-replacement_char ""
```

```
prompt> report_name_rules SMORG
```

```
*****  
Report : name_rules  
Name Rules : SMORG  
Version: v3.0  
Date   : Fri Sep 27 11:00:09 1991  
*****
```

```
Rules Name: SMORG  
  Equal port and net names: false  
  Collapse name space: true  
  Case insensitive: true  
  Special rules: none  
  Reserved words: {IN, OUT, INOUT}
```

		Max	Repl	Rem		
Rules	Type	Len	Char	Chrs	Prefix	Allowed Chars
Port	Rules	16	'*'	no	P	Use "A-Z a-z 0-9_*"
Cell	Rules	none	'_'	no	CELL	Don't use "()[\{\}]"
Net	Rules	none	'*'	no	N	Use "A-Za-z_*"
						First: Don't use "0-9"
						Last: Don't use "0-9"

SEE ALSO

change_names(2)
report_name_rules(2)

```
report_names(2)
bus_naming_style(3)
```


define_proc_attributes

Defines attributes of a Tcl procedure, including an information string for help, a command group, a set of argument descriptions for help, and so on. The command returns the empty string.

SYNTAX

```
string define_proc_attributes  
proc_name  
[-info info_text]  
[-define_args arg_defs]  
[-command_group group_name]  
[-hide_body]  
[-hidden]  
[-dont_abbrev]  
[-permanent]
```

Data Types

<i>proc_name</i>	string
<i>info_text</i>	string
<i>arg_defs</i>	list
<i>group_name</i>	string

ARGUMENTS

proc_name
Name of the existing procedure.

-info *info_text*
Provides a help string for the procedure. This is printed by the **help** command when you request help for the procedure. If you don't specify *info_text*, the default is "Procedure".

-define_args *arg_defs*
Defines each possible procedure argument for use with **help -verbose**. This is a list of lists where each list element defines one argument.

-command_group *group_name*
Defines the command group for the procedure. By default, procedures are placed in the "Procedures" command group.

-hide_body
Hides the body of the procedure from **info body**.

-hidden
Hides the procedure from **help** and **info proc**.

-dont_abbrev
The procedure can never be abbreviated. By default, procedures can be abbreviated, subject to the value of the **sh_command_abbrev_mode** variable.

-permanent

Defines the procedure as permanent. You cannot modify permanent procedures in any way, so use this option carefully.

DESCRIPTION

The **define_proc_attributes** command associates attributes with a Tcl procedure. These attributes are used to define help for the procedure, locate it in a particular command group, and protect it.

When a procedure is created with the **proc** command, it is placed in the Procedures command group. It has no help text for its arguments. You can view the body of the procedure with **info body**, and you can modify the procedure and its attributes. The **define_proc_attributes** command allows you to change these aspects of a procedure.

Note that the arguments to Tcl procedures are all named positional arguments. They can be programmed with default values, and there can be optional arguments by using the special argument name *args*. The **define_proc_attributes** command does not relate the information that you enter for argument definitions with **-define_args** to the actual argument names. If you are describing anything other than positional arguments, it is expected that you are also using **parse_proc_arguments** to validate and extract your arguments.

The *info_text* is displayed when you use the **help** command on the procedure.

Use **-define_args** to define help text and constraints for individual arguments. This makes the help for the procedure look like the help for an application command. The value for **-define_args** is a list of lists. Each element has this format:

arg_name option_help value_help data_type attributes

The *arg_name* is the name of the argument. *option_help* is a short description of the argument. The *value_help* is typically the argument name for positional arguments, or a one word description for dash options. It has no meaning for a boolean option. The *data_type* and *attributes* are optional and will be used for option validation. The *data_type* can be any of: string, list, boolean, int, float, or one_of_string. The default is string. The *attributes* is itself a list that can have any of the following entries:

- "required" - This argument must be specified. This attribute is mutually exclusive with optional.
- "optional" - Specifying this argument is optional. This attribute is mutually exclusive with required.
- "value_help" - Indicates that the valid values for a one_of_string argument should be listed whenever argument help is shown.

- "values {<list of allowable values>}" - If the argument type is `one_of_string`, you must specify the "values" attribute.

- "merge_duplicates" - When this option appears more than once in a command its values are concatenated into a list of values. The default behavior is that the right-most value for the option specified is used.

- "remainder" - Specifies that any additional positional arguments should be returned in this option. This option is only valid for string option types, and by default the option is optional. You can require at least one item to be specified by also including the required option.
The default for *attributes* is "required".

Change the command group of the procedure using **-command_group**. Protect the contents of the procedure from being viewed by using **-hide_body**. Prevent further modifications to the procedure by using **-permanent**. Prevent abbreviation of the procedure by using **-dont_abbrev**.

EXAMPLES

The following procedure adds two numbers together and returns the sum. For demonstration purposes, unused arguments are defined.

```
prompt> proc plus {a b} {return [expr $a + $b]}
prompt> define_proc_attributes plus -info "Add two numbers"
? -define_args { {a "first addend" a string required}
  {b "second addend" b string required}
  {"-verbose" "issue a message" "" boolean optional}}
```

```
prompt> help -verbose plus
Usage: plus      # Add two numbers
    [-verbose]      (issue a message)
    a                (first addend)
    b                (second addend)

prompt> plus 5 6
11
```

In the following example, the procedure `argHandler` accepts an optional argument of each type supported by **define_proc_attributes**, then displays the options and values received.

```
proc argHandler {args} { parse_proc_arguments -args $args results
  foreach argname [array names results] { echo $argname = $results($argname)
}
}

define_proc_attributes argHandler
  -info Arguments processor
  -define_args { {{-Oos oos help AnOos one_of_string
    {required value_help {values {a b}}}}
```

```
{-Int int help AnInt int optional}  
{-Float float help AFloat float optional}  
{-Bool "bool help"  boolean optional}  
{-String "string help" AString string optional}  
{-List "list help" AList list optional}}  
{-IDup int dup AIDup int {optional merge_duplicates}}}
```

SEE ALSO

help(2)
info(2)
sh_command_abbrev_mode(3)

define_routing_rule

Defines design-specific, nondefault routing rules that are stored in the design database.

SYNTAX

```
status define_routing_rule
rule_name
[-reference_rule_name ref_rule_name
 | -default_reference_rule]
[-widths {layer_name_and_width_pairs}]
[-spacings {layer_name_and_spacing_pairs}]
[-shield]
[-shield_widths {layer_name_and_shield_width_pairs}]
[-shield_spacings {layer_name_and_shield_spacing_pairs}]
[-snap_to_track]
[-via_cuts {via_definition_list}]
[-taper_level tapering_level]
[-multiplier_width layer_width]
[-multiplier_spacing layer_spacing]
[-taper_distance tapering_distance]
```

Data Types

<i>rule_name</i>	string
<i>ref_rule_name</i>	string
<i>layer_name_and_width_pairs</i>	list
<i>layer_name_and_spacing_pairs</i>	list
<i>layer_name_and_shield_width_pairs</i>	list
<i>layer_name_and_shield_spacing_pairs</i>	list
<i>tapering_level</i>	float
<i>layer_width</i>	float
<i>layer_spacing</i>	float
<i>tapering_distance</i>	float

ARGUMENTS

rule_name
Specifies the name for the new nondefault routing rule.
This is a required argument.

-reference_rule_name ref_rule_name
Specifies the name of the reference (source) rule to use.
The **-reference_rule_name** and **-default_reference_rule** options are mutually exclusive; you must specify only one. By default, the command uses the default routing rule as the reference rule.

-default_reference_rule
Specifies that the default routing rule is used as the reference rule.
The **-reference_rule_name** and **-default_reference_rule** options are mutually exclusive; you must use only one.

`-widths {layer_name_and_width_pairs}`

Specifies the routing width for each named routing layer. Each entry in the list is comprised of a routing layer name and its width in microns, separated by a space. The width portion of the pair is a floating point number. You can specify only a single width per layer.

If you do not specify the width for a layer, the tool sets the width based on the reference rule.

`-spacings {layer_name_and_spacing_pairs}`

Specifies the minimum different-net spacing allowed between wires for each named routing layer. Each entry in the list is comprised of a routing layer name and its spacing in microns, separated by a space. The spacing portion of the pair is a floating point number. You can specify only a single spacing per layer.

If you do not specify the minimum spacing for a layer, the tool sets the minimum spacing based on the reference rule.

`-shield`

Defines shielding with default spacing and default width for the new nondefault routing rule.

`-shield_widths {layer_name_and_shield_width_pairs}`

Specifies the shielding width for each named routing layer. Each entry in the list is comprised of a routing layer name and its shielding width in microns, separated by a space. The shielding width portion of the pair is a floating point number. You can specify only a single width per layer.

If you do not specify the shielding width for a layer, the tool sets the shielding width based on the reference rule.

`-shield_spacings {layer_name_and_shield_spacing_pairs}`

Specifies the minimum shield spacing allowed between wires for each named routing layer. Each entry in the list is comprised of a routing layer name and its shield spacing in microns, separated by a space. The shield spacing portion of the pair is a floating point number. You can specify only a single spacing per layer.

If you do not specify the shield spacing for a layer, the tool sets the shield spacing based on the reference rule.

`-snap_to_track`

Snaps the shielding wires to the track.

`-via_cuts {via_definition_list}`

Creates vias automatically during routing, using the via definitions.

For Zroute, each entry in the *via_definition_list* argument consists of the routable via name, its horizontal and vertical cut count numbers, and its rotation setting, separated by spaces. The cut number portion of the triplet is in the format *mXn* or *mxn*, where *m* is the horizontal cut number and *n* is the vertical cut number, separated by the X or x character. The rotation portion of the triplet is specified by R or NR, where R indicates a rotated via and NR indicates an unrotated via. The default rotation value is NR. Only those modes and rotations that are explicitly specified are allowed by the nondefault routing rule. For example, to allow all via array modes (swapped and unswapped) and rotations (rotated and unrotated) for a 1x2 via array named VIAHH, you must enter the following definition:

`-via_cuts {{VIAHH 1x2 NR} {VIAHH 1x2 R} {VIAHH 2x1 NR} {VIAHH 2x1 R}}`

For the classic router, each entry in the *via_definition_list* argument consists of the routable via name and its horizontal and vertical cut count numbers, separated by a space. The cut number is in the format *mXn* or *mxn*, where *m* is the horizontal cut number and *n* is the vertical cut number. The classic router supports only a single via definition for each layer; if you specify multiple vias for a given layer, the first definition is used and additional definitions are ignored. If you specify rotation information, it is ignored by the classic router.

You must ensure that the specified vias exist in the reference routing rule; otherwise, the command fails.

If you do not specify a via for a layer, the tool uses the default vias specified in the technology file for that layer.

-taper_level *tapering_level*

Defines the tapering level.

Valid values for the tapering level are

- 0

Enables pin-based tapering.

- 1 (the default)

Disables tapering on all pins.

- Positive integer greater than 1 Enables fanout-based tapering with up to the specified number of widths. In this case, the router uses the default width at the input pin and the maximum width, as defined in the routing rule, is used at the output pin and at the top-level pin.

-multiplier_width *layer_width*

Defines the layer width multiplier. You do not have to know the units or exact metal width currently defined in the library.

For example, to define double width, specify **-multiplier_width 2.0**.

-multiplier_spacing *layer_spacing*

Defines the layer spacing multiplier. You do not have to know the units and exact layer spacing currently defined in the library.

For example, to define double spacing, specify **-multiplier_spacing 2.0**.

-taper_distance *tapering_distance*

Sets the maximum length in microns for tapering wires.

The tapering wire width is the default width for the respective layer, unless you set the **pin_taper_mode** Zroute detail route option.

DESCRIPTION

This command defines design-specific, nondefault routing rules that are stored in the design database.

Use the **define_routing_rule** command to define width and spacing rules and the via types associated with them, as needed, without predefining them in the physical library.

Use the **set_net_routing_rule** command to assign nondefault routing rules to specific nets.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example defines a nondefault routing rule named `new_rule` that uses the default routing rules as the reference rule:

```
prompt> define_routing_rule new_rule -default_reference_rule \  
-widths {m1 0.8 m4 0.9} -spacings {m1 1.0 m4 1.0}
```

The following example defines a nondefault routing rule named `new_rule` that specifies multiple contacts with both rotated and unrotated vias for certain layers. Note that multiple contacts and rotation are supported only for Zroute.

```
prompt> define_routing_rule new_rule \  
-via_cuts {{VIA1_HH 2x1 NR} {VIA1_HH 2x1 R} {VIA1_HV 2x1 NR} {VIA1_HV 2x1 R}}
```

The following example reports the nondefault rule named `new_rule`:

```
prompt> define_routing_rule new_rule  
prompt> report_routing_rules new_rule
```

SEE ALSO

```
remove_routing_rules(2)  
report_net_routing_rules(2)  
report_routing_rules(2)  
set_net_routing_rule(2)
```


define_scaling_lib_group

Defines a scaling library group to support voltage and/or temperature scaling.

SYNTAX

```
status define_scaling_lib_group  
[-name name]  
[lib_file_names]
```

Data Types

<i>name</i>	string
<i>lib_file_names</i>	list

ARGUMENTS

-name *name*
Specifies a name for the scaling library group. The name can be used in a subsequent **set_scaling_lib_group** command.

lib_file_names
Specifies the set of library files that comprise the scaling library group. Use file names to ensure they can be found using **search_path**.

DESCRIPTION

The **define_scaling_lib_group** command defines a group of libraries that the tool uses for interpolation for voltage and/or temperature scaling.

You can define more than one group to cover different portions of a design, but the groups cannot share libraries. Each library can only be part of one scaling library group.

A minimum of two libraries is required for one-dimensional scaling, such as voltage or temperature. A minimum of four libraries is required for two-dimensional scaling, such as voltage and temperature.

The scaling library group defined in the current session will not be written back to the design. In the new session, the same scaling library group settings must be manually reapplied.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example defines a scaling library group for two voltage domains about the nominal:

```
prompt> define_scaling_lib_group -name g1 {0.9.db 1.1.db 1.3.db}
```

SEE ALSO

```
create_scenario(2)  
remove_scaling_lib_group(2)  
set_operating_conditions(2)  
set_scaling_lib_group(2)
```

define_user_attribute

Defines a new user-defined attribute.

SYNTAX

```
int define_user_attribute  
-type data_type  
-class class_list  
[-range_min min]  
[-range_max max]  
[-one_of values]  
[-quiet]  
attr_name
```

Data Types

<i>data_type</i>	string
<i>class_list</i>	values
<i>min</i>	double
<i>max</i>	double
<i>values</i>	list
<i>attr_name</i>	string

ARGUMENTS

-type *data_type*
Specifies the data type of the attribute. The supported data types are **string**, **int**, **float**, **double**, and **Boolean**.

-class *class_list*
Specifies the list of class names for the user-defined *attr_name* attribute. Valid classes are design, port, cell, net, etc.

-range_min *min*
Specifies the minimum value for numeric ranges. This option is valid only when the **-type** *data_type* is **int** or **double**. Specifying a minimum constraint without a maximum constraint creates an attribute that accepts a value greater than or equal to *min*.

-range_max *max*
Specifies the maximum value for numeric ranges. This option is valid only when the **-type** *data_type* is **int** or **double**. Specifying a maximum constraint without a minimum constraint creates an attribute that accepts a value less than or equal to *max*.

-one_of *values*
Provides a list of allowable strings. This option is valid only when the data type is *string*.

-quiet
Turns off the warning message that would otherwise be issued if the attribute or classes are improper.

attr_name
Specifies the name of the attribute.

DESCRIPTION

This command defines a new attribute. Use the **list_attributes** command to list the attributes that you have defined. The return value is set to **1** if the operation is successful; otherwise, it is set to **0**.

The definition for a user-defined attribute can be persistent if it has been operated on to a specified object and stored into the database.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example defines *attr_1* as greater than or equal to **2** and less than or equal to **3.2** on **cell** and **net** classes:

```
prompt> define_user_attribute -class {cell net} -type double \  
-range_max 3.2 -range_min 2 attr_1  
Info:User-defined attribute 'attr_1' on class 'cell'.  
Info:User-defined attribute 'attr_1' on class 'net'.  
1
```

The following example defines *attr_2* to **string** with a value of either **true** or **false** on **net** class:

```
prompt> define_user_attribute -class net -type string \  
-one_of {true false} attr_2  
Info:User-defined attribute 'attr_2' on class 'net'.  
1
```

The following example shows how to list the attribute definitions using the **list_attributes** command:

```
prompt> list_attributes  
*****  
Report : List of Attribute Definitions  
Design :  
Version:  
Date   : Mon Dec 22 17:08:51 2003  
*****
```

Properties:
A - Application-defined
U - User-defined
I - Importable from db (for user-defined)

Attribute Name	Object	Type	Properties	Constraints

attr_1	cell	double	U	2 to 3.2
attr_1	net	double	U	2 to 3.2
attr_2	net	string	U	true, false

SEE ALSO

get_attribute(2)
list_attributes(2)
remove_attribute(2)
set_attribute(2)

define_via

Creates a special via that is not defined in the physical library.

SYNTAX

```
int define_via  
via_name  
-rect {layer_name X1 Y1 X2 Y2}
```

ARGUMENTS

via_name
Specifies the name of a special via.

-rect {*layer_name* *X1* *Y1* *X2* *Y2*}
Defines layer geometries for the via. You can specify multiple **-rect** options. Each one specifies a rectangle on layer *layer_name*. The list for the option starts with *layer_name* followed by four x y coordinates: one set for the lower-left corner and the other for the upper-right corner. Coordinates are in micron units.

DESCRIPTION

Creates a special via that is not defined in the physical library. The created via is added into VIA_DEF (see pdef file) for the current design.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command creates a via between layers named *METAL1* and *METAL2*.

```
prompt> define_via my_via12 -rect \  
{METAL1 1 1 4 4} -rect {CUT12 2 2 3 3} \  
-rect {METAL2 1 1 4 4}
```

SEE ALSO

`create_net(2)`
`report_net(2)`

define_zrt_redundant_vias

Sets options for redundant via insertion.

SYNTAX

```
status define_zrt_redundant_vias
[-from_via {list_of_from_vias}]
[-to_via {list_of_to_vias}]
[-to_via_x_size {list_of_contact_numbers}]
[-to_via_y_size {list_of_contact_numbers}]
[-to_via_weights {list_of_weights}]
```

Data Types

<i>list_of_from_vias</i>	list of strings
<i>list_of_to_vias</i>	list of strings
<i>list_of_contact_numbers</i>	list of integers
<i>list_of_weights</i>	list of integers

ARGUMENTS

-from_via {*list_of_from_vias*}
Specifies the vias to be replaced during redundant via insertion.
The default value is an empty list.

-to_via {*list_of_to_vias*}
Specifies the vias to be inserted during redundant via insertion.
The default value is an empty list.

-to_via_x_size {*list_of_contact_numbers*}
Specifies the number of contacts for the vias inserted during redundant via insertion.
The default value is an empty list.

-to_via_y_size {*list_of_contact_numbers*}
Specifies the number of contacts for the vias inserted during redundant via insertion.
The default value is an empty list.

-to_via_weights {*list_of_weights*}
Specifies the weights of contacts for the vias inserted during redundant via insertion. The weight is an integer between 1 and 10 (inclusive). Via mappings having higher weight are preferred over via mappings having lower weight.
The default value is 1 for all via mappings.

DESCRIPTION

The **define_zrt_redundant_vias** command sets options for redundant via insertion.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets the options for via replacement with double vias:

```
prompt> define_zrt_redundant_vias -from_via {via1 via2} \  
        -to_via_x_size {1 1} -to_via_y_size {2 2}
```

SEE ALSO

```
insert_zrt_redundant_vias(2)
```


delete_operating_conditions

Deletes a specific set of operating conditions from a library.

SYNTAX

```
status delete_operating_conditions  
-library library_name  
-name op_cond_name
```

Data Types

<i>library_name</i>	string
<i>op_cond_name</i>	string

ARGUMENTS

```
-library library_name  
    Specifies the name of the library that stores the operating conditions.  
  
-name op_cond_name  
    Specifies the name of the set of operating conditions.
```

DESCRIPTION

The **delete_operating_conditions** command deletes a specific set of operating conditions from the specified library.

To create a new set of operating conditions, use the **create_operating_conditions** command.

To set operating conditions on the current design, use the **set_operating_conditions** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example deletes a specific set of operating conditions called BEST in the a_lib library.

```
prompt> delete_operating_conditions -library a_lib -name BEST
```

SEE ALSO

```
create_operating_conditions(2)  
set_operating_conditions(2)
```

derive_constraints

Propagates design environment, constraints, and attribute settings from the top-level design to the specified subdesigns.

SYNTAX

```
int derive_constraints  
[-attributes_only]  
[-verbose]  
[-budget]  
cell_list
```

ARGUMENTS

- attributes_only**
Propagates only the attributes. By default, both constraints and attributes are propagated.
- verbose**
Enables additional messages during propagation.
- budget**
Performs RTL budgeting to come up with delay constraints.
- cell_list**
Specifies the list of cells to which the environment is to be propagated. The following validations are performed on the specified cells:
- Cell is hierarchical
 - Cell is unique or is a master instance
 - Reference design is in Design Compiler memory

DESCRIPTION

Propagates design environment, constraints, and attribute settings from the top-level design to the specified subdesigns. Although this command works on both mapped and unmapped designs, design budgeting provides more accurate constraints for mapped designs.

This command uses the **target_library** variable. Ensure that this variable is set before running the command.

Multicorner-Multimode Support

This command uses information from the current scenario only.

SEE ALSO

`write_environment(2)`
`target_library(3)`

derive_mpc_macro_options

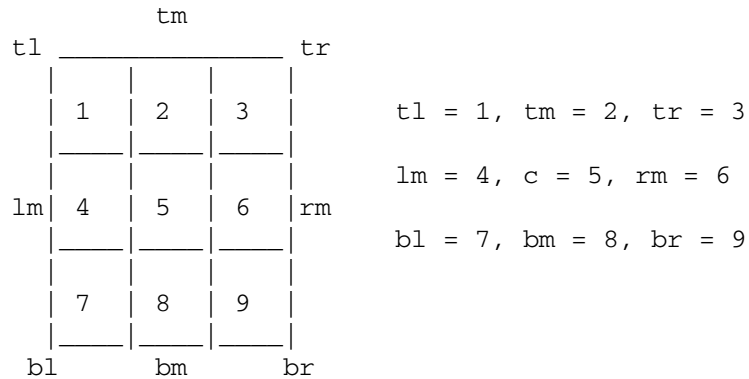
Derives placement constraints for a specific macro or derives macro array constraints for a group of macros.

SYNTAX

```
int derive_mpc_macro_options  
[-output filename]  
[-append]  
[-array]  
[-footprint]  
[-array_name array_name]  
[-location {exact | anchor_bound | anchor_offset | anchor_orient | none}]  
[-apply]  
[-verbose]  
list_of_macros
```

ARGUMENTS

- output *filename***
Specifies the file where the derived scripts are written. If this option is not specified, the tool applies the derived constraints directly on the design.
- append**
Appends to the current file without overwriting it. If this option is not specified, the current file is overwritten.
- array**
Derives the given macros as a macro array. The name of the array can be given by the **-array_name** option. If **-array_name** is not specified, the tool generates a random name. This option is only valid when more than one macro is selected. The tool scans the current macro locations to determine the array structure. The **set_mpc_macro_array** command is written out to specify the macro array. If the placement is too irregular to be recognized as an array, the tool does not finish deriving macro array and quits. Use the **-verbose** option to see how the tool looks at the possible array structure.
- footprint**
Allows multiple orientations with the same footprint.
- array_name *array_name***
Specifies the name for the macro array. It implies the **-array** flag if **-array** is not specified.
- location {exact | anchor_bound | anchor_offset | anchor_orient | none}**
Specifies how to derive the location constraint for a specific macro or the derived macro array (with **-array** specified). For one macro cell, the **exact** value extracts the macro location and orientation by **set_cell_location**, **rotate_objects -to** and **set_dont_touch_placement**. For other options, the appropriate **set_mpc_macro_options** are written. The **anchor_bound** value derives the **anchor_bound** constraint for a macro (**-anchor_bound**).



The **anchor_bound** is determined per the 9 regions shown above. The center of the cell falls into one of the regions and that region will be the **anchor_bound** for this cell. The **anchor_offset** value derives **anchor_bound** and the offset distance to the concerned anchor point (**-anchor_bound**, **-x_offset**, and **-y_offset**). The **anchor_orient** value derives **anchor_bound** and orientation (**-anchor_bound**, **-legal_orientation**).

For a macro array, the bounding box (including all macros) is used like the cell box to determine **anchor_bound** or offset for the array. The **set_mpc_macro_options array_name** is output. Only **anchor_bound** and **anchor_offset** are valid for macro array. The **none** value directs the tool to not to write any macro constraints regarding the location. The default for this option is **none**.

-apply

Directly annotate the derived constraints to design objects.

-verbose

Sets debug mode in which the tool will print more information on input and output data.

list_of_macros

Specifies the macros to derive constraints for. This is a required argument.

DESCRIPTION

This command derives placement constraints for a specific macro or macro array constraints for a group of macros. The **-array** option can be used with or without the **-location** option. With **-location**, the tool derives **anchor_bound** or **anchor_offset** for the macro array. Without **-location**, the location of the macro array is irrelevant.

Multicorner-Multimode Support

This command is not supported in a multi-scenario design flow.

EXAMPLES

The following example shows how to derive a macro array for *RAM1*, *RAM2*, and *RAM3*:

```
prompt> derive_mpc_macro_options [get_cells RAM1 RAM2 RAM3] \  
       -array my_ram_array -location anchor_bound
```

SEE ALSO

```
create_placement(2)  
derive_mpc_options(2)  
derive_mpc_port_options(2)  
set_mpc_macro_array(2)  
set_mpc_macro_options(2)  
set_mpc_options(2)  
set_mpc_port_options(2)  
report_mpc_macro_array(2)  
report_mpc_macro_options(2)  
report_mpc_options(2)  
report_mpc_port_options(2)
```

derive_mpc_options

Derives design level floorplan information and generic information for the ports and macros.

SYNTAX

```
int derive_mpc_options
[-output filename]
[-append]
[-origin ll | center]
[-core exact | relative | height_only | width_only]
[-pnets exact | model | none]
[-ports exact | relative | side | none]
[-macros exact | anchor_bound | anchor_orient | anchor_offset | none]
[-footprint]
[-corner_keepout exact | relative | none]
[-apply]
[-verbose]
```

Data Types

filename string

ARGUMENTS

- output *filename***
Specifies the file where the derived scripts are written. If this option is not specified, the tool applies the derived constraints directly on the design.
- append**
Appends to the current file without overwriting it. If this option is not specified, the current file is overwritten.
- origin *ll* | *center***
Specifies how to derive origin of the core area. The values are either **ll** (lower-left) or **center**. The coordinates of the lower-left corner are output to the **set_mpc_option -origin {x y}** command. If **center** is specified, **set_mpc_option -shift_to_center** is added. If there is a die area, **-io_margin** options are also output. The default is **ll**.
- core *exact* | *relative* | *height_only* | *width_only***
Specifies how to derive the core area. The valid values **exact**, **relative**, **width_only**, and **height_only**. The **set_mpc_option** command is written out to reflect the appropriate options specified. The **exact** value writes out the physical size of the core width and height (**-core_width**, **-core_height**). The **relative** value writes out the core parameters in the form of utilization and aspect ratio (**-utilization** and **-aspect_ratio**). If **height_only** or **width_only** are specified, the corresponding parameter (height or width) is output along with the utilization factor. The default for this option is **relative**.

-pnets exact | model | none

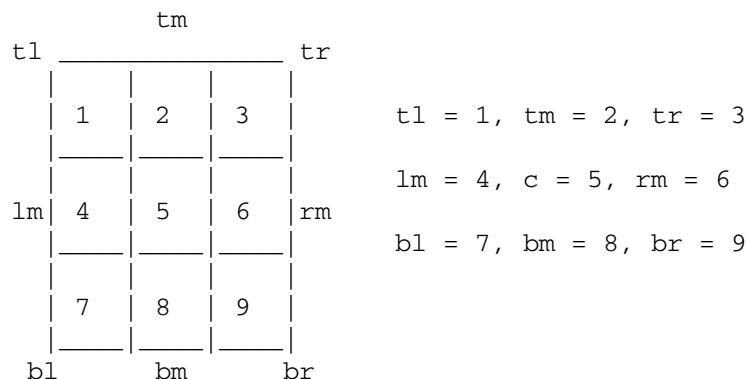
Specifies to derive the current floorplan power grid environment. The **exact** value writes out the exact pnet topology with **create_net -only_physical**, **create_routing_path**, and **define_via** commands. The **model** value derives pnet patterns and outputs the **set_mpc_pnet_options** command for each pnet object. The **none** value directs the tool to not write out any commands to represent pnet. The default value for this option is **model**.

-ports exact | relative | side | none

Specifies how to derive port location constraints. The **exact** value writes out the exact port location and port shape to the **set_port_location** command. The **relative** value writes out the relative coordinates (which are a percentage of the core width or core height) and the layer, if defined. The **set_mpc_port_options -coordinate -layer** command is used to reflect the information. The **side** value extracts the side where the port is located and the layer, if defined. The **set_mpc_port_options -side -layer** command is written out. The **none** value directs the tool to not write out the port constraints. The default for this option is **side**.

-macros exact | anchor_bound | anchor_orient | anchor_offset | none

Specifies how to derive all of the macros in the current floorplan environment. The **exact** value extracts macro location and orientation by the **set_cell_location**, **rotate_objects -to**, and **set_dont_touch_placement** command. For other options, the appropriate **set_mpc_macro_options** are written. The **anchor_bound** value derives the **anchor_bound** constraint for a macro (**-anchor_bound**).



The **anchor_bound** is determined by the 9 regions shown above. The center of the cell falls into one of the regions and that region is the **anchor_bound** for this cell. The **anchor_offset** value derives **anchor_bound** and the offset distance to affected anchor point (**-anchor_bound**, **-x_offset**, **-y_offset**). The **anchor_orient** value derives **anchor_bound** and orientation (**-anchor_bound**, **-legal_orientation**). The **none** value directs the tool not to write macro constraints. The default value for this option is **anchor_bound**.

-footprint

Allows multiple orientations with the same footprint.

-corner_keepout exact | relative | none

Specifies how to derive port keepouts for corners of the floorplan. The **set_mpc_options -corner_keepout** is written out. The **exact** value directs the tool to keep the actual distance for each corner to the closest port in the

horizontal and vertical direction as the corner keepout. The **relative** value uses the same principle, except the distance is not absolute; it is turned into a percentage of the core height or width. The **none** option does not extract a corner keepout constraint. The default value for this option is **none**.

-apply

Specifies to directly annotate the derived constraints to design objects.

-verbose

Sets debug mode in which the tool prints more information on input and output data.

DESCRIPTION

This command derives design level physical constraints and generic information for port and macro locations. The input for this command is a design with floorplan and/or placement information. The output of this command is a script file containing commands to set physical constraints. Switches in this command specify how to derive this information. For ports and macros, this command only concerns generic location constraints for all of the ports or macros. For an individual port, port group, macro, or macro array, use the **derive_mpc_port_options** or the **derive_mpc_macro_options** command.

Multicorner-Multimode Support

This command is not supported in a multi-scenario design flow.

EXAMPLES

The following example shows how to derive a floorplan for the current design. This involves utilization and aspect ratio for the core area. Since the **-port side** and **-macros anchor_bound** are the required settings and these are also the defaults, the following command meets the requirements:

```
prompt> derive_mpc_options
```

This following command set produces the same results as the example above:

```
prompt> derive_mpc_options -core relative -origin 11 \  
-ports side -macros anchor_bound
```

SEE ALSO

```
create_placement(2)  
derive_mpc_port_options(2)  
derive_mpc_macro_options(2)  
report_mpc_macro_options(2)  
report_mpc_options(2)  
report_mpc_port_options(2)
```

```
set_mpc_macro_options(2)
set_mpc_options(2)
set_mpc_port_options(2)
```

derive_mpc_port_options

Derives the physical constraints for a specific port or a group of ports.

SYNTAX

```
int derive_mpc_port_options  
[-output filename]  
[-append]  
[-group]  
[-group_name group_name]  
[-order]  
[-location {exact | relative | side | none}]  
[-apply]  
[-verbose]  
port_list
```

Data Types

<i>filename</i>	string
<i>group_name</i>	string
<i>port_list</i>	list or collection

ARGUMENTS

- output *filename***
Specifies the file where the derived scripts will be written. This is an optional option. If not specified, the tool will apply the derived constraints on design directly.
- append**
Indicates to append to the current file and not to overwrite it. If this options is not specified then the current file will be written over.
- group**
Specifies to derive port group constraint for the given ports. **-group_name** can be used to specify the name for the port group. The **set_mpc_port_options** command will be written out to place the selected ports as a group. To be a group, the selected ports should be on same side. If they are not on same side, the tool will error out with error message. Space between port will be checked and if they are not evenly spaced, the minimum space will be used as port group pitch.
- group_name *group_name***
Specifies the name for the derived port group. It implies **-group** flag.
- order**
Indicates to derive the order between the group of ports. The ports in output command **set_mpc_port_options** will be ordered per clockwise or counter-clockwise according to their current locations.
- location {exact | relative | side | none}**
Specifies how to derive port location constraint. If concern is individual

port (without -group), The 'exact' option writes out the exact port location and port shape to command set_port_location. The 'relative' option writes out the relative coordinates (being a percentage of the core/die width or core height) and port shape if defined. Command set_mpc_port_options -coordinate will be used to reflect these information. The 'side' option will extract the side where the port is located and port shape if defined. set_mpc_port_options -side will be written out. If concerned is a port group (with -group), the option of -location relative | exact derives start_location for the port group. No port shape will be derived. The 'none' option directs the tool not to write out port constraints or not to write - start_location for a port group. The default for this switch is none.

-apply

Directly annotate the derived constraints to design objects.

-verbose

Sets debug mode in which the tool will print more information on input and output data.

port_list

Specifies the ports to derive constraints for. This is a required argument.

DESCRIPTION

This command will be used to specify more specific constraints regarding a port or a group of ports. Group cannot occur across the corner of the core area. If a user tries this, an error will be issued. User should perform multiple group extraction if ports are along the corners. The group can be used with location or without location. With location, this option will tell the group startpoint in the constraint (-start_location) to be an exact or relative coordinate. Without - location, start_location is not cared.

Multicorner-Multimode Support

This command is not supported in a multi-scenario design flow.

EXAMPLES

The following example shows how to derive a bus pin group DATA[0-9].

```
prompt> derive_mpc_port_options [get_port DATA*] -group -order -location relative
```

SEE ALSO

```
derive_mpc_options(2)
derive_mpc_macro_options(2)
set_mpc_options(2)
set_mpc_port_options(2)
set_mpc_macro_options(2)
report_mpc_options(2)
report_mpc_port_options(2)
report_mpc_macro_options(2)
```

derive_mpc_port_options

```
create_placement(2)
```

derive_pg_connection

Connects power and ground pins and tie-off pins to the power and ground nets. It supports both manual mode, where you specify the power and ground nets, and automatic mode, where the tool derives the power and ground nets from the power domain connections. Automatic mode is supported only for multivoltage designs with UPF descriptions and requires logic libraries with power and ground pins.

SYNTAX for Manual Mode

status **derive_pg_connection**

```
[-tie]
[-power_net net_name]
[-ground_net net_name]
[-power_pin pin_name]
[-ground_pin pin_name]
[-cells cells]
[-create_ports none | top | all]
```

Data Types

<i>net_name</i>	string
<i>pin_name</i>	string
<i>cells</i>	collection

SYNTAX for Automatic Mode

status **derive_pg_connection**

```
[-create_nets]
[-tie]
[-all]
[-reconnect]
[-verbose]
```

ARGUMENTS

-tie

Connects the tie-high and tie-low pins. The power and ground network must exist prior to connecting the tie-high and tie-low pins.

In manual mode, the tool connects the tie-off pins to the power and ground nets specified in the **-power_net** and **-ground_net** options, respectively.

In automatic mode, the tool connects the tie-off pins to the proper power nets based on the power connections of their cell instances. A tie-high pin follows the connection of its related power pin if the **related_power_pin** attribute is defined on the pin in the library cell; otherwise, the tie-high pin of a regular cell is connected to the primary power net of the cell's power domain and the tie-high pin of an always-on cell is connected to the backup power net.

The **-tie** option is mutually exclusive with the **-all** option; you can specify only one. If you do not specify either option, **derive_pg_connection** connects

only the power and ground pins.

-power_net net_name

Specifies the name of the power net to use for power and tie-high connections. Using this option invokes the tool in manual mode.

-ground_net net_name

Specifies the name of the ground net to use for ground and tie-low connections. Using this option invokes the tool in manual mode.

-power_pin pin_name

Specifies the name of the power pin on the cells specified in the **-cells** option to connect to the net specified in the **-power_net** option.

-ground_pin pin_name

Specifies the name of the ground pin on the cells specified in the **-cells** option to connect to the net specified in the **-ground_net** option.

-cells cells

Specifies the cells to connect to the power and ground nets specified by the **-power_net** and **-ground_net** options, respectively.

-create_ports none | top | all

Creates ports for the nets specified by the **-power_net** and **-ground_net** options.

Valid values for this option are

* **none** (the default)

Does not create any ports.

* **top**

Creates only a top-level port.

* **all**

Creates a top-level port and ports as needed on child cells.

-create_nets

Creates power and ground nets based on the power domain definitions and supply nets. If multiple supply nets are connected through supply ports, only one power net is created with the name following the supply net inside the highest logic hierarchy.

This option is valid only in automatic mode. It is ignored if either **-power_net** or **-ground_net** is used.

-all

Connect both power and ground pins and tie-off pins.

The **-all** option is mutually exclusive with the **-tie** option; you can specify only one. If you do not specify either option, **derive_pg_connection** connects only the power and ground pins.

This option is valid only in automatic mode and has no effect on tie-off connections. It is ignored if either **-power_net** or **-ground_net** is used.

-reconnect

Reconnects power and ground pins that have existing connections.

By default, the command works only on unconnected power and ground pins.

When you specify this option, the command works on all power and ground pins

of the design, whether or not they have existing connections. This option is valid only in automatic mode and has no effect on tie-off connections and physical only cells. It is ignored if either **-power_net** or **-ground_net** is used.

-verbose

Prints the details of the changes to the power and ground or tie-off connections.

DESCRIPTION

This command creates the power and ground network for your design by connecting the power and ground pins and the tie-off pins to the power and ground nets in your design.

The command supports two modes: manual mode and automatic mode. In manual mode, you use the **-power_net** and **-ground_net** options to specify the power and ground nets. Using at least one of these options invokes the tool in manual mode. You can use manual mode for single-voltage or multivoltage designs.

In automatic mode, the tool derives the power and ground nets from the power domain connections. At a minimum, each power domain has a primary power net and a primary ground net. A power domain might also have backup power and ground nets. You can use automatic mode only for UPF multivoltage designs. In addition, automatic mode requires logic libraries with power and ground pins. In automatic mode, the tool can create the power and ground nets, if they do not already exist.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses **derive_pg_connection** in automatic mode to connect all unconnected power and ground pins based on the power domain connections.

```
prompt> derive_pg_connection
```

Power/Ground Connection Summary:

P/G net name	P/G pin count(previous/current)
Power net "VDD":	0/2547
Power net "VDDM":	0/4798
Power net "VDDI":	200/250
Unconnected power pins:	7395/0
Ground net "VSS":	6368/7368
Unconnected ground pins:	1000/0

Information: connections of 8395 power/ground pin(s) are created or changed.

The following example uses **derive_pg_connection** in manual mode to connect all unconnected power pins to VDD and and unconnected ground pins to VSS.

```
prompt> derive_pg_connection -power_net VSS -ground_net VDD
```

SEE ALSO

```
check_mv_design(2)  
connect_power_domain(2)  
connect_power_net_info(2)  
connect_supply_net(2)  
create_power_domain(2)  
create_power_net_info(2)  
create_supply_net(2)  
create_supply_port(2)
```

detect_flcc_hotspot

Performs lithography hot spot detection. FLCC stands for Fast Lithography Compliance Check.

SYNTAX

```
detect_flcc_hotspot  
[-cell_name
```

Data Types

cName string

ARGUMENTS

```
-cell_name cName  
    Specifies the cell name in which the detected flcc hotspots info will be  
    stored. It is a Milkyway CEL view cell and is used exclusively to store flcc  
    hotspot info, similar to drc error cell. User can use ICC error browser to  
    open it and the flcc hotspots will be classified according to its categories.  
    If this param is not specified, the flcc hotspot info will be stored in  
    original top cell. If user opens the top cell with error browser, user still  
    can view these flcc hotspots, but no category classification is supported.
```

DESCRIPTION

The command `detect_flcc_hotspot` requires two files to execute, `hotSpots.ev` and `hotSpots.util`. User should have these two files in the working directory before running this command.

If flcc errors are not stored in original top cell, it will facilitate user to browse and compare the flcc errors before and after fixing. The param `-cell_name` is used to store flcc errors. The error cell is a Milkyway CEL view cell, and only contains error types, error objects and related descriptions.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command performs `detect_flcc_hotspot`.

```
prompt> detect_flcc_hotspot -cell_name A
```

SEE ALSO

`extract_flcc_hotspot(2)`

`detect_flcc_hotspot`
680

```
fix_flcc_hotspot(2)
```

detect_lcc_hotspot

Performs full-chip lithography compliance check (LCC) hotspot detection.

SYNTAX

```
status detect_lcc_hotspot  
-lcc_file_path lcc_path_name  
-layers list_of_layers  
[-dp_hosts list_of_dp_hosts]
```

Data Types

<i>lcc_path_name</i>	string
<i>list_of_layers</i>	list
<i>list_of_dp_hosts</i>	list

ARGUMENTS

-lcc_file_path *lcc_path_name*
Specifies a directory of LCC files, which contains necessary runsets needed by full-chip LCC detection jobs. The *lcc_path_name* value must be a full path.

-layers *list_of_layers*
Specifies the layers to run full-chip LCC detection.

-dp_hosts *list_of_dp_hosts*
Specifies the machines to be used. This option is required, unless you have a .dprc file under the working directory or you use the **-lcc_file_path** option, which also specifies the machines to be used. The value of the **-dp_hosts** option overrides the presence of a .dprc file.

DESCRIPTION

This command performs full-chip LCC detection. It outputs two fix guidance files under the working directory. These two files contains hotspot information and are necessary to run the **fix_lcc_hotspot** command. Full-chip LCC detection has a long run time, and using distributed processing is a requirement.

You should specify the machines to be used by the **-dp_hosts** option, or the .dprc file under the working directory, or the .dprc file specified under the *lcc_path_name* directory. If you do not do this, the command will not run and will not report an error.

The distributed processing engine used by the **detect_lcc_hotspot** command is different from that used by the **fix_lcc_hotspot** command and many other routing commands. You do not need to run the **set_distributed_route** command before running the distributed processing **detect_lcc_hotspot** command. If you run **set_distributed_route** before attempting to run **detect_lcc_hotspot**, you should run the **remove_distributed_route** command to release network resources; otherwise, the communication port between machines might become occupied and block distributed processing functionality.

An error cell named "<top_cell_name>_lcchs.err" will be created after running the **detect_lcc_hotspot** command. You can open this error cell to browse the hotspots in the current design.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows the command running on multiple layers and multiple Linux hosts.

```
prompt> detect_lcc_hotspot\  
-lcc_file_path /LCC/full-chip\  
-layers {M2 M3 M4}\  
-dp_hosts {lin1 lin2 lin3 lin4 lin5 lin6 lin7 lin8}
```

SEE ALSO

```
fix_lcc_hotspot(2)  
report_lcc_hotspot(2)
```

disconnect_net

Disconnects a net from pins or ports.

SYNTAX

```
status disconnect_net  
net  
object_list | -all
```

Data Types

<i>net</i>	string
<i>object_list</i>	list

ARGUMENTS

- net*
- Specifies the net name or net instance name to disconnect. A net must exist in the current design.
- object_list*
- Specifies the pins and ports disconnected from the net. Only pins and ports existing in the current design are specified. Either *object_list* or **-all** must be specified.
- all**
- Specifies to break all connections on the net. Either **-all** or *object_list* must be specified.

DESCRIPTION

The **disconnect_net** command breaks the connections between a net or a net instance and its pins or ports. The net, pins, and ports are not removed.

This command accepts only scalar (single bit) nets, but not bused nets.

`disconnect_net` can be used to disconnect PG or Tie nets from connected pins also.

To connect nets, use the **connect_net** command. To display the pins and ports connected to a net, use the **all_connected** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following examples disconnect nets using the **disconnect_net** command:

```
disconnect_net  
684
```

```
prompt> disconnect_net NET0 [get_ports A1]  
Disconnecting net 'NET0' from port 'A1'.
```

```
prompt> disconnect_net NET0 [get_pins U1/A]  
Disconnecting net 'NET0' from pin 'U1/A'.
```

The following example breaks all connections on a net using the **disconnect_net** command:

```
prompt> disconnect_net MY_NET_1 -all  
Disconnecting net 'MY_NET_1' from port 'PORT1'.
```

```
prompt> all_connected [get_nets MY_NET_1]
```

SEE ALSO

```
all_connected(2)  
connect_net(2)  
create_net(2)  
current_design(2)  
remove_net(2)
```

disconnect_power_net_info

Removes the exceptional power net information hookups with the power pin.

SYNTAX

```
status disconnect_power_net_info  
object_list  
-power_pin_name power_pin_name
```

Data Types

<i>object_list</i>	list
<i>power_pin_name</i>	string

ARGUMENTS

object_list
Specifies a list of leaf cells for which the exceptional power net information connections are being removed.

-power_pin_name *power_pin_name*
Specifies the name of the power pin information.

DESCRIPTION

The **disconnect_power_net_info** command removes exceptional power connections for a specific power pin information of a leaf cell. After an exceptional power connection is removed, it inherits the domain-wise power hookup from the domain to which it belongs.

See the **report_power_pin_info** man page for more information.

EXAMPLES

The following example makes and then removes an exceptional power hookup:

```
prompt> connect_power_net_info PD0_INST/I0 \  
        -power_pin_name PWR -power_net_name exp_VDD  
1  
prompt> disconnect_power_net_info PD0_INST/I0 -power_pin_name PWR  
1
```

SEE ALSO

```
connect_power_net_info(2)  
report_power_pin_info(2)
```


display_flip_chip_route_flylines

Displays flylines of flip-chip nets in the layout view of the IC Compiler GUI.

SYNTAX

```
status display_flip_chip_route_flylines  
[-nets | -nets_in_file nets_file]  
[-open_nets [-output_open_nets open_net_file]]
```

Data Types

<i>collection_of_nets</i>	collection
<i>nets_file</i>	string
<i>open_net_file</i>	string

DATA TYPES

<i>collection_of_nets</i>	collection
<i>nets_file</i>	string
<i>open_net_file</i>	string

ARGUMENTS

-nets *{collection_of_nets}*
Specifies the nets to be displayed. By default, all flip-chip nets are displayed. This option and **-nets_in_file** are mutually exclusive; you can specify only one.

-nets_in_file *nets_file*
Specifies the name of a file that contains the nets to be displayed. By default, all flip-chip nets are displayed. This option and **-nets** are mutually exclusive; you can specify only one.

-open_nets
Display the open net flylines of the specified nets and write the open nets to the file specified in **-output_open_nets**. By default, the open net flylines are not displayed.

-output_open_nets *open_net_file*
Specifies the name of the output file for the open net listing. This option is valid only when **-open_nets** is used. By default, the file is named `.trUnrouted`.

DESCRIPTION

This command displays the flylines of the specified flip-chip nets in the layout window of the IC Compiler GUI. This command requires that the GUI is active.

EXAMPLES

The following example displays all open flip-chip nets in the layout view of the IC Compiler GUI.

```
prompt> display_flip_chip_route_flylines -open_nets
prompt> display_flip_chip_route_flylines -open_nets \
        -output_open_nets RDLopen
```

SEE ALSO

```
set_route_flip_chip_options(2)
route_flip_chip(2)
create_stack_via_on_pad_pin(2)
write_flip_chip_nets(2)
push_flip_chip_route(2)
remove_flip_chip_route(2)
```

distance

Describes basic command types for describing mask geometry.

SYNTAX

None.

ARGUMENTS

None.

DESCRIPTION

This command describes basic types of distance for describing mask geometry.

A *distance* is a basic command syntax unit, like strings, and floating-point numbers. In ICC, a distance must be a fixed-point number. Thus 7, 142.65 and 100000.000 are all distances, but 3.7e6 is not.

Distances are returned by some commands, and some attributes are distances. Distances can be used as arguments to some commands.

Distances can be grouped into lists to form points. A *point* is a Tcl list of two distances, which are interpreted as the X and Y values of the coordinate. For example, the origin is {0 0}.

A *rectangle* is a Tcl list of two points, which are interpreted as the lower-left and upper-right corners. For example, the unit square is { {0 0} {1 1} }.

To access a distance when given a point, you might want to use the lindex function.

EXAMPLES

The following examples show how to set several uses of distance.

```
prompt> set spacing 12.25

prompt> set lower_left_x 2.5
prompt> set lower_left_y 44.125

prompt> set corner {22.54 1235.75}

prompt> set base [list $lower_left_x $lower_left_y]

prompt> set box [list $base [list 222.14 [expr 200.2 * 7]]]

prompt> get_attribute $cell11 bbox
prompt> {241.5 700.1} {246.4 703.1}
```

SEE ALSO

`move_object(2)`

distribute_objects

Distributes one or more objects.

SYNTAX

```
status distribute_objects
[-anchor object]
[-parent]
[-from value_point_rect]
[-to value_point_rect]
[-side {left | right | top | bottom | hcenter | vcenter}]
[-spread]
[-vertical]
[-offset real]
[-wiretrack_offset int]
[-resize]
[-keep_area]
[-ignore_fixed]
objects
```

Data Types

<i>object</i>	collection
<i>value_point_rect</i>	int, point, or rectangle
<i>objects</i>	collection

ARGUMENTS

-anchor *object*
Specifies the first object that other objects are distributed from.
The argument should consist of a collection of one object.

-parent
Specifies that objects are distributed starting at the parent edge.

-from *value_point_rect*
Specifies that objects are distributed starting at the specified position.
The position can be a single value (x for horizontal, y for vertical), a point, or a rectangle.

-to *value_point_rect*
Specifies that the objects are distributed between two positions. The first specified by the **-from** argument and the second specified by this argument.
The position can be a single value (x for horizontal, y for vertical), a point, or a rectangle.

-side {left | right | top | bottom | hcenter | vcenter}
Specifies the side of the anchor object that the objects are distributed from.
The default depends on the aspect ratio (height/width) of the objects' surrounding bbox. If the aspect ratio is less than or equal to 1 then left is the default otherwise bottom is the default.
left , 1 : Left Side

right , r : Right Side
top , t : Top Side
bottom , b : Bottom Side
hcenter, hc, ch : Horizontal Center
vcenter, vc, cv : Vertical Center
The default is **left**.

-spread

Specifies that the specified objects should be spread between the leftmost and rightmost specified fixed objects. If no fixed objects are specified the leftmost and rightmost objects are used.

-vertical

Specifies that the specified objects should be spread vertically rather than horizontally when using either the **-to** or **-from** options.

-offset *real*

Specifies the spacing between each distributed object.
This option is ignored if used together with **-wiretrack_offset** (see below) and all distributed objects are either pins or terminals.
The default is **0.0**.

-wiretrack_offset *int*

Specifies the spacing between each distributed object in terms of number of wire tracks. This option is valid only when the objects to be distributed are all pins or terminals. It is otherwise ignored.
For example, **-wiretrack_offset 2** means that there should be two wiretracks between adjacent pins (or terminals).
The default value is 1. This default value comes into effect if the objects to be distributed are all pins or terminals and inter-object distance is not otherwise specified or implied. For example, **distribute_objects -side top \$select_pins** results in pins in the collection "select_pins" getting placed on every other wiretrack. However, the default value is not applicable for **distribute_objects -spread \$select_pins** because option **-spread** implies that the pins are to be equally spaced between the first and the last pins.

-resize

Whether to resize the object when distributing.

-keep_area

Whether to keep the original object's area when resizing.

-ignore_fixed

Normally fixed objects will not be distributed. If this flag is supplied then fixed objects will also be distributed.

objects

List of objects to distribute.

DESCRIPTION

Distribute a list of unfixed objects so that they are placed side by side horizontally or vertically with an optional spacing between each object.

The user can specify an anchor object or anchor position, or let the command automatically determine the anchor object from the supplied list of objects using the following algorithm:

- If any objects are marked as fixed then the anchor objects is selected from the set of fixed objects. The rightmost, leftmost, bottommost or topmost fixed object for alignment of left, right, top and bottom respectively.
- If no objects are marked as fixed then the anchor object is taken as the leftmost, rightmost, topmost or bottommost object for alignment of left, right, top and bottom respectively.

The command essentially has two modes:

- &. Distribute - If 'to position' is not specified
- &. Spread - If 'to position' is specified then the objects

Distribute Mode

After each object is moved or resized to be next to the anchor object, the anchor object is updated to be the distributed object. The next object in the sequence is then selected to be the leftmost, rightmost, topmost or bottommost unfixed object for distribute left, right, top and bottom respectively.

When distributing an object the position of the opposite edge can be retained so that the object will be resized instead of moved. The position of the other sides of the object will also be retained unless the user specifies that the area is to be kept in which case the other sides are moved to achieve the original object area.

Spread Mode

The anchor object or anchor position and to position mark a range over which the objects are spread. The objects are then evenly spaced (moved) over this range or evenly resized to fill this range (with any specified offset between them).

The existing left->right or bottom->top order specifies the order in which the objects are placed.

When -resize is not specified any movable object can be aligned. If -resize is specified then only resizable objects can be aligned. See `get_edit_property(2)` man page for details on which objects can be moved and resized.

The spread direction is by default horizontal. Using the **-vertical** option changes it to vertical.

NOTES

Snapping is done automatically using global snap settings.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example moves the currently selected objects so that they are placed side by side in the horizontal direction with 20 units between each.

```
> distribute_objects [get_selection] -parent -offset 20
```

The following example resizes the currently selected objects so that they are an even sized between 0 and 800.

```
> distribute_objects [get_selection] -from 0 -to 800 -resize -vertical
```

SEE ALSO

```
move_objects(2)  
remove_objects(2)  
resize_objects(2)  
rotate_objects(2)  
align_objects(2)  
expand_objects(2)  
flip_objects(2)  
set_object_snap_type(2)  
get_object_snap_type(2)
```


drive_of

Returns the drive resistance value of the specified library cell pin.

SYNTAX

```
float drive_of  
library_cell_pin  
[-rise | -fall]  
[-piece best | worst | average average_value]  
| [-min]
```

Data Types

library_cell_pin string

ARGUMENTS

library_cell_pin
Specifies the name of the library cell pin whose drive value is to be returned. Use the following format to specify this option: *library_name/lib_cell_name/pin_name*. The specified library must already be loaded into the shell.

-rise
Specifies that the command is to return the *library_cell_pin* rise drive resistance.

-fall
Specifies that the command is to return the *library_cell_pin* fall drive resistance.

-piece best | worst | average *average_value*
Specifies to return the **best**, **worst**, or *average_value* of all of the pieces. The *average_value* corresponds to the *average_value* integer index value. This option can be used only with piecewise-linear libraries.

-min
Gets the wire's drive value.

DESCRIPTION

This command returns the drive or wire_drive resistance of the specified library cell pin. The resistance is primarily used as an argument to the **set_drive** command. The **set_driving_cell** command provides a more convenient and accurate method of describing the drive capability of a port.

If you do not use either the **-rise** or **-fall** option, the command returns the largest of the rise or fall drive resistances.

If the command cannot find the specified *library_cell_pin*, it issues an error message and returns 0.0.

Because drive resistances are associated with timing arcs, it is possible to have more than one distinct rise and fall drive associated with a specified library pin. In this case, the command returns the largest resistance.

If the specified library uses the nonlinear (table-lookup) delay model, the command returns an estimated linear drive value. The **set_driving_cell** command handles nonlinear drives and produces more accurate results than the **set_drive** command for these libraries.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command returns the rise drive of the *my_pin* pin of the *lib_cell* cell from the *tech_lib* library.

```
prompt> drive_of -rise tech_lib/lib_cell/pin
```

The following command searches through all of the pieces of the **rise_pin_resistance** attribute on the *my_pin* pin of the *lib_cell* cell. The command returns the worst value for all of the pieces. This assumes that the *tech_lib* library is modeled as a piecewise-linear library; otherwise, the command returns the normal *rise_resistance*.

```
prompt> drive_of -piece "worst"\  
-rise tech_lib/lib_cell/pin
```

SEE ALSO

```
set_drive(2)  
set_driving_cell(2)
```

echo

Echos arguments to standard output.

SYNTAX

```
string echo  
[-n] [argument...]
```

ARGUMENTS

-n
By default, **echo** adds a newline. This option suppresses it.

DESCRIPTION

Prints out the value of the given arguments. Each of the arguments are separated by a space. The line is normally terminated with a newline, but if the **-n** option is specified, multiple **echos** are printed onto the same output line.

The **echo** command is used to print out the value of variables and expressions in addition to text strings. The output from **echo** can be redirected using the **>** and **>>** operators.

EXAMPLES

```
prompt> echo "Running version" $sh_product_version  
Running version v3.0a
```

```
prompt> echo -n "Printing to" [expr (3 - 2)] > foo  
prompt> echo " line." >> foo  
prompt> sh cat foo  
Printing to 1 line.
```

SEE ALSO

eco_netlist

Performs engineering change order (ECO) operations on the current Milkyway cell.

SYNTAX

```
status eco_netlist  
[-by_verilog_file verilog_filename]  
[-physical]  
[-compare_pg]  
[-freeze_silicon]  
[-write_changes write_changes_file_name]  
[-cells hier_cell_instance_names]
```

Data Types

<i>verilog_filename</i>	string
<i>hier_cell_instance_names</i>	list

ARGUMENTS

-by_verilog_file *verilog_filename*
Specifies the Verilog source file name for the ECO change.

-physical
Enables special handling of the physical-only cell instances and their existing connections.
When you use this option, if the Verilog netlist contains a physical-only cell instance and the working design does not, ECO adds it into the design. However, if the working design contains a physical-only cell instance that is not present in the Verilog netlist, ECO discards it from the design. By default, ECO updates only the logical netlist objects according to the input golden Verilog file and ignores any differences in physical-only cell instances. If the Verilog netlist contains a physical-only cell instance and the working design does not, ECO does *not* add it into the design. However, if the working design contains a physical-only cell instance that is not presented in the Verilog netlist, the ECO keeps it in the design.

-compare_pg
Compares power and ground (PG) related nets and connections. When the option is enabled, the command updates PG data, such as signal data, in the working design according to the golden Verilog file. Thus, the following conditions occur:

- If the working design contains a PG network and the golden Verilog file does not, ECO removes the PG network from the working design.
- If the working design does not contain a PG network and the golden Verilog file does, ECO adds the new PG network to the working design.
- If both the working design and the golden Verilog file contain a PG network, ECO treats the Verilog PG network as golden and performs PG changes in the working design according to the Verilog PG network.

By default, ECO ignores the PG differences between the working design and the golden Verilog file. No changes are made to the PG network in the working design.

-freeze_silicon

Turns on the ECO freeze silicon mode. In the freeze silicon mode, ECO behaves in the following way:

- When removing a cell instance, **eco_netlist** does not actually remove the cell from the design. The cell instance name stays the same as it is. The cell is marked as a spare cell.

The PG connections on PG pins are kept as they are, disregarding the **-compare_pg** option. All signal pins are disconnected. The input signal pins are tied to the direct tie-low net while the output signal pins are left floating.

- When adding a cell instance, **eco_netlist** checks if there is a spare cell with the same name in the design. If yes, ECO renames the spare cell, then continues to add the new cell.

The convention used for renaming cells is SPARE_OriginalCellName. If the name still conflicts with another cell, SPARE_OriginalCellName_# will be used, where # is a number starting from zero.

- When replacing or resizing a cell instance, ECO first removes the old cell, then adds the new cell. ECO behaves exactly the same when removing and adding cell instances.

Before applying any changes, ECO first checks the feasibility of the spare cells for all cell-adding operations. This checking is to make sure each of the newly added cells has enough and proper space to be placed. One cell-adding operation is verified to be feasible only if the space is found. Once ECO detects there is not enough or proper space to place a newly added cell, it will output a warning message.

-write_changes *write_changes_file_name*

Generates a Tcl script file that lists all the changes the **eco_netlist** command does.

With the Tcl-based netlist editing commands in the output Tcl file, you can repeat the changes on the original working cell and create a new cell. This new cell is exactly the same as the cell created by the **eco_netlist** command. The **eco_netlist** command supports generating Tcl script files under both unconstrained mode and freeze silicon mode.

-cells *hier_cell_instance_names*

Specify the target hierarchical cell instances on which **eco_netlist** will work. Note that you should provide the full hierarchical cell instance name, not the module name.

Only the cell instances specified by **cells** and their sub-instances will be compared and changed.

The **-cells** option does not support the name of the top instance name as input. If you want to do ECOs on the top cell instance, run **eco_netlist** without the **-cells** option.

DESCRIPTION

Compares the current design netlist with the golden Verilog file, and then changes the current design based on the golden Verilog file. A cell must be open before using this command. The changed cell is not saved automatically.

The comparison and changes are only applied on the current design, the softmacros instantiated in the current design are not supported.

Adding and removing hierarchical cell instances are not allowed. This means the hierarchy of the Verilog must be exactly the same as it is in the Milkyway database.

For details about using this command in the IC Compiler ECO flow, see *IC Compiler Implementation User Guide*, which is available on SolvNet.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

Gate Array Cell Support

Before applying any changes, ECO first runs a feasibility check on gate array cells before implementing any changes. This is to make sure all gate array eco_add cells, which are being added directly or generated by replacing or resizing operation, have enough space to be placed.

For more information, see *IC Compiler Implementation User Guide*, which is available on SolvNet.

EXAMPLES

The following example shows the most common use of this command.

```
prompt> open_mw_cel working
prompt> eco_netlist -by_verilog_file golden.v \
-write_changes fileName.tcl

...
[ECO] ADD_CELL_INSTANCE 'c4_celi_add_SIG1' (AND2T FRAM).
[ECO] ADD_PORT 'min1' (input) on cell instance 'c12_hiport_rm_SIG'
[ECO] ADD_NET 'min1' into 'c12_hiport_rm_SIG'
[ECO] CONNECT net 'min1' in cell instance 'c12_hiport_rm_SIG' with port 'min1'.
[ECO] CONNECT net 'min1' in cell instance 'c12_hiport_rm_SIG' with port instance 'c
12_leaf_celi/A'.
[ECO] DISCONNECT net 'min1' in cell instance 'c10_hiport_rm_SIG' from port instance
'c10_leaf_celi/A'.
[ECO] DISCONNECT net 'min1' in cell instance 'c10_hiport_rm_SIG' from port 'min1'.
[ECO] REMOVE_NET 'min1' from 'c10_hiport_rm_SIG'
[ECO] REMOVE_PORT 'min1' on cell instance 'c10_hiport_rm_SIG'.
...
```

The following examples show the use of the **-cells** option.

1. Specify instances by list:

```
prompt> eco_netlist -by_verilog_file golden.v \  
-cells {hier_path1/instance_name1, hier_path2/instance_name2}
```

2. Specify instances by collection:

```
prompt> eco_netlist -by_verilog_file golden.v \  
-cells [get_cells -hier *abc*]
```

3. Specify instances by list and collection:

```
prompt> eco_netlist -by_verilog_file golden.v \  
-cells *(lqhier_path1/instance_name1, hier_path2/instance_name2, \  
[get_cells -hier *abc*])*(rq
```

SEE ALSO

```
report_eco_history(2)  
place_freeze_silicon(2)  
route_eco(2)
```

end_fp_trace_mode

Removes the trace mode design from memory.

SYNTAX

```
status end_fp_trace_mode
```

DESCRIPTION

This command removes the trace mode design from memory.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLE

The following example shows a basic output of the **end_fp_trace_mode** command.

```
prompt> get_fp_trace_mode  
Design is in trace mode  
0  
prompt> end_fp_trace_mode  
prompt> get_fp_trace_mode  
Design is not in trace mode  
1
```

SEE ALSO

```
get_fp_trace_mode(2)  
set_fp_trace_mode(2)
```


error_info

Prints extended information on errors from last command.

SYNTAX

string **error_info**

DESCRIPTION

The **error_info** command is used to display information after an error has occurred. Tcl collects information showing the call stack of commands and procedures. When an error occurs, the **error_info** command can help you to focus on the exact line in a block which caused the error.

EXAMPLES

This example shows how **error_info** can be used to trace an error. The error is that the iterator variable "s" is not dereferenced in the 'if' statement. It should be '\$s == "a" '.

```
prompt> foreach s $my_list {?          if {s == "a"} {?          echo "Found 'a'!"
?}
?}
Error: syntax error in expression " s == "a" "
      Use error_info for more info. (CMD-013)
prompt> error_info
Extended error info:
syntax error in expression " s == "a" "
      while executing
      "if {s == a} {          echo "Found 'a'"
      }"
      ("foreach" body line 2)
      invoked from within
      "foreach s [list a b c] {    if {s == a} {          echo "Found 'a'"
      }
      }"
      -- End Extended Error Info
```

SEE ALSO

estimate_fp_area

Performs automated die size exploration for finding the smallest die size possible to decrease the cost per die. Can also be applied to blocks.

SYNTAX

```
status estimate_fp_area
[-min_height min_height]
[-max_height max_height]
[-min_width min_width]
[-max_width max_width]
[-acceptable_overflow percentage]
[-sizing_type fixed_width | fixed_height | fixed_aspect_ratio]
[-core_sizing_only]
[-keep_blockages]
[-increase_area area]
[-replace_io]
[-maintain_iopad_alignment]
[-power_net_names list_of_names]
[-ir_drop_value target_value]
[-run_preroute_script preroute_script_name]
[-run_pns_script script_name]
[-save_as name]
[-maintain_power_structure]
[-estimate_optimization]
[-shrink_only]
```

Data Types

<i>min_height</i>	float
<i>max_height</i>	float
<i>min_width</i>	float
<i>max_width</i>	float
<i>percentage</i>	float
<i>area</i>	float
<i>list_of_names</i>	list
<i>target_value</i>	float
<i>preroute_script_name</i>	string
<i>name</i>	string

ARGUMENTS

-min_height *min_height*
Specifies the lower bound of core height, in microns, required to stop this command. By default, no die sizes that result in utilizations greater than 95% are considered.

-max_height *max_height*
Specifies the upper bound of core height, in microns, required to stop this command. By default, no die sizes that result in utilizations lower than 30% are considered.

`-min_width min_width`
 Specifies the lower bound of core width, in microns, required to stop this command. By default, no die sizes that result in utilizations greater than 95% are considered.

`-max_width max_width`
 Specifies the upper bound of core width, in microns, required to stop this command. By default, no die sizes that result in utilizations lower than 30% are considered.

`-acceptable_overflow percentage]`
 Specifies the overall gcell overflow percentage in the range of (0.0 - 0.25) to determine whether a design is routable. By default, the tool determines routability by analyzing neighborhoods of individual gcells.

`-sizing_type fixed_width | fixed_height | fixed_aspect_ratio`
 Defines constraints on the search space of aspect ratios. By default, the tool allows the aspect ratio to vary.

`-core_sizing_only`
 Allows core shrink only. That is, it leaves the I/O pad ring untouched, as in the input floorplan. By default, the block or chip shrinks with the core and the I/O pad or pin placement is scaled.

`-keep_blockages`
 Keeps placement blockages in the input floorplan undeleted. By default, the tool deletes all blockages in the input design.

`-increase_area area`
 Provides the ability to reserve some extra space for cells to be added later in the flow. The area is specified in square microns. The default value is 0 (zero).

`-replace_io`
 Replaces all I/O pads and pins based on Top Design Format (TDF) constraints. If no TDF file exists, a TDF file is created with side and order constraints. By default, the command maintains the relative I/O placement of the input floorplan.

`-maintain_iopad_alignment`
 Aligns the I/O pads in different rings when multi-ring I/O structures are present. By default, the command separately scales each I/O ring.

`-power_net_names list_of_names`
 Invokes power network synthesis during die size exploration. The *pair_of_names* value can be a list of strings. Power and ground net names must be provided as a list of net names. By default, the command does not run power network synthesis.

`-ir_drop_value target_value`
 Specifies the target IR drop value. The default is 10% of VDD value, in mV, for power network synthesis.

`-run_preroute_script preroute_script_name`
 Pre-routes certain nets after power network synthesis (PNS) by using a script

file. By default, the command does not create pre-routes.

-run_pns_script *script_name*

Runs PNS by using an input script instead of the default PNS run. By default, the command does not do PNS unless you use either the **-run_pns_script** or **-power_nets** option.

-save_as *name*

Specifies a cell name in which to save the die size exploration result. By default, it is "DesignName"EstimateArea.

-maintain_power_structure

Maintains the power structure.

-estimate_optimization

Estimate the amount of area required by optimization and account for this during the search for the minimal die size. This option is recommended if the input netlist is not optimized.

DESCRIPTION

This command automates die size exploration. It explores the solution space of valid die areas to determine the smallest, routable die size. The command maintains relative placement of important elements in the floorplan. When applied to full chips, it maintains the relative placement of hard macros and the side, order, and relative placement of I/O pads. When applied to rectangular or rectilinear blocks, it maintains the relative placement of hard macros and the side, order, layer, and relative placement of pins. The relative locations of plan groups and voltage areas are maintained, and the plan groups and voltage areas are sized in proportion to the core area for both blocks and chips.

Multicorner-Multimode Support

Depending on the options used, this command either uses the active scenarios or has no dependency on scenario-specific information.

EXAMPLE

The following example performs die size exploration with all defaults.

```
prompt> estimate_fp_area
```

The following example performs PNS and PG grids generation in the **estimate_fp_area** process.

```
prompt> estimate_fp_area -power_net_names {VDD VSS}
```

The following example changes the die size only in the y-dimension.

estimate_fp_area

```
prompt> estimate_fp_area -sizing_type fixed_width
```

SEE ALSO

estimate_fp_black_boxes

Sets the size of a black box based on an estimation of the objects that it will contain when replaced with real logic.

SYNTAX

```
status estimate_fp_black_boxes
[[{-sm_size size
  | -polygon {polygon_area}]
 | -sm_gate_equiv gate_count}]
[-sm_util util]
[-hard_macros names]
[-fixed_shape]
[-reset_shape]
black_boxes
```

Data Types

<i>size</i>	list
<i>polygon_area</i>	list
<i>gate_count</i>	integer
<i>util</i>	float
<i>names</i>	string
<i>black_boxes</i>	collection

ARGUMENTS

-sm_size *size*

Estimates the area as a soft macro of the specified size. The estimated area is determined by multiplying width by height, given by a list of 2 floating point numbers. If you also use the **-sm_util** option, an equivalent gate count value is calculated internally based on the *util* value. Otherwise, the tool uses a value of 1.0 for utilization when *sm_size* is used, but *sm_util* is not used. The **-sm_size** and **-polygon** options are mutually exclusive: you can use only one. By default, this option is off.

-polygon {*polygon_area*}

Creates a rectilinear black box. Substitute a list of defining points of the polygon you want for *polygon_area*. A polygon is identified by specifying the coordinates of all its points. The format is {{x1 y1} {x2 y2} ... {x1,y1}}. Identify the points as absolute coordinates in microns. The **-sm_size** and **-polygon** options are mutually exclusive: you can use only one. When you use the **-polygon** option to estimate black box size, the *sm_util* variable defaults to 1.0. By default, this option is off.

-sm_gate_equiv *gate_count*}

Estimates the area as a soft macro with the specified number of equivalent gates. If you also use the **-sm_util** option, an equivalent gate count value is calculated internally based on the *util* value. Otherwise, the tool uses a value of 0.7 and a standard gate area of 2-input-nand gate to estimate the total area.

-sm_util *util*

Estimates the area as a soft macro of the specified utilization. To use this option, you must also use the **-sm_gate_equiv** option specifying a positive value for *gate_count*. The default value is 0.7 (except for when you also use the **-sm_size** or **-polygon** options).

-hard_macros *names*

Specifies a list of reference library hard macro names in a string with the names separated by spaces or commas. Each name can have an optional instance count. The instance count is specified as a number in parentheses () appended to the name. For example, 10 instances of the mem macro is specified as mem(10).

The size estimation includes the area of all the hard macros in the specified list. This estimation mode does not consider the shapes of the actual hard macros.

-fixed_shape

Specifies whether to mark the black box as fixed shape after estimation.

-reset_shape

Specifies whether to reset the shape to rectangular if it is currently rectilinear.

black_boxes

Specifies the black boxes to estimate.

DESCRIPTION

This command allows you to set the size of a black box from the estimation of the objects that it will contain when finally replaced with real logic.

The estimation can be as a soft or hard macro. As a soft macro it has one, but not both, of the following characteristics:

- a size, specified by using the **-sm_size** or **-polygon** option
- the number of gate-equivalent cells with utilization, specified by using the **-sm_gate_equiv** option.

You can specify utilization in any of the 3 cases: **-sm_size**, **-polygon**, or **-sm_gate_equiv**. As a Hard Macro it has either a specified size (**-sm_size**) or number of gate equivalent cells (**-sm_gate_equiv**).

If you do not use the **-hard_macros** option, then you must use the **-sm_size**, **-polygon**, or **-sm_gate_equiv** option. If you specify both **-sm_size** and **-sm_gate_equiv** together, then the *size* value takes precedence and the *gate_count* value is ignored; and, instead, becomes a dependent variable that is calculated based upon the *size* value. Similarly, you specify both the **-polygon** and **-sm_gate_equiv** options together, then the *polygon_area* value takes precedence and the *gate_count* value is ignored; and, instead, becomes a dependent variable that is calculated based upon the *polygon_area* value.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example estimates a black box named alu1 to a soft macro size of 100x100 with a utilization of 0.7.

```
prompt> estimate_fp_black_boxes\  
-sm_size {100 100} -sm_util 0.7\  
[get_cells -filter "is_physical_black_box==true" alu1]
```

The following example estimates a black box named alu1 to a rectilinear soft macro.

```
prompt> estimate_fp_black_boxes\  
-polygon {{1723.645 1925.365} {1723.645 1722.415}\  
{1803.595 1722.415} {1803.595 1530.535}\  
{799.915 1530.535} {799.915 1925.365}\  
{1723.645 1925.365}}\  
[get_cells -filter "is_physical_black_box==true"\  
I_ORCA_TOP/I_PCI_TOP]\  
[get_cells -filter "is_physical_black_box==true" alu1]
```

SEE ALSO

```
get_cells(2)  
import_fp_black_boxes(2)
```


estimate_rc

Estimates RC coefficients based on annotated data.

SYNTAX

```
status estimate_rc
[-bound max_float_capacitance]
[-min]
[-net net_list]
[-nworst_nets percentage]
[-threshold min_float_capacitance]
```

Data Types

<i>max_float_capacitance</i>	float
<i>net_list</i>	list
<i>percentage</i>	integer
<i>min_float_capacitance</i>	float

ARGUMENTS

-bound *max_float_capacitance*
Specifies the maximum back-annotated capacitance below which nets are to be considered for estimation.

-min
Specifies to use the minimum conditions. By default, the command uses maximum conditions.

-net *net_list*
Estimates only for those nets on *net_list*. If you also specify either the **-threshold** or **bound** option, only those nets that meet the criteria in the current instance are used. When you use both the **-net** and **-nworst_nets** options, the command ignores the **-nworst_nets** option.

-nworst_nets *percentage*
Specifies the percentage of nets, based on timing criticality, to select for estimation of RC coefficients. If it is specified out of range (0 to 100), all nets that meet the threshold and bound criteria in the current instance are used. The default is 100 percent. When you use both the **-net** and **-nworst_nets** options, the command ignores the **-nworst_nets** option.

-threshold *min_float_capacitance*
Specifies the minimum back-annotated capacitance above which nets are to be considered for the estimation.

DESCRIPTION

This command uses back-annotated data for net loads and pin-to-pin delays and computes a set of RC coefficients that would best fit the provided data. These coefficients, if used to compute the net loads and pin-to-pin delays, minimize the

error between the annotated and computed values.

The Standard Delay Format (SDF) files and **set_load** files must be sourced before running this command. The command prints out the suggested values for RC coefficients on the standard output. If you use these values, use the **set_delay_estimation_options** command to set them before performing optimization.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example runs the **estimate_rc** command.

```
prompt> estimate_rc -nworst_nets 10  
Loading design 'test_design'  
Capacitance - horizontal 0.0001696 vertical : 5.845e-06  
Resistance - horizontal 1.692e-07 vertical : 9.987e-08
```

SEE ALSO

```
read_sdf(2)  
report_delay_estimation_options(2)  
set_delay_estimation_options(2)  
set_load(2)
```

exit

Terminates the application.

SYNTAX

```
string exit  
[exit_code]
```

Data Types

```
exit_code      int
```

ARGUMENTS

```
exit_code  
    Return code to the operating system. Default is 0.
```

DESCRIPTION

This command exits from the application. You have the option to specify a code to return to the operating system.

EXAMPLES

The following example exits the current session and returns the code 5 to the operating system. At a UNIX operating system prompt, verify the return code as shown.

```
prompt> exit 5
```

```
5
```

SEE ALSO

```
quit(2)
```

expand_flip_chip_cell_locations

Pushes flip-chip bump or driver cells away proportionally or pulls them closer according to a specified ratio.

SYNTAX

```
status expand_flip_chip_cell_locations
-flip_chip_cells cell_list
-expand_ratio ratio
```

Data Types

<i>cell_list</i>	collection
<i>ratio</i>	float

ARGUMENTS

```
-flip_chip_cells cell_list
    Specifies the flip chip bump or driver cells to push or pull. You must specify
    at least two cells because this command changes their relative positions.

-expand_ratio ratio
    Specifies a ratio by which the selected flip chip bump or driver cells are
    to be expanded. If the given ratio is less than 1.0, cells are pulled closer.
    The ratio should be somewhere between 0.01 and 100.
```

DESCRIPTION

This command proportionally expands or shrinks the relative locations of given flip chip bump or driver cells. You must specify at least two cells so that the relative locations can be expanded or shrunk.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example expands the locations of the flip chip bumps by a factor of 1.2.

```
prompt> expand_flip_chip_cell_locations\
-flip_chip_cells\
{BUMP_ring D1 BUMP_ring D2 BUMP_ring D3 \
BUMP_ring D4 BUMP_ring D5 BUMP_ring D6}
<bold-expand_ratio 1.2
```

SEE ALSO

`place_flip_chip_array(2)`
`place_flip_chip_ring(2)`

expand_objects

Expands one or more objects by moving each side out until it hits another object or the core boundary (number of sides retained) or make object fill available space around it (number of sides is not retained)

SYNTAX

```
status expand_objects
[-side {left | right | top | bottom | all}]
[-fill]
[-offset real]
[-hit_types string_list]
[-ignore_fixed]
objects
```

ARGUMENTS

-side {left | right | top | bottom | all}
Specifies the side(s) of the object that are expanded:
left Left Side
right Right Side
top Top Side
bottom Bottom Side
all All Sides
Note: if the user wishes to expand more than one side but not all sides then multiple calls are required.
The default is **all**.

-fill
Specifies that instead of moving each side outwards until it hits one of the specified object types (no fill) the shape is expanded to fill the space available around it which may result in new sides being created.

-offset *real*
Specifies the spacing between the side(s) and the surrounding objects.
Note: this value can be negative as well as positive
The default is **0.0**.

-hit_types *string_list*
The types of objects considered for collisions.
Valid types are:
macro Hard Macro or Soft Macro
soft_macro Soft Macro
hard_macro Hard Macro
movebound Move Bound
plan_group Plan Group
blockage Blockage
polygon Polygon
(Default: macro, plan_group, blockage and polygon)

-ignore_fixed
Normally fixed objects will not be expanded. If this flag is supplied then

fixed objects will also be expanded.

objects

List of objects to expand

DESCRIPTION

Expands an object's sides until they hit another object or the core boundary.

The expansion is done either by moving each specified object side outwards until it hits one the specified object types (no fill) or the object fills the available space (fill) and new sides are created to match the shape of the available space.

If an offset is specified then the resultant shape's selected sides are shrunk (or grown in the case of negative offsets) by this offset after expansion.

Only resizable objects can be expanded. In the case where fill is used and the resultant shape is rectilinear only objects supporting rectilinear boundaries will be affected.

See `get_edit_property(2)` man page for a list of objects which can be resized or support rectilinear boundary.

NOTES

Specified objects must be completely inside the core area.

If the operation results in an invalid object boundary (self overlapping) the command will fail

Snapping is done automatically using global snap settings.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example resizes the alu1 cell so that it's bounding box is 20 units from any of the collision objects in all directions.

```
> expand_objects alu1 -side all -offset 10
```

SEE ALSO

`move_objects(2)`
`remove_objects(2)`
`resize_objects(2)`
`rotate_objects(2)`
`align_objects(2)`
`distribute_objects(2)`

```
flip_objects(2)  
set_object_snap_type(2)  
get_object_snap_type(2)
```


explore_power_switch

Explores and estimates the placement of MTCMOS header or footer cells based on IR drop.

SYNTAX

```
status explore_power_switch
-lib_cells lib_cells_or_power_switches
-real_pg_net real_pg_net_name
-virtual_pg_net virtual_pg_net_name
[-header]
[-bounding_box bounding_box]
[-x_increment {x_inc_list}]
[-y_increment {y_inc_list}]
[-orientation {N | W | S | E | FN | FE | FS | FW}]
[-voltage_area voltage_area]
[-no_placement]
[-legalize_placement]
[-no_pns]
[-run_pns_script file_name]
[-design hierarchy_name]
[-save_placement prefix]
[-preroute_mode_real_pg_port {rail | tie | net}]
[-no_preroute_real_pg_port]
[-preroute_mode_virtual_pg_port {rail | tie | net}]
[-no_preroute_virtual_pg_port]
```

Data Types

<i>lib_cells_or_power_switches</i>	collection
<i>real_pg_net_name</i>	collection
<i>virtual_pg_net_name</i>	collection
<i>bounding_box</i>	rectangle
<i>x_inc_list</i>	list
<i>y_inc_list</i>	list
<i>voltage_area</i>	collection
<i>file_name</i>	string
<i>hierarchy_name</i>	string
<i>prefix</i>	string

ARGUMENTS

-lib_cells *lib_cells_or_power_switches*
Specifies the header or footer multithreshold CMOS (MTCMOS) cells to be used for exploration. Use power switches for Unified Power Format (UPF) mode, and use lib cells for non-UPF mode. The *lib_cells_or_power_switches* argument can be a collection or list.

-real_pg_net *real_pg_net_name*
Specifies a real power or ground net on which to perform power network analysis. Real power nets connect to the real power pins of the MTCMOS cells. The real power net is not powered down. The *real_pg_net_name* argument can be

a collection or list of one item.

Under UPF mode, this option is optional. The default is to get the input supply net connectivity from the input port of the power switches.

-virtual_pg_net *virtual_pg_net_name*

Specifies a virtual power or ground net on which to perform IR drop analysis. Virtual power nets connect to the virtual power pins of the MTCMOS cells. The virtual power net can be powered down by the MTCMOS header or footer cells to reduce leakage power. The *virtual_pg_net_name* argument can be a collection or list of one item.

Under UPF mode, this option is optional. The default is to get the output supply net connectivity from the output port of the power switches.

-header

Places and analyzes header cells (typically PMOS transistors) that connect to power nets. The command cannot insert header and footer cells simultaneously. For header cells, specify the power (VDD) nets to be analyzed. For footer cells, specify the ground nets to be analyzed.

By default, the tool analyzes the footer cells (typically NMOS transistors) that connect to ground nets. Under UPF mode, the default is to get the power and ground type from the supply net of power switches; getting the header for power and footer for ground.

-bounding_box *bounding_box*

Defines the bounding box (in microns) of the area in which to place the header or footer cells.

If you specify both the **-voltage_area** and **-bounding_box** options, header/footer cells are added to the intersecting region of the given bounding box and voltage area. If you do not use the **-voltage_area** option, the specified bounding box coordinates are relative to the origin of the design.

If you use the **-bounding_box** option and also use the **set_mtcmos_pna_strategy** command with its **-relative_to_voltage_area** option, the specified bounding box coordinates are relative to the lower-left corner of the voltage area's bounding box. The default *bounding_box* is the core area.

-x_increment {*x_inc_list*}

Specifies a distance as a list of x-increments in microns. The values in the list are used as x-direction spacings between two header or footer cells during placement exploration. By default, the command inserts two header or footer cells, one at each end of a standard cell row.

-y_increment {*y_inc_list*}

Specifies a distance as a list of y-increments in microns. The values in the list are used as y-direction spacings between two header or footer cells during placement exploration. By default the command determines the y-increment from the height of the MTCMOS cell.

-orientation {**N** | **W** | **S** | **E** | **FN** | **FE** | **FS** | **FW**}

Specifies the orientation of header or footer cell placement. The valid values are **N**, **W**, **S**, **E**, **FN**, **FE**, **FS**, and **FW**. The default orientation is **N**.

-voltage_area *voltage_area*

Specifies a collection that contains only one voltage area. The header or footer cells are placed in the specified voltage area. By default, the top level is assumed.

-no_placement
Assumes that the current design is already properly placed. This skips the initial placement step and proceeds to do MTCMOS header or footer exploration and IR drop analysis. By default, the tool assumes that cells are not properly placed and performs a quick standard cell and macro placement.

-legalize_placement
Calls the **legalize_fp_placement** command after inserting a header or footer.

-no_pns
Skips all internal power network analysis (PNS) steps, but runs a user-specified PNS script.

-run_pns_script *file_name*
Runs a user-specified PNS TCL script instead of running an internal PNS step while exploring. The default is null.

-design *hierarchy_name*
Specifies the hierarchy name under which the header or footer is inserted.

-save_placement *prefix*
Saves each placement generated during explorations. The name of the saved design is derived from the name of the library cells, x-increment values, y-increment values, and the specified prefix. The default prefix is EHF_, so the default names of the saved design will be EHF_libCellName_yincrement_xincrement_designName. The number of saved designs depends on the number of placements explored. The saved placement might contain overlapping cells.

-preroute_mode_real_pg_port {rail | tie | net}
Specifies the mode of preroute when it is a preroute for the real power and ground port of a header or footer. The default mode is **net**.

-no_preroute_real_pg_port
Skips to preroute the real power and ground port of a header or footer.

-preroute_mode_virtual_pg_port {rail | tie | net}
Specifies the mode of preroute when it is the preroute for the virtual power and ground port of a header or footer. The default mode is **rail**.

-no_preroute_virtual_pg_port
Skips to preroute the virtual power and ground port of a header or footer.

DESCRIPTION

This command explores many placements of multithreshold XMOS (MTCMOS) header or footer cells with different x-spacings and y-spacings. Header and footer cells typically contain low leakage P/N transistors to reduce leakage current during power-down mode.

This command specifies a list of header or footer library reference cells and a two-dimensional grid for the header or footer cell placement. It runs fast placement of existing standard cells and macros and attempts all possible combinations of header or footer cells and spacings.

For each placement of header or footer cells, it estimates the IR drop using the power network analysis (PNA) engine and reports the results. Based on the results of the command, you can select the appropriate header or footer cells and their placement pitch to achieve the desired IR drop of the design.

Use this command to explore the placement of MTCMOS header or footer cells in the power network at an early floorplanning stage. It assumes that the metal preroutes of the power network do not exist. If the power straps of the design are already constructed, use the **create_power_switch_array** command to insert MTCMOS header and footer cells based on the existing power straps. By default, the **explore_power_switch** command does not modify the current design. Use the **-save_placement** option if you want to keep an explored placement.

Prerequisites

Before executing this command, you must perform the following tasks, in order, that are necessary for the Power Network Analysis (PNA) engine to compute the IR drop.

1. Define the power-on resistance of the header or footer cell by using the **set_attribute** command to set the **mtcmos_resistance** attribute.
2. Specify the metal layer for the power pins by using the **set_attribute** command to set the **mtcmos_pin_layers** attribute.
3. Define the power pad or virtual power pad by using the **create_fp_virtual_pad** command.
4. Verify that the virtual and real power and ground nets of all standard cells have been properly connected logically in the netlist. Use the **derive_pg_connection** command, if necessary. A standard cell to be powered down should connect to a virtual power or ground net. A standard cell that always stays on should be connected to a real power or ground net.
5. Use **set_mtcmos_pna_strategy** to define PNA, PNS, insert header-related and footer-related options.

EXAMPLES

The following example explores three placements of footer cells FOOTER_8, FOOTER_16, and FOOTER_32 at {20um, 50um, 100um} spacing in voltage area VA_1. The ground nets to be analyzed are VIRTUAL_VSS and VSS. IR drop analysis of each placement is reported.

```
prompt> explore_power_switch \  
-lib_cells {"FOOTER_8" "FOOTER_16D" "FOOTER_32D"}\  
-x_increment {20 50 100} -voltage_area VA_1\  
-virtual_pg_net VIRTUAL_VSS -real_pg_net VSS
```

The following example shows the use of the command under UPF mode, using power switches PS_A, PS_B, and PS_C.

```
prompt> explore_power_switch \  
-lib_cells {"PS_A" "PS_B" "PS_C"}\  
-x_increment {20 50 100} -voltage_area VA_1\  
-virtual_pg_net VIRTUAL_VSS -real_pg_net VSS
```

SEE ALSO

`create_power_switch_array(2)`
`analyze_fp_rail(2)`
`create_fp_placement(2)`
`legalize_fp_placement(2)`
`optimize_power_switch(2)`
`preroute_standard_cells(2)`
`replace_power_switch(2)`
`report_mtcmos_pna_strategy(2)`
`set_attribute(2)`
`set_mtcmos_pna_strategy(2)`
`synthesize_fp_rail(2)`

extract_blockage_pin_via

Creates a FRAM view of a standard cell or pad cell by extracting blockage, pin, and via information from the CEL view.

SYNTAX

```
status extract_blockage_pin_via
-library_name library_name
-cell_name cell_name
[-generate_boundary collection_of_sides]
[-cell_types collection_of_cell_types]
[-preserve_all_metal_blockage]
[-routing_blockage_output_layer metBlk | rGuide | zeroG]
[-treat_all_blockage_as_thin_wire]
[-treat_metal_blockage_as_thin collection_of_layers]
[-extract_pin_connectivity_through collection_of_layers]
[-poly_pin_text_layers list_of_layers]
[-m1_pin_text_layers list_of_layers]
[-m2_pin_text_layers list_of_layers]
[-m3_pin_text_layers list_of_layers]
[-m4_pin_text_layers list_of_layers]
[-m5_pin_text_layers list_of_layers]
[-m6_pin_text_layers list_of_layers]
[-m7_pin_text_layers list_of_layers]
[-m8_pin_text_layers list_of_layers]
[-m9_pin_text_layers list_of_layers]
[-m10_pin_text_layers list_of_layers]
[-m11_pin_text_layers list_of_layers]
[-m12_pin_text_layers list_of_layers]
[-m13_pin_text_layers list_of_layers]
[-m14_pin_text_layers list_of_layers]
[-m15_pin_text_layers list_of_layers]
[-nwell_pin_text_layers list_of_layers]
[-pwell_pin_text_layers list_of_layers]
[-deep_nwell_pin_text_layers list_of_layers]
[-deep_pwell_pin_text_layers list_of_layers]
[-pin_must_connect_area_layer_number {layer number_name ...}]
[-auto_pin_must_connect_area_threshold collection_of_layer_values]
[-skip_rotated_via_region]
[-contact_selections collection_of_contactCode_numbers]
```

Data Types

<i>library_name</i>	string
<i>cell_name</i>	string
<i>collection_of_sides</i>	collection
<i>collection_of_cell_types</i>	collection
<i>collection_of_layers</i>	collection
<i>list_of_layers</i>	list
<i>collection_of_layer_values</i>	collection
<i>collection_of_contactCode_numbers</i>	collection

ARGUMENTS

`-library_name library_name`

Specifies the name of the library containing the cell.

`-cell_name cell_name`

Specifies the name of the standard cell or pad cell for which to create a FRAM view. The CEL view must already exist in the library.

`-generate_boundary collection_of_sides`

Specifies the sides of the cell boundary to generate, consisting of one or more of **LEFT**, **RIGHT**, **TOP**, and **BOTTOM**. By default, the extraction process uses the boundaries defined in the CEL view, if any, and generates boundaries for any edges not defined in the CEL view.

`-cell_types collection_of_cell_types`

Specifies the types of cells to extract when the `-cell_name` option uses the wild card character (*) or otherwise specifies multiple cells of different types. By default, all standard cells, macro cells, and pad cells specified by `-cell_name` are extracted. To restrict the types of cells extracted, use the `-cell_types` option and specify one or more of **STD**, **MACRO**, and **PAD**; then only the listed types of cells are extracted.

The cell types are defined as follows:

STD: standard cell, double height, triple height,
triple+ height, standard filler, tap, via cell

PAD: pad, corner, pad filler, flip chip pad

MACRO: macro, cover, flip chip driver

`-preserve_all_metal_blockage`

Preserves all system metal blockage layers as they exist in the CEL view and converts all system metal blockage layers to real metal layers. This option is recommended for standard cell input in LEF format.

By default, the extraction process trims system metal blockages around pins as needed to prevent design rule violations. Using this option, the system metal blockages are retained exactly, so be sure that the preserved system metal blockages do not violate the design rules.

`-routing_blockage_output_layer metBlk | rGuide | zeroG`

Specifies the type of output layer produced in the FRAM view to represent routing blockages: **metBlk** (system metal blockage), **rGuide** (route guide), or **zeroG** (zero spacing route guide). The router recognizes all three types of layers as blockage and will not route over them. The default output layer is **metBlk** (system metal blockage). The router treats system metal blockage as thin metal for all design rule checking.

`-treat_all_blockage_as_thin_wire`

This option works only for non-standard cells. It causes real metal blockages to be treated as thin wire for design rule checking purposes. By default, real metal blockages are treated as fat wire if fat or thin wire if thin. The spacing rules can be different for thin versus fat wire, which affects the spacing required between routes and real metal blockages. This option setting also affects the trimming of real metal blockages during extraction. The extraction process trims real metal blockages around pin areas to allow connections to be routed to the pins without violating the spacing to the real metal blockage areas.

-treat_metal_blockage_as_thin *collection_of_layers*

Use this option when system metal blockage covers all of the cell in a specific layer and you want the router to keep a spacing of at least minSpacing between the route wire and the macro cell boundary in that layer. With this option, the command shrinks the system metal blockage by (max value in fatTblSpacing) - (minSpacing) and converts it to real metal blockage. This option still treats the real metal blockage as fat metal but reshapes the real metal blockage to cause the router to keep a spacing of at least minSpacing between the route wire and the macro cell boundary in that layer.
These are the mask layers that can specified in this option:
poly m1 m2 m3 ... m15

-extract_pin_connectivity_through *collection_of_layers*

Specifies the cut layers through which to extract pin connectivity. Using this option causes the command to traverse connectivity through the specified cut and via layers to identify connections between pins on different layers. By default, the tool does not traverse connectivity through cut layers. These are the mask layers that can specified in this option:
polyCont v1 v2 v3 ... v14 bv1 bv2

-poly_pin_text_layers *list_of_layers*

Specifies the text layers associated with poly pins. If the pin text is on the same layer as the pin geometry, you do not need to use this option. However, if the pin text is on one or more layers different from the pin geometry layer, use this option to identify the associated text layers. The layers can be identified by the layer name or layer number.

-m1_pin_text_layers *list_of_layers*

Specifies the text layers associated with m1 pins. If the pin text is on the same layer as the pin geometry, you do not need to use this option. However, if the pin text is on one or more layers different from the pin geometry layer, use this option to identify the associated text layers. The layers can be identified by the layer name or layer number.

-nwell_pin_text_layers *list_of_layers*

Specifies the text layers associated with N-well bias pg pins. If the pin text is on the same layer as the pin geometry, you do not need to use this option. However, if the pin text is on one or more layers different from the pin geometry layer, use this option to identify the associated text layers. The layers can be identified by the layer name or layer number.

-pwell_pin_text_layers *list_of_layers*

Specifies the text layers associated with P-well bias pg pins.

-deep_nwell_pin_text_layers *list_of_layers*

Specifies the text layers associated with deep N-well bias pg pins.

-deep_pwell_pin_text_layers *list_of_layers*

Specifies the text layers associated with deep P-well bias pg pins.

-pin_must_connect_area_layer_number {*layer number_name* ...}

Specifies for each connection layer the associated spare layer being used to designate the required via connection area within the pin geometry. By default, the router can connect a wire to any part of a large pin. To restrict

the connection to a subarea of a large pin, specify the subarea polygon on a spare layer and specify the spare area associated with each connection layer using this option. List the connection layers and associated spare layers as in the following example:

```
{m1 31 m2 32 m3 33 ...}
```

You can specify each layer by its layer number or layer name. These are the mask layers that can be specified in this option:

```
poly m1 m2 m3 ... m15
```

-auto_pin_must_connect_area_threshold *collection_of_layer_values*

Specifies the width threshold for automatic determination of the required via connection area within the pin geometry. By default, the router can connect a wire to any part of a large pin. To automatically restrict the connection to a subarea of a pin with an irregular shape, use this option and specify each layer and its associated width threshold. For example,

```
{m1 0.56 m2 0.63 m3 0.63 ... }
```

Areas within a pin having a width smaller than the threshold are excluded from allowing a via connection to be made. For a fat pin, the first-dimension fat contact threshold is used to calculate the allowed connection area, so that the router does not drop a fat contact on the thin part of a fat pin. These are the mask layers that can be specified in this option:

```
poly m1 m2 m3 ... m15
```

-skip_rotated_via_region

Skips consideration of rotated vias during calculation of via regions. By default, the extraction process considers the use of both default and rotated vias during via region calculation.

-contact_selections *collection_of_contactCode_numbers*

Specifies the contact code numbers of any additional vias that are to be considered during calculation of via regions. By default, the extraction process considers the use of the default via only.

DESCRIPTION

This command creates a FRAM view of a standard cell or pad cell by extracting blockage, pin, and via information from the CEL view. The FRAM view of a cell contains all the information needed for placement and routing and excludes cell-internal information that is not needed for placement and routing. The exclusion of unnecessary data can reduce the database size, increase the routing accuracy, and reduce the routing time.

Both the **extract_blockage_pin_via** command and the **create_macro_fram** command create FRAM views of cells by extracting blockage, pin, and via information from CEL views. Use the **extract_blockage_pin_via** command for extraction of standard cells and pads. Use the **create_macro_fram** command for extraction of macros. The extraction process is the same for both commands. However, they offer slightly different options with respect to extraction of blockages and via regions.

System metal blockages defined in the CEL view can be converted into system metal blockages, route guides, or zero-spacing route guides in the FRAM view. Proper extraction of system metal blockages can be used to prevent DRC violations between routes and cell-internal metal, while allowing the greatest possible flexibility to the router. You can choose to preserve all metal blockages as defined in the CEL

view or have them trimmed away from pins to allow route connections while preventing DRC violations. For a non-standard cell, you can also choose to have some or all routing layers treated as thin wire rather than fat wire for blockage extraction and DRC purposes.

If the CEL view contains multiple-layer pins, use the -
extract_pin_connectivity_through option to specify the cut and via layers used in those pins. Then the command traces the connectivity through the specified cut and via layers to identify connections between pins on different layers.

If the layers used for pin text are different from the pin geometry layers, use the respective command options to specify which text layers are associated with each pin layer (for example, **-m1_pin_text_layers** for the m1 layer).

As part of the FRAM extraction process, the tool determines the area within each pin where the center of a via can connect. This region, called the via region for the pin, is part of the generated FRAM view. This information gives the router guidance about where it can make connections to the pin. As a result, the router can find the via connection position quickly. Calculating the via region at extraction time saves significant runtime for router. However, this information only provides guidance. Even without the via region information, the router can still find the via connection position, but it must spend more runtime to find it.

The extraction process offers two methods for determining the via region for each pin, invoked by the following options:

```
-pin_must_connect_area_layer_number {layer number_name ...}
```

```
-auto_pin_must_connect_area_threshold {layer width ...}
```

Using the "must connect area layer" method, you explicitly specify the geometry of the via region for each pin on a spare layer. Then you use the option to specify each pin layer and its corresponding via region layer.

Using the "automatic pin must connect area" method, the tool automatically determines the via region for each pin. This method is suitable for pins with irregular shapes such as fingered geometries. For each pin layer, you specify a width threshold. Areas within a pin having a width smaller than the threshold are excluded from allowing a via connection to be made. For a fat pin, the first-dimension fat contact threshold is used to calculate the allowed connection area, so that the router will not drop a fat contact on the thin part of fat pin.

You can optionally specify whether to use rotated vias and additional available vias for automatic determination of the via region; use the **-skip_rotated_via_region** option and the **-contact_selections** options. By default, the extraction process considers only the default via and its rotation.

If you use neither the "must connect area layer" option nor the "automatic pin must connect area" option, the command determines the via region of each pin by itself. If no via region is produced on a pin, the router must determine an acceptable connection to the pin by itself.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example extracts the blockage, pin, and via information from the CEL view of the cell called `my_cell` in the library called `arc_rams_LM`, thereby produced a FRAM view of the cell containing only the information needed for placement and routing. The pin text for layer `m3` is contained in layer number 33, not in the `m3` layer itself.

```
prompt> extract_blockage_pin_via -library_name arc_rams_LM \  
        -cell_name my_cell \  
        -m3_pin_text_layers 33
```

The following example performs FRAM view extraction for all standard cells and pad cells, but not macro cells, in the library called `arc_rams_LM`.

```
prompt> extract_blockage_pin_via -library_name arc_rams_LM \  
        -cell_name * \  
        -cell_types {STD PAD}
```

The following example performs FRAM view extraction for all the cell called `my_cell12` in the library called `arc_rams_LM`. The blockages are produced as route guides rather than metal blockage.

```
prompt> extract_blockage_pin_via -library_name arc_rams_LM \  
        -cell_name my_cell12 \  
        -routing_blockage_output_layer rGuide
```

SEE ALSO

`create_macro_fram(2)`

extract_flcc_hotspot

Performs lithography hot spot extraction (model-based). FLCC stands for Fast Lithography Compliance Check.

SYNTAX

```
extract_flcc_hotspot  
[-cell_name
```

Data Types

cName string

ARGUMENTS

```
-cell_name cName
```

Specifies the cell name in which the extracted flcc hotspots will be stored. It is an Milkyway CEL view cell, and is used exclusively to store flcc hotspot info, similar to drc error cell. User can use ICC error browser to open it and the flcc hotspots will be classified according to its categories. If this param is not specified, the flcc hotspot info will be stored in original top cell. If user opens the top cell with error browser, user still can view these flcc hotspots, but no category classification is supported.

DESCRIPTION

The command `flcc_extract_hotspot` requires two files to execute, `viewHotSpots.ev` and `viewHotSpots.util`. User should have these two files in the working directory before running this command.

If flcc errors are not stored in original top cell, it will facilitate the user to browse and compare the flcc errors before and after fixing. The param `-cell_name` is used to store flcc errors. The error cell is a Milkyway CEL view cell, and only contains error types, error objects and related descriptions.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command performs `extract_flcc_hotspot`:

```
prompt> extract_flcc_hotspot -cell_name B
```

SEE ALSO

`detect_flcc_hotspot(2)`

`extract_flcc_hotspot`
730

```
fix_flcc_hotspot(2)
```

extract_fp_rail_to_constraints

Extracts existing power network structure to a file with a set of power network synthesis constraints. The file can be used to recreate a power network that is the same as the existing one.

SYNTAX

```
status extract_fp_rail_to_constraints
[-power_net power_net_name]
[-output_file output_file_name]
[-ground_net ground_net_name]
[-rail_type number | pitch]
[-ignore_layers layer_names]
[-preroute_script script_file_name]
```

Data Types

<i>power_net_name</i>	string
<i>output_file_name</i>	string
<i>ground_net_name</i>	string
<i>layer_names</i>	string
<i>script_file_name</i>	string

ARGUMENTS

-power_net *power_net_name*
Specifies the name of the power net to be extracted.

-output_file *output_file_name*
Specifies the name of the output file.

-ground_net *ground_net_name*
Specifies the name of the ground net to be extracted.

-rail_type *number* | *pitch*
Specifies the rail type of the existing power network. The valid values are *number* or *pitch*. Use the *number* option to create constraints with a fixed number of straps. Use the *pitch* option to create constraints with a fixed pitch. The default is *number*.

-ignore_layers *layer_names*
Specifies the names of the layers to be ignored during the extraction process. By default, the command does not ignore any layers.

-preroute_script *script_file_name*
Specifies the name of a preroute constraints file with preroute Tcl commands, such as the **set_preroute_advanced_via_rule** command, and options.

DESCRIPTION

This command analyzes the existing power network with its core ring, block rings,

and straps, and then creates a file with power network synthesis (PNS) constraints that is used to recreate the original power network. In the constraints, the width for each PG type, such as core ring, block rings, and straps for each layer is fixed. The widths in the constraints are obtained from the average widths of each PG type of the original PG structure.

There are two types of strap creations in the file: fixed straps number and fixed straps pitch. Fixed straps number keeps the same number of straps while design size is changed; and fixed straps pitch maintains the same pitch.

You can use the **-preroute_script** option to add customized preroute constraints, such as **set_preroute_advanced_via_rule** constraints, to fulfill a special requirement, such as advanced via rule. Apply the command before executing the **estimate_fp_area** command for MinChip flow. After that, you can run the **estimate_fp_rail -run_pns_script** command, specifying the preroute constraints file, to honor the PG structure during minimum die size exploration. The following example shows a created power network synthesis (PNS) constraints file.

```
# Skip IR drop simulation
set_fp_rail_strategy -pns_skip_ir true

# Define layer constraints with the same pitch
set_fp_rail_constraints -add_layer -layer m5 -direction horizontal \
    -max_pitch 47.460 -min_pitch 47.460 -max_width 6.870 \
    -min_width 6.870 -spacing minimum
set_fp_rail_constraints -add_layer -layer m6 -direction vertical \
    -max_pitch 47.460 -min_pitch 47.460 -max_width 6.890 \
    -min_width 6.890 -spacing minimum

# Define core ring constraints
set_fp_rail_constraints -set_ring -nets {VDD VSS} -horizontal_ring_layer {m5} \
    -vertical_ring_layer {m6} -ring_width 9.500 -extend_strap core_ring

# Define block ring constraints
set_fp_block_ring_constraints -add -horizontal_layer m5 -horizontal_width 4.000 \
    -vertical_layer m6 -vertical_width 4.000 -block_type instance \
    -block {tarang1/i_RAM} -nets {VDD VSS}
set_fp_block_ring_constraints -add -horizontal_layer m5 -horizontal_width 4.000 \
    -vertical_layer m6 -vertical_width 4.000 -block_type instance \
    -block {tarang2/i_RAM} -nets {VDD VSS}
set_fp_block_ring_constraints -add -horizontal_layer m5 -horizontal_width 4.000 \
    -vertical_layer m6 -vertical_width 4.000 -block_type instance \
    -block {tarang3/i_RAM} -nets {VDD VSS}
set_fp_block_ring_constraints -add -horizontal_layer m5 -horizontal_width 4.000 \
    -vertical_layer m6 -vertical_width 4.000 -block_type instance \
    -block {tarang4/i_RAM} -nets {VDD VSS}

# synthesize and then commit
synthesize_fp_rail -nets {VDD VSS}
commit_fp_rail
```

Multicorner-Multimode Support

This command is not supported in a multi-scenario design flow.

EXAMPLES

The following example extracts a PG network with power net named VDD and a ground net named VSS. It then creates a power network constraints file named PG.tcl that has the same pitch.

```
prompt> extract_fp_rail_to_constraints \  
        -power_net VDD -ground_net VSS -output_file PG.tcl -type pitch
```

The following example extracts a PG network with power net named VDD and a ground net named VSS. It then creates a power network constraints file named PG.tcl that has the same straps number, ignoring a layer named M2, and specifying a file that has preroute constraints named preroute.tcl.

```
prompt> extract_fp_rail_to_constraints \  
        -power_net VDD -ground_net VSS -output_file PG.tcl \  
        -ignore_layers {M2} -preroute_script preroute.tcl
```

SEE ALSO

```
analyze_fp_rail(2)  
commit_fp_rail(2)  
estimate_fp_area(2)  
set_fp_rail_constraints(2)  
set_fp_rail_strategy(2)  
set_preroute_advanced_via_rule(2)
```


extract_fp_relative_location

Extracts relative location constraints from the current placement.

SYNTAX

```
status extract_fp_relative_location
[-target_cells cells]
[-target_corner bl | br | tl | tr]
[-anchor_object object_name]
[-anchor_corner bl | br | tl | tr]
[-output_file file_name]
```

Data Types

<i>cells</i>	collection
<i>object_name</i>	string
<i>file_name</i>	string

ARGUMENTS

-target_cells *cells*
Specifies the macro cells whose locations are to be extracted in the format of relative location constraints. The default is to extract all macro cells.

-target_corner bl | br | tl | tr
Specifies the corner of the target cell to be used to extract the constraints. The valid values and their descriptions are listed below.
bl is the bottom-left corner.
br is the bottom-right corner.
tl is the top-left corner.
tr is the top-right corner.
The default is *bl*.

-anchor_object *object_name*
Specifies the name of the anchor object. An anchor object can be a plan group, fixed macro cell, or core area. The default is the name of a core area.

-anchor_corner bl | br | tl | tr
Specifies the corner of the anchor object's bounding box to be used in extracting constraints. The valid values and their descriptions are listed below.
bl is the bottom-left corner.
br is the bottom-right corner.
tl is the top-left corner.
tr is the top-right corner.
The default is *bl*.

-output_file *file_name*
Specifies the name of the file to which to dump the extracted constraints. The default is the name of the output to the xterm and the IC Compiler GUI log windows.

DESCRIPTION

This command extracts the location constraints of macro cells relative to an anchor object from the current placement. The output constraints from this command can be loaded back to the tool to restore the current placement of the macro or to restrict macro placement to respect these constraints during design planning.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example extracts the location constraints for the top-right of all macro cells, with respect to the bottom-right corner of the core area bounding box.

```
prompt>extract_fp_relative_location\  
-target_corner "tr" -anchor_corner "bl"
```

SEE ALSO

```
remove_fp_relative_location(2)  
set_fp_relative_location(2)
```

extract_hier_antenna_property

Extracts the hierarchical antenna properties of all the top-level ports in the given cell.

SYNTAX

```
integer extract_hier_antenna_property  
[-cell_name cell_name]
```

Data Types

cell_name string

ARGUMENTS

-cell_name *cell_name*
Cell instance name in the design from which to extract antenna properties.
This option is required.

DESCRIPTION

This command extracts the hierarchical antenna properties of all the top-level ports in the given cell routed by Astro or IC Compiler. The hierarchical antenna properties include gate size, routing area, and diode protection for all the ports of the macros. These properties can be used by the tool for checking antenna rules on the nets connected to the macro pins.

The **extract_hier_antenna_property** command extracts all information relevant to antenna rules connected to each port. You can use this information for antenna calculation in the higher-level cell hierarchy to avoid repetitive calculation, and flattening of the data.

The hierarchical antenna properties are calculated based on the following information:

- Routing contained in the CEL view of the top-level cell
- The hierarchical antenna properties of the cell instances
- The gate sizes and the diode protection for each port of the library cells, which you set by loading the CLF file.

The hierarchical antenna properties are stored in the FRAM view of the cell. Both the CEL and FRAM views of the cell must exist before this command is used.

Prerequisites

The cell library must be opened first.

Both the CEL view and the FRAM view of the cell must exist and be writable before

you run this command.

The antenna properties of all port instances in the cell must be properly set. Otherwise, the antenna property calculated will not be accurate. You can set the antenna properties by doing either of the following:

- Running the **extract_hier_antenna_property** command on the child cells, if the child cell is placed and routed by Astro or IC Compiler
- Loading a CLF file that contains the gate size and diode protection value of each port, if the child cell is a standard cell

Limitations:

The **extract_hier_antenna_property** command does not work on standard library cells or hard macros because Astro or IC Compiler do not read the diffusion layer in the layout. The results may not be accurate if hard macros exist in the cell design.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

```
prompt> extract_hier_antenna_property -cell_name my_cell
```

extract_rc

Executes 2.5D extraction for routes in a design.

SYNTAX

```
status extract_rc  
[-coupling_cap]  
[-estimate]  
[-routed_nets_only]
```

ARGUMENTS

- coupling_cap**
Specifies for the extraction to consider coupling capacitance. Specifying this option can cause the Standard Parasitic Exchange Format (SPEF) file to be very large. By default, extraction does not consider coupling capacitance.
- estimate**
Performs RC estimation for both routed and unrouted nets. By default, the **extract_rc** command performs detailed RC calculation for fully routed nets and RC estimation for unrouted nets. For RC estimation, the command performs virtual routing and delay calculation, and back-annotates the delay and capacitance on the current design. After running this command, you can use the **report_timing** command to see the estimated timing results.
- routed_nets_only**
Performs RC extraction of fully routed nets only, and skips RC estimation of unrouted routed nets. By default, the **extract_rc** command performs both RC extraction from fully routed nets and RC estimation of unrouted nets.

DESCRIPTION

This command performs 2.5D extraction from the routes in memory. It calculates delay based on the Elmore delay model.

If you use the **set_tlu_plus_files** command to specify the TLUPlus technology files, the tool performs extraction based on TLUPlus technology.

If your physical library does not contain extraction parameters, the tool derives the resistance and capacitance values of routes from the available RC estimation method in your physical library. Use the **RCEX-015** message to determine the RC estimation method used.

To extract coupling capacitance, use the **-coupling_cap** option.

To fine-tune the extraction results, use the **set_extraction_options** command. This command allows you to specify various extraction options, such as scaling factors for resistance and capacitance values.

Multicorner-Multimode Support

In post-route, this command extracts parasitic corners from both active and inactive scenarios. In pre-route, this command extracts parasitic corners from active scenarios.

EXAMPLES

The following performs virtual routing and updates back-annotated net delays.

```
prompt> extract_rc -estimate
```

The following example considers coupling capacitance.

```
prompt> extract_rc -coupling_cap
```

SEE ALSO

```
extract2plib(1)  
read_parasitics(2)  
report_timing(2)  
set_extraction_options(2)  
set_operating_conditions(2)  
set_tlu_plus_files(2)  
write_parasitics(2)
```

extract_rp_group

Extracts a relative placement group from your design based on your specifying either a collection of cells or a bounding box.

SYNTAX

```
status extract_rp_group
-group_name group_name
[-rows number_of_rows]
[-columns number_of_columns]
[-output file_name]
[-append]
[-physical]
[-nosplit]
[-objects object_list]
[-coordinates coordinate_list]
[-apply]
[-descending]
```

Data Types

<i>group_name</i>	string
<i>number_of_rows</i>	integer
<i>number_of_columns</i>	integer
<i>file_name</i>	string
<i>object_list</i>	collection
<i>coordinate_list</i>	list

ARGUMENTS

-group_name *group_name*
Specifies a name for the extracted relative placement group. Each relative placement group in the design must have a unique name.

-rows *number_of_rows*
Specifies the number of rows in the extracted relative placement group. By default, the relative placement groups contains one row for each cell in the group.

-columns *number_of_columns*
Specifies the number of columns in the extracted relative placement group. By default, the relative placement group contains one column.

-output *file_name*
Specifies the name of the output file to which to write the constraints. By default, the constraints are written to the screen.

-append
Appends constraints to the file specified by the **-output** option. By default, if the specified file exists, it is overwritten.

-physical
 Uses the physical location of the cells when extracting the structure of the relative placement group.

-nosplit
 Prevents line splitting when writing the relative placement constraints.

-objects *object_list*
 Specifies the list of cells for the tool to use to extract the relative placement structure.

-coordinates *coordinate_list*
 Gets the cells from the specified bounding box and then uses the physical locations of these cells to extract the relative placement structure. The *coordinate_list* value is in integers.

-apply
 Applies the relative placement constraints generated by the command to the design. By default, the constraints are written out, but are not applied to the design.

-descending
 Adds the cells to the relative placement group in reverse order.

DESCRIPTION

This command allows the creation of relative placement groups using existing characteristics of your design. It extracts relative placement groups from that design in either of the following ways.

- You can specify the contents of the relative placement group either by specifying a collection of cells or by specifying a bounding box containing the cells.
- You can either specify the structure of the relative placement group by specifying the number of rows and columns, or you can let the tool generate the structure based on the physical locations of the cells in the group.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following examples extract relative placement groups.

```
prompt> extract_rp_group -group_name misc2\  
-objects [get_cells -hier {U2/I12?}] -row 1  
1  
prompt> extract_rp_group -group_name misc9\  
-objects [get_cells -hier {U2/I9?}] -row 1\  
-output rp.tcl -append
```



```
1
prompt> extract_rp_group -group_name ph1 -physical\
-coordinates {0 0 100 100}
1
prompt> extract_rp_group -group_name ph2 -physical\
-objects [get_cells -hier {u2/I9?}]
```

SEE ALSO

```
create_rp_group(2)
get_rp_groups(2)
remove_rp_groups(2)
write_rp_groups(2)
```

extract_zrt_hier_antenna_property

Extracts the hierarchical antenna properties of all the top-level ports in the specified cell. You should use this command for designs routed with Zroute.

SYNTAX

```
status extract_zrt_hier_antenna_property  
-cell_name cell_name
```

Data Types

cell_name string

ARGUMENTS

```
-cell_name cell_name  
           Specifies the cell instance name in the design from which to extract antenna  
           properties.
```

DESCRIPTION

This command extracts the hierarchical antenna properties of all the top-level ports in the given cell. It is recommended that you use this command for designs routed with zroute. The hierarchical antenna properties include gate size, routing area, and diode protection.

This command should be run within the child cell of each macro and saved. When antenna analysis is done at the top level, the properties that were previously set can be seen in each macro pin. When nets connecting to macro pins are analyzed, the antenna information from each macro pin is used for analyzing that pin. This eliminates repetitive calculation of antenna properties of each macro or flattening the design.

The hierarchical antenna properties are calculated based on the following information:

- Routing contained in the CEL view of the top-level cell.
- The hierarchical antenna properties of the cell instances
- The gate sizes and the diode protection for each port of the library cells, which can be set by loading the CLF file.

The hierarchical antenna properties are stored in the FRAM view of the cell. Both the CEL and FRAM views of the cell must exist before you execute this command.

Prerequisites

The design library must be opened first.

Both the CEL view and the FRAM view of the cell must exist and be writable before the command is run.

The antenna properties of all port instances in the cell must be properly set. Otherwise, the antenna property calculated will not be accurate. The antenna properties can be set by:

- Running the **extract_hier_antenna_property** command on child cells, if the child cell is routed by zroute.
- Loading a CLF file that contains the gate size and diode protection value of each port, if the child cell is a standard cell

Limitations

The **extract_hier_antenna_property** command does not work on standard library cells or hard macros because the tool does not read the diffusion layer in the layout. The results might not be accurate if hard macros exist in the design.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows the use of the command on a cell instance named my_cell.

```
prompt> extract_zrt_hier_antenna_property -cell_name my_cell
```

```
SH "SEE ALSO"
```

```
extract_hier_antenna_property(2)
```

filter_collection

Filters a collection, resulting in a new collection. The base collection remains unchanged.

SYNTAX

```
collection filter_collection  
base_collection  
expression  
[-regexp [-nocase]]
```

Data Types

<i>base_collection</i>	collection
<i>expression</i>	string

ARGUMENTS

base_collection
Specifies the base collection to be filtered. This collection is copied to the result collection. Objects are removed from the result collection if they are evaluated as false by the conditional *expression*.

expression
Specifies an expression with which to filter *collection*.

-regexp
Indicates that the real regular expressions are to be used by the =~ and !~ filter operators. By default, these filter operators use simple wildcard pattern matching with the '*' and '?' wildcards.

-nocase
Makes the pattern match case-insensitive. You can use the **-nocase** option only if you also use the **-regexp** option.

DESCRIPTION

This command filters a collection after the collection has been created. In many cases, the commands that create collections support a **-filter** option, which performs the filtering as part of the collection process. Using the **-filter** option in those commands is almost always more efficient than using the **filter_collection** command to filter the collection after it has been created. The **filter_collection** command is most useful in the situation where you want to filter the same large collection many times using different criteria.

The **filter_collection** command results in either a new collection containing the subset of the objects in the input specified by using the *base_collection*, or it results in the empty string (an empty collection) indicating that the *expression* filtered out all elements of the input *base_collection*.

The basic form of the conditional expression is a series of relations joined

together with AND and OR operators. Parentheses are also supported. The basic relation contrasts an attribute name with a value by using a relational operator. For example, given the following information:

```
is_hierarchical == true and area <= 6
```

the relational operators are the following:

```
==    Equal
!=    Not equal
>     Greater than
<     Less than
>=    Greater than or equal to
<=    Less than or equal to
=~    Matches pattern
!~    Does not match pattern
```

and the basic relational rules are the following:

- String attributes can be compared with any operator.
- Numeric attributes cannot be compared with pattern match operators.
- Boolean attributes can be compared only with == and !=. The value can be only true or false.

For a complete description of conditional expressions, see the reference manual for the tool.

Regular expression matching is the same as in the Tcl **regexp** command. When using the **-regexp** option, be careful how you quote the filter *expression*. Using rigid quoting with curly braces around regular expressions is recommended. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search simply by adding `".*"` to the beginning or end of the expressions, as needed. You can make the regular expression search case-insensitive by using the **-nocase** option with the **-regexp** option.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates a collection of only hierarchical cells.

```
prompt> set a [filter_collection [get_cells *]"is_hierarchical == true"]\
{"Adder1", "Adder2"}
```

The following example creates a collection with an input range from 18 to 23.

```
prompt> set a [filter_collection [all_inputs]-regex {name=~"in\[([1][8-9] | [2][0-3])\]"}}\
{"in[23]", "in[22]", "in[21]", "in[20]", "in[19]", "in[18]"}
```

SEE ALSO

`collections(2)`

fix_flcc_hotspot

Performs lithography hot spot fixing. FLCC stands for Fast Lithography Compliance Check.

SYNTAX

```
fix_flcc_hotspot
```

DESCRIPTION

This command requires two files to execute: fixHotSpots.ev and fixHotSpots.util. User should have these two files in the working directory before running this command.

No error cell will be generated after fix_flcc_hotspot, since there is no direct way to know if the original flcc hotspots are fixed or if new flcc hotspots are created by fixing. User has to save the design and rerun detect_flcc_hotspot and extract_flcc_hotspot to get such info.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command performs fix_flcc_hotspot.

```
prompt> fix_flcc_hotspot
```

SEE ALSO

```
detect_flcc_hotspot(2)  
extract_flcc_hotspot(2)
```

fix_isolated_via

Fixes isolated via violations by inserting hang-on vias in specified range.

SYNTAX

```
integer fix_isolated_via  
[-isolated_via_spacing distance]  
[-isolated_via_quadrant_spacing distance]
```

Data Types

distance distance

ARGUMENTS

-isolated_via_spacing *distance*

Determines the adjacent via spacing, in microns, that triggers isolated via violation. If there is no adjacent via located less than *n* microns from the via, the violation of type "Isolated Via" is fixed. If set, this argument will be stored in a "droute" cell parameter "isolatedViaSpacing". The range is 0.0 to 20.0. The default is the value, set in a "droute" cell parameter "isolatedViaSpacing", which in turn has a default of 1.0.

To set a distance in a "droute" cell parameter "isolatedViaSpacing", use
set_parameter -type real -module droute \
 -name isolatedViaSpacing -value *n*

-isolated_via_quadrant_spacing *distance*

Determines the adjacent via spacing, in microns, in four quadrants. If adjacent vias that cover all four quadrants are not located less than *n* microns from the via, the violation of type "Isolated Via" is fixed. If set, this argument will be stored in a "droute" cell parameter "isolatedViaQuadrantSpacing". The range is 0.0 to 50.0. The default is the value, set in a "droute" cell parameter "isolatedViaQuadrantSpacing", which in turn has a default of 10.0.

To set a distance in a "droute" cell parameter "isolatedViaQuadrantSpacing", use
set_parameter -type real -module droute \
 -name isolatedViaQuadrantSpacing -value *n*

DESCRIPTION

This command is used to fix isolated vias rule violations. If two vias are not aligned, corner-to-corner spacing is measured for *isolated_via_spacing* and *isolated_via_quadrant_spacing*. If two vias are aligned, edge-to-edge spacing is measured for *isolated_via_spacing* and *isolated_via_quadrant_spacing*. This command will check and fix *isolated_via_spacing* first, and then *isolated_via_quadrant_spacing*. The value of *isolated_via_quadrant_spacing* should be greater than *isolated_via_spacing*. Please be advised that greater values affect the speed of command execution. The via inserted by this feature will be connected to one metal layer, and hence will be a hang-on via. This command needs to be used just before GDS out.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example fixes isolated vias rule violations for adjacent vias that are located less than 5.0 microns from the via (`isolated_via_quadrant_spacing` is 10.0 by default):

```
prompt> fix_isolated_via -isolated_via_spacing 5.0
```

APPENDIX

The following settings stored in "droute" cell parameters, and set in command **set_parameter**, will control defaults and operation of **fix_isolated_via** command:

Common for all via layers parameters:

"isolatedViaSpacing" and "isolatedViaQuadrantSpacing".

The range for "**isolatedViaSpacing**" is 0.0 to 20.0. The default is 1.0. The range for "**isolatedViaQuadrantSpacing**" is 0.0 to 50.0. The default is 10.0.

To set these parameters, use:

```
set_parameter -type real -module droute \  
              -name isolatedViaSpacing -value n  
set_parameter -type real -module droute \  
              -name isolatedViaQuadrantSpacing -value n
```

To fix isolated via violations on specified via layers only, the following settings should be set with command **set_parameter**:

"checkIsolatedViaPerLayer" - if this parameter is set, the rule is applied only to cut layers, which have "per-layer" isolated via spacing set, and if it is not set, the rule is applied to all cut layers. Valid values are 0 (which is the default) and 1. If it is set to 1, the values of common for all via layers parameters "**isolatedViaSpacing**" and "**isolatedViaQuadrantSpacing**" are ignored. The arguments to **fix_isolated_via** command that also can set these common parameters, - `isolated_via_spacing` and `-isolated_via_quadrant_spacing`, are also ignored.

To set or reset this run-time integer parameter, use:

```
set_parameter -type int -module droute \  
              -name checkIsolatedViaPerLayer -value n
```

To set "per-layer" isolated via spacing and isolated via quadrant spacing, use the following real parameters:

```
set_parameter -type real -module droute \  
              -name polyContIsolatedViaSpacing -value n  
set_parameter -type real -module droute \  
              -name polyContIsolatedViaQuadrantSpacing -value n
```

```

        -name via1IsolatedViaSpacing -value n
set_parameter -type real -module droute \
        -name via2IsolatedViaSpacing -value n
set_parameter -type real -module droute \
        -name via3IsolatedViaSpacing -value n
set_parameter -type real -module droute \
        -name via4IsolatedViaSpacing -value n
set_parameter -type real -module droute \
        -name via5IsolatedViaSpacing -value n
set_parameter -type real -module droute \
        -name via6IsolatedViaSpacing -value n
set_parameter -type real -module droute \
        -name via7IsolatedViaSpacing -value n
set_parameter -type real -module droute \
        -name via8IsolatedViaSpacing -value n
set_parameter -type real -module droute \
        -name via9IsolatedViaSpacing -value n
set_parameter -type real -module droute \
        -name via10IsolatedViaSpacing -value n
set_parameter -type real -module droute \
        -name via11IsolatedViaSpacing -value n

set_parameter -type real -module droute \
        -name polyContIsolatedViaQuadrantSpacing -value n
set_parameter -type real -module droute \
        -name via1IsolatedViaQuadrantSpacing -value n
set_parameter -type real -module droute \
        -name via2IsolatedViaQuadrantSpacing -value n
set_parameter -type real -module droute \
        -name via3IsolatedViaQuadrantSpacing -value n
set_parameter -type real -module droute \
        -name via4IsolatedViaQuadrantSpacing -value n
set_parameter -type real -module droute \
        -name via5IsolatedViaQuadrantSpacing -value n
set_parameter -type real -module droute \
        -name via6IsolatedViaQuadrantSpacing -value n
set_parameter -type real -module droute \
        -name via7IsolatedViaQuadrantSpacing -value n
set_parameter -type real -module droute \
        -name via8IsolatedViaQuadrantSpacing -value n
set_parameter -type real -module droute \
        -name via9IsolatedViaQuadrantSpacing -value n
set_parameter -type real -module droute \
        -name via10IsolatedViaQuadrantSpacing -value n
set_parameter -type real -module droute \
        -name via11IsolatedViaQuadrantSpacing -value n

```

The range for "per-layer" isolated via spacing is 0, 20.0; the default is 0.0, which disables processing isolated via rule on the specified via layer. The range for "per-layer" isolated via quadrant spacing is 0, 50.0; the default is 0.0, which disables processing isolated via quadrant rule on the specified via layer.

SEE ALSO

report_isolated_via(2)

```
set_parameter(2)
report_parameter(2)
```

fix_lcc_hotspot

Performs lithography compliance check (LCC) hotspot fixing.

SYNTAX

```
status fix_lcc_hotspot  
-lcc_file_path lcc_path_name  
[-types list_of_types]  
[-level level]  
[-num_loops number_of_loops]  
[-num_cpus number_of_cpus]
```

Data Types

<i>lcc_path_name</i>	string
<i>list_of_types</i>	list
<i>level</i>	double
<i>number_of_loops</i>	integer
<i>number_of_cpus</i>	integer

ARGUMENTS

-lcc_file_path *lcc_path_name*
Specifies a directory of LCC files, which contains the runsets necessary for LCC detection jobs. The *lcc_path_name* value must be a full path.

-types *list_of_types*
Specifies the types of hotspots to fix and detect. Each type represents a specific geometric combination of shapes. Currently, LCC has the following 5 valid types: **line**, **lineend**, **space**, **slotend**, and **voc**. You can choose any list of types to fix. The default is to fix all types.

-level *level*
Specifies the levels to fix when performing level-based LCC fixing. If you use this option, the command fixes only hotspots whose level is under the *level* value or equal to it. This means that the hotspots under this level are more serious than the hotspots at *level* or above.
The two level systems are LCC-t and LCC-g. The LCC-t level system has the following 4 valid values: **1**, **2.1**, **2.2**, and **2.3**. For example, if you set *level* to 2.1, the command fixes only hotspots whose level is 1 or 2.1 and ignores hotspots at level 2.2 and 2.3. The default value for the LCC-t level system is 2.1. The LCC-g level system has the following 5 valid values: **1**, **2**, **3**, **4**, and **5**. The default value for the LCC-g level system is 2. IC Compiler gets the correct level system (LCC-t or LCC-g) from LCC runsets, but you should know which level system is used and should enter the correct level value for this option.

-num_loops *number_of_loops*
Specifies the number of loops the command is to run. If the command executes the specified number of loops and all DRC (design rule check) and LCC hotspots are clean, the command stops. The default value is 50.

`-num_cpus` *number_of_cpus*

Specifies the number of CPUs the command uses to launch LCC checking jobs.
The default value is 1.

DESCRIPTION

This command attempts to fix LCC hotspots with an iterative tool-fixing and LCC-checking flow. Before running this command, the design must be DRC clean; otherwise there is no guarantee of convergence or fixing quality. Before fixing, two fix-guidance files named `out00` and `fout00` must exist under the working directory. These two files are generated by the **detect_lcc_hotspot** command and contain information on detected hotspots.

In each iteration, the tool first tries to fix hotspots by local-fix and reroute. Since new DRC violations or new LCC hotspots might be generated after fixing, the tool performs internal DRC checking and LCC jobs are launched to check the changed patterns to determine if new hotspots have been generated. The process runs until all LCC hotspots and DRC violations are clean or the iteration number has reached the value specified by *number_of_loops* in the **-num_loops** option. Most convergeable designs converge in less than 10 iterations.

Since run time can be long, distributed processing flow is suggested. Before running the distributed processing **fix_lcc_hotspot** command, you must run the **set_distributed_route** command to set the machines to be used. In this case, you should set *number_of_cpus* in the **-num_cpus** option to a value greater than 1.

An error cell named `<top_cell_name>_lcchs.err` is created after running the **fix_lcc_hotspot** command. You can open this error cell to browse remaining LCC hotspots in the current design.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows the command running on multiple Linux hosts.

```
prompt> set_distributed_route\  
-jp_machines {linux1 linux2 linux3 linux4 linux5}  
prompt> fix_lcc_hotspot\  
-lcc_file_path /root/LCC/Local-LCC\  
-types {line space lineend slotend}\  
-num_cpus 10 -level 1
```

SEE ALSO

```
detect_lcc_hotspot(2)  
report_lcc_hotspot(2)  
set_distributed_route(2)
```

flatten_clock_gating

Restructures integrated clock gating (ICG) cells to flatten the hierarchy of the clock tree for clock mesh creation.

SYNTAX

```
status flatten_clock_gating  
-icgs cells  
    | -mesh_nets nets  
    | -clocks clocks  
[-max_depth positive_integer]  
[-optimize]
```

Data Types

<i>cells</i>	collection of one or more cells
<i>nets</i>	collection of one or more nets
<i>clocks</i>	collection of one or more clocks

ARGUMENTS

-icgs *cells*
Specifies the ICG cells to be restructured. Each specified cell must be an ICG, and its clock input must be driven by another ICG. Cells that do not meet these requirements are ignored.

-mesh_nets *nets*
Flattens the ICG cells that are driven by the specified mesh nets.

-clocks *clocks*
Flattens the ICG cells that are on the specified clock paths.

-max_depth *positive_integer*
Specifies the maximum hierarchical level of ICGs driving other ICGs after flattening. The default is 1. A value greater than 2 is unusual.

-optimize
Restructures the logic after flattening to reduce the fanout or optimizes the structure.

DESCRIPTION

To create a good clock mesh design requires a low hierarchical level of logic tree under the mesh; otherwise, you do not get the benefit of low clock skew. You should avoid creating high logic-level trees under the clock mesh nets.

Because most ICGs are added by Power Compiler, the simplest approach is to limit the hierarchical level of ICGs by setting the appropriate options in Power Compiler. You can use **remove_clock_gating** in Power Compiler or **flatten_clock_gating** in IC Compiler to flatten the circuitry under the mesh. The **remove_clock_gating** command removes ICGs created by Power Compiler and replaces them with MUXs on the inputs to

registers. Rather than removing ICGs, the **flatten_clock_gating** command reconnects ICGs that are driven by other ICGs to make them directly driven by the clock source.

To retain the same logic behavior, the command adds circuitry to use the original enable signal and the enable signal of the parent ICG as inputs to the logic AND function. (Logical OR is used if the ICG enable pin has a negative enable sense.) Therefore, the resulting clock output should have the same logic behavior as the original output, but with one less level of gating between it and the clock root.

You should specify which gates to flatten when using this command. You can specify a collection of ICGs with the **-icgs** option. If you specify **-mesh_net**, the command searches recursively for ICGs driven by the specified mesh nets. ICGs that are deeper than the specified value by **-max_depth** will be flattened. The **-clock** option works similarly to the **-mesh_net** option, except that it searches for ICGs from the root of the clock tree.

By default, the command restructures the logic with an AND or OR gate for each flattened ICG. If the **-optimize** option is used, it restructures the logic to reduce electrical rule violations. However, the circuitry generated by this command might require further synthesis to meet timing and electrical constraints.

LIMITATIONS

This command is not as powerful or general as Power Compiler commands. It is better to use **remove_clock_gating**, **optimize_clock_gates**, or other standard synthesis operations if they can be applied to your designs.

This command only removes the lowest level of ICGs driving other ICGs. That is, if there are three levels of ICGs, this command will reduce them to two levels. If further restructuring is needed, repeat this command.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example flattens the ICGs in the clk1 clock path to a hierarchical level of 2.

```
prompt> flatten_clock_gating -clock clk1 -max_depth 2
```

SEE ALSO

```
identify_clock_gating(2)  
report_clock_gating_check(2)
```

flatten_fp_black_boxes

Flattens objects to remove them from the physical cell and restore them to hierarchy preservation (logical view).

SYNTAX

```
status flatten_fp_black_boxes  
black_boxes
```

Data Types

black_boxes collection

ARGUMENTS

black_boxes
 Specifies a collection of black boxes to flatten.

DESCRIPTION

This command flattens objects to remove them from the physical cell and restore them to hierarchy preservation (logical view).

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example flattens the black box named *alu1*.

```
prompt> flatten_fp_black_boxes [get_cells alu1]
```

SEE ALSO

```
estimate_fp_black_boxes(2)  
get_cells(2)
```


flatten_fp_hierarchy

Removes a level of hierarchy.

SYNTAX

```
int flatten_fp_hierarchy  
cell_list  
[-input_sdc_file input_file_name]  
[-output_sdc_file output_file_name]  
[-load_internal_pins]  
[-load_back_sdc]  
[-prefix prefix_name]  
[-simple_name]  
[-no_backslash_for_hierarchy_delimiter]
```

Data Types

<i>cell_list</i>	collection
<i>input_file_name</i>	string
<i>output_file_name</i>	string
<i>prefix_name</i>	string

ARGUMENTS

cell_list

Specifies a collection of cells in the current design that are to be flattened. The contents of these cells are brought up to the same level as the cells.

You can specify a collection of cells using the full name of each cell. Note that the following conditions might cause you to encounter a "User input data check failed" error:

- You specified a non-existent cell.
 - You specified a leaf cell.
 - The logical parent of the cell is multiply instantiated.
 - A plan group is defined on the cell you specified.
 - Two cells are on the same hierarchy path (for example, both A/B and A/B/C are specified).
- You can pass the "input data check" by analyzing your *cell_list* specification to correct for these conditions.

-input_sdc_file input_file_name

Specifies a Synopsys Design Constraints (SDC) file to update to reflect the netlist change. This SDC file should contain valid SDC commands for the current design.

-output_sdc_file output_file_name

Specifies the name of the output SDC file into which to write the updated SDC

constraints. You should use this option with the **-input_sdc_file** option.

-load_internal_pins

Indicates the design contains "internal pin" sdc constraints. This option is used for sdc constraints update only.

This option should be used with "-input_sdc_file" and "-output_sdc_file" options.

-load_back_sdc

Specifies that the updated SDC constraints should also be applied on the current design. You should use this option with the **-input_sdc_file** option.

-prefix *prefix_name*

Specifies the prefix in naming flattened cells. The naming format is as follows:

`<prefix>original_cell_name`

Avoid using the hierarchy delimiter in the specified prefix.

This option is mutually exclusive with the **-simple_name** and **-no_backslash_for_hierarchy_delimiter** options.

-simple_name

Specifies that simple, non-hierarchical names are to be used for those cells under the flattened cell.

For this option, cells maintain their original names. Since name clashing might happen when using the simple name, this command simply adds a "_1" suffix to avoid duplication. For example, if a cell named *abc* is under a flattened module named *B* and there is already an *abc* cell in the same level as the flattened module *B*, this command names the original cell *B/abc* to *abc_1* to avoid clashing with the same level cell *abc*.

This option is mutually exclusive with the **-prefix** and **-no_backslash_for_hierarchy_delimiter** options.

-no_backslash_for_hierarchy_delimiter

Specifies that the command does not add backslashes before the hierarchy delimiter. By default, the flattened cell's name and a backslashed hierarchy delimiter is used as the prefix of a new name.

For example, the cell named *C* under module *B*'s full name is *A/B/C*. After flattening module *B*, the cell *C*'s full name is *A/B\C* and its single name is *B\C*. By doing this, the application can easily process *B\C* as a whole single name, not a normal full name such as *B/C*. If you want *C*'s single name to look like *B/C*, you must specify this option.

This option is mutually exclusive with the **-simple_name** and **-prefix** options.

DESCRIPTION

Removes a single level of hierarchy from the current design by bringing up the contents of the specified cells to the same level as the cells.

By default, this command uses the flattened cell's name as the prefix of those flattened cells. For example, after flattening *A/B*, the original cell named *C* under cell *B* would have the new name *B\C*, if the hierarchy delimiter is a "/" (slash).

By specifying the **-no_backslash_for_hierarchy_delimiter** option, the new name

contains no backslash for the hierarchy delimiter and looks like *B/C*.

By specifying the **-simple_name** option, the cell *C*'s name would not be changed in the example shown above.

By specifying the **-prefix** option with the name *my_prefix*, the cell *C*'s name would be prefixed and changed to *my_prefixC*.

If you do not specify a prefix, the default is in the following format:

```
<flatten_cell_name>/old_cell_name
```

This command does not automatically update Synopsys Design Constraints (SDC) information stored in the current design unless you specify the **-input_sdc_file** and **-output_sdc_file** options or you reapply the SDC file later.

This command also removes design constraints stored in the current design, since the merge operation would produce inconsistencies with the existing design constraints. You must reapply design constraints after running this command.

This command ignores the **dont_touch** attribute on hierarchy cells. Any specified cell would be flattened by this command, so you must pay attention to the flatten cell list if you really want to leave such cells unchanged.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example flattens a specified list of cells.

```
prompt> flatten_fp_hierarchy {u1 u2 u3}
```

The following example specifies a collection of cells.

```
prompt> flatten_fp_hierarchy [get_cells u*]
```

The following example flattens a specific cell and specifies the prefix to use.

```
prompt> flatten_fp_hierarchy {u1} -prefix "U1:"
```

The following example flattens the cell instances named *MID1/BOT1/FOO1* and *MID1/BOT1/FOO2*. *MID1/BOT1* is a unique instantiation of a design named *BOT*.

```
prompt> flatten_fp_hierarchy {MID1/BOT1/FOO1 MID1/BOT1/FOO2}
```

SEE ALSO

`merge_fp_hierarchy(2)`

flip_mim

Flips the cell placement in a multiple instantiated module (MIM) plan group.

SYNTAX

```
status flip_mim  
[-direction X | Y]  
collection
```

Data Types

collection list

ARGUMENTS

-direction X | Y
 Identifies the direction of the flip. X(Y) direction is a flip around the y(x)-axis.

collection
 Specifies a list containing the multiple instantiated plan groups to flip.

DESCRIPTION

This command flips MIMs on an x-axis or y-axis. MIMs are modules that have the same reference. A set of these modules is called a MIM group. A design can have more than 1 MIM group. For example, if modules A1 and A2 have reference A and modules B1 and B2 have reference B, they form 2 MIM groups.

This command flips MIMs during the virtual flat stage. This is the period when MIMs are physically represented in the design as plan groups. Flipping includes flipping the placement of all cells in the given MIM and also flipping the shape of the MIM. During flipping, the bounding box of the plan group does not move. One way to visualize this process is that the resulting placement after flipping is the same as if the plan group were converted to a soft macro, flipped, and then converted back to a plan group.

Note that even if standard cells are placed at legal locations before flipping, it is not guaranteed that they will be legal after flipping. This command prints a warning if it determines that the resulting cells might not be placed at legal locations after the flip.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows flipping a MIM in the x-direction (around the y-axis).

```
prompt> flip_mim -direction X M1
```

SEE ALSO

```
copy_mim(2)  
report_mim(2)
```

flip_objects

Flips one or more movable objects.

SYNTAX

```
status flip_objects
[-anchor anchor_point]
[{-direction {x | y} | -x float | -y float}]
[-flip_transform]
[-ignore_fixed]
```

Data Types

anchor_point string

ARGUMENTS

-anchor *anchor_point*

Specifies an anchor point for the flip.

Use one of the valid values for *anchor_point*.

Valid values and their descriptions are the following:

ll - lower-left corner of the object.

ur - upper-right corner of the object.

center - center of the object.

The default is **ll**.

-direction {x | y

Specifies the direction of the flip if neither -x nor -y options are specified.

-x *float*

If no anchor point is specified then this option specifies the explicit x coordinate. If an anchor point is specified then this option specifies the x offset from the x coordinate of the anchor point.

The resultant x coordinate specifies the position of the vertical line to flip the objects about.

The value is in user-units.

-y *float*}

If no anchor point is specified then this option specifies the explicit y coordinate. If an anchor point is specified then this option specifies the y offset from the y coordinate of the anchor point.

The resultant y coordinate specifies the position of the horizontal line to flip the objects about.

The value is in user-units.

-flip_transform

Specifies that the transform of the object (if it has one) is flipped as well as the position.

By default the transform is not flipped.

`-ignore_fixed`

Normally fixed objects will not be flipped. If this flag is supplied then fixed objects will also be flipped.

DESCRIPTION

If the `-x` option and `-y` option are not specified then this command flips the position of the supplied objects by swapping the corresponding anchor point of each object (See 'Swapping Objects' section below).

If the `-x` option is specified then the objects are flipped about the vertical line at the specified x coordinate (See 'Flipping at X' section below).

If the `-y` option is specified then the objects are flipped about the horizontal line at the specified y coordinate (See 'Flipping at Y' section below).

Flipping is essential a move operation. For details on which objects are moveable see **`get_edit_property(2)`**.

Swapping Objects

To swap objects, the objects are first sorted left->right or top->bottom depending on whether the bounding box of all objects is larger horizontally (sort left->right) or vertically (sort top->bottom).

The objects are swapped using the following algorithm:

1. get object at the start of the object list (object 1)
2. get object at end of the list (object 2)
3. swap object anchor positions (object 1 is moved to object 2's anchor position and object 2 is moved to object 1's anchor position).
4. remove the two selected objects from the list
5. repeat until list has less than 2 entries

i.e. for objects A B C D E, A will be swapped with E and B with D. C will not be touched.

If the object has a transform and the `-flip_transform` option is specified then the transform is also mirrored in the X direction (sort left->right) or Y direction (sort top->bottom).

This operation is useful with sets of objects of the same size and type (e.g. pins or hard macros) where the order needs to be reversed.

Flipping at X

To flip objects about an x coordinate, the x values of the boundaries of each object are mirrored about the x value of the vertical line specified by the `-x` and `-anchor` options. The object then has its boundary set to the new value.

If the object has a transform and the *-flip_transform* option is specified then the transform is also mirrored in the X direction.

This operation is useful when swapping the order of a set of standard cells on a horizontal row.

Flipping at Y

To flip objects about an y coordinate, the y values of the boundaries of each object are mirrored about the y value of the vertical line specified by the *-y* and *-anchor* options. The object then has its boundary set to the new value.

If the object has a transform and the *-flip_transform* option is specified then the transform is also mirrored in the Y direction.

This operation is useful when swapping the transform of a set of standard cells on a horizontal row.

NOTE

Snapping is done automatically by using global snap settings.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example swaps the lower-left corner of each selected object with its corresponding selected object.

```
rompt> flip_objects [get_selected] -anchor ll
```

The following example flips position and transform of the selected objects about the center of the objects' surrounding bounding box.

```
rompt> flip_objects [get_selected] -x 0 -anchor center -flip_transform
```

The following example flips the selected objects about the horizontal line at y=100.

```
rompt> flip_objects [get_selected] -y 100
```

SEE ALSO

```
align_objects(2)
distribute_objects(2)
expand_objects(2)
get_object_snap_type(2)
move_objects(2)
remove_objects(2)
resize_objects(2)
rotate_objects(2)
```

```
set_object_snap_type(2)
```

focal_opt

Performs postroute optimization to fix setup, hold, or logical design rule constraint (DRC) violations on the design. The selected optimization is referred to as the *focal metric*.

SYNTAX

```
status focal_opt
-setup_endpoints all | file_name
    | -hold_endpoints all | file_name
    | -drc_nets all | file_name
    | -drc_pins all | file_name
[-effort medium | high]
[-size_only_mode density | in_place | footprint]
[-prioritize]
```

ARGUMENTS

-setup_endpoints all | file_name
Optimizes the specified endpoints by using setup time as the focal metric. To optimize all endpoints with setup violations, specify **all**. To optimize specific endpoints, specify the name of a file that contains the list of endpoints. The format for defining the endpoints in the file is described in the FILE FORMATS section of this man page. The tool will automatically set the slack value of the specified endpoint with the value specified in the given file. This feature can be turned off by using the **focalopt_endpoint_margin** variable.
The **-setup_endpoints**, **-hold_endpoints**, **-drc_pins** and **-drc_nets** options are mutually exclusive. You must specify one, and only one, of these options.

-hold_endpoints all | file_name
Optimizes the specified endpoints by using hold time as the focal metric. To optimize all endpoints with hold violations, specify **all**. To optimize specific endpoints, specify the name of a file that contains the list of endpoints. The format for defining the endpoints in the file is described in the FILE FORMATS section of this man page. The tool will automatically set the slack value of the specified endpoint with the value specified in the given file. This feature can be turned off by using the **focalopt_endpoint_margin** variable.
The **-setup_endpoints**, **-hold_endpoints**, and **-drc_nets** options are mutually exclusive. You must specify one, and only one, of these options.

-drc_nets all | file_name
Optimizes the specified nets by using logical DRCs as the focal metric.

-drc_pins all | file_name
Optimizes the specified endpoints by using logical DRCs as the focal metric. To optimize all nets with logical DRC violations, specify **all**. To optimize specific nets, specify the name of a file that contains the list of nets. The format for defining the nets in the file is described in the FILE FORMATS section of this man page.
The **-setup_endpoints**, **-hold_endpoints**, and **-drc_nets** options are mutually

exclusive. You must specify one, and only one, of these options.

-effort medium | high

Specifies the effort level for postroute optimizations in the **focal_opt** flow. Valid values are

- * **medium** (the default)

When you specify **medium** effort, more solution spaces are explored while fixing the violations, including solutions that exceed the density limit.

- * **high**

When you specify **high** effort, even more solution spaces are explored, which requires more CPU time. In addition, optimization uses runtime-expensive accurate signal integrity re-estimation, which is not required for all designs.

You should use **high** effort only when you have a few remaining violations to close.

These aggressive approaches are employed in the medium and high effort levels to enable **focal_opt** to achieve good results; however, the aggressive approaches can result in convergence issues.

Both effort levels honor **set_cost_priority** and **set_fix_hold_options**.

-size_only_mode density | in_place | footprint

Restricts the topology-based optimizations to cell sizing.

You can use this option for designs very sensitive to postroute optimization changes. You can control the sizing effort by using the different sizing modes:

- * **density**

Use regular density-based sizing.

- * **in_place**

Use in-place sizing.

- * **footprint**

Use footprint-preservation sizing.

-prioritize

Performs an extra optimization iteration in which the cost priority changes to give the focal metric the highest priority. The **-prioritize** switch will be ignored if user has redefined cost priority by using **set_cost_priority** command..

DESCRIPTION

The **focal_opt** command fixes either setup, hold, or logical DRC violations in the design by performing topology-based optimization. You must select one metric to use for each **focal_opt** run.

The optimization techniques used in **focal_opt** are more expensive and aggressive than those used in **route_opt**. The **focal_opt** command is best used for fixing the few violations remaining after running **route_opt** and **route_opt -incremental**. It should not be used as a replacement for **route_opt** and **route_opt -incremental** because it focuses on only one metric in each iteration.

FILE FORMAT

Endpoint File

The **-setup_endpoints**, **-hold_endpoints** **-drc_pins** options use a file that specifies the endpoints to optimize.

For each endpoint, you can specify either only the endpoint or the endpoint with a slack value, as shown below. The best way to generate these lines is by editing the file generated by the **report_constraint** command.

A. Only the endpoint

```
I_STACK_TOP/I3_STACK_MEM/Stack_Mem_reg_2__1_/D
```

B. Endpoint with slack

```
I_STACK_TOP/I3_STACK_MEM/Stack_Mem_reg_2__1_/D      -0.14 (VIOLATED)
I_STACK_TOP/I3_STACK_MEM/Stack_Mem_reg_2__1_/D  0.58          0.44 r      -
0.14 (VIOLATED)
```

C. Endpoint with slack for MCMM designs

```
I_STACK_TOP/I3_STACK_MEM/Stack_Mem_reg_2__1_/D      -0.14 (VIOLATED)  SCENARIO1
_I_STACK_TOP/I3_STACK_MEM/Stack_Mem_reg_2__1_/D  0.58          0.44 r      -
0.14 (VIOLATED)  SCENARIO1
```

Net File

The **-drc_nets** option use a file that specifies the nets to optimize.

For each net, you can specify either only the net name or the net name with the violation, as shown below. The best way to generate these lines is by editing the file generated by the **report_constraint** command.

A. Only the net name

```
I_STACK_TOP/n342
```

B. Net name with violation

```
I_STACK_TOP/n342          0.70          0.89          -0.19 (VIOLATED)
```

EXAMPLES

The following example runs **focal_opt** to fix the setup violations.

```
prompt> focal_opt -setup_endpoints all
```

The following example runs **focal_opt** to fix the subset of hold violations specified in the hold_vio.txt file with hold as the highest priority cost.

```
prompt> focal_opt -hold_endpoints hold_vio.txt -prioritize
```

SEE ALSO

report_constraint(2)

```
route_opt(2)
set_route_opt_strategy(2)
set_cost_priority(2)
routeopt_density_limit(3)
focalopt_endpoint_margin(3)
```

foreach_in_collection

Iterates over the elements of a collection.

SYNTAX

```
string foreach_in_collection  
itr_var  
collections  
body
```

Data Types

<i>itr_var</i>	collection
<i>collections</i>	list
<i>body</i>	string

ARGUMENTS

<i>itr_var</i>	Specifies the name of the iterator variable.
<i>collections</i>	Specifies a list of collections over which to iterate.
<i>body</i>	Specifies a script to execute per iteration.

DESCRIPTION

This command is used to iterate over each element in a collection. You cannot use the Tcl-supplied **foreach** command to iterate over collections, because **foreach** requires a list, and a collection is not a list. Using **foreach** on a collection would cause the collection to be deleted.

The arguments to **foreach_in_collection** parallel those of **foreach**: an iterator variable, the collections over which to iterate, and the script to apply at each iteration. Note that **foreach_in_collection** does not allow a list of iterator variables.

During each iteration, the *itr_var* option is set to a collection of exactly one object. Any command that accepts *collections* as an argument accepts *itr_var*, because they are of the same data type (collection).

You can nest the **foreach_in_collection** command within other control structures, including **foreach_in_collection**.

Note that if the body of the iteration is modifying the netlist, it is possible that all or part of the collection involved in the iteration could be deleted. The **foreach_in_collection** command is safe for such operations. If a command in the body causes a collection to be removed, at the next iteration, the iteration ends with a message indicating that the iteration ended prematurely.

An alternative to collection iteration is to use complex filtering to create a collection that includes only the desired elements, and then apply one or more commands to that collection. If the order of operations does not matter, the following examples are equivalent. The first is an example without iterators.

```
prompt> set s [get_cells {U1/*}]\
command1 $s\
command2 $s\
unset s
```

The following example is equivalent to the example above, only it uses the **foreach_in_collection** command.

```
prompt> foreach_in_collection itr [get_cells {U1/*}] {\
command1 $itr\
command2 $itr\
}
```

For collections with large numbers of objects, the non-iterator version is more efficient, although both produce the same results if the commands are order-independent.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes the wire load model from all hierarchical cells in the current instance.

```
prompt> foreach_in_collection itr [get_cells *] {\
if {[get_attribute $itr is_hierarchical] == "true"} {\
remove_wire_load_model $itr\
}\
}
```

```
Removing wire load model from cell 'i1'.
Removing wire load model from cell 'i2'.
```

SEE ALSO

```
collections(2)
filter_collection(2)
query_objects(2)
```


format_lib

Converts one or more input libraries to equivalent output libraries.

SYNTAX

```
status format_lib -f file_name
```

Data Types

file_name string

ARGUMENTS

-f *file_name*
 Specifies a command file *file_name*.

DESCRIPTION

This command converts one or more input libraries to equivalent output libraries, according to commands and options in the command file specified by *file_name*.

Command-File Syntax

The various settings and options that are required by the **format_lib** command are specified in the command-file using the simple syntax shown below:

```
set option_name value
```

where the *option_name* can be one of the following described in the sections below.

Compact CCS Options

```
set compact_ccs (true|false)  
      convert expand CCS library to compact CCS library. [false]
```

```
set input_library string  
      input library file name
```

```
set output_library string  
      output library file name
```

Expand CCS Options

```
set expand_ccs (true|false)  
      convert compact CCS library to expand CCS library. [false]
```

set input_library *string*
input library file name

set output_library *string*
output library file name

VA merge Options

set va_merge (true|false)
merge nominal library with several va_libraries into one Unified VA library.
[false]

set nominal_library <nomlib_name> <par1> <nval1> <par2> <nval2>... <parN> <nvalN>
set input nominal library file name, followed by va_parameter name and their nominal values. va_parameter and nominal values must appear in pair. No default value for any va_parameters.

set output_library *string*
output library file name

set va_library_list <lib1> <var11> <val21> ... <valN1> <lib2> <var12> <var22> ... <varN2>...
set va_libraries, with va_values for each va_parameters followed each library name. The order of va_values must be exactly same with the order of va_parameters and va_nominal_values in "set nominal_library" command.

set slew_indexes <n1> <n2> ... <nx>
slew indexes picking. If specified, data corresponding to selected indexes in va_libraries will be output into Unified VA library. [all]

set load_indexes <n1> <n2> ... <nx>
load indexes picking. If specified, data corresponding to selected indexes in va_libraries will be output into Unified VA library. [all]

CCS noise merge Options

set ccsn_merge (true|false)
merge one CCS timing library and one CCS noise library into one. [false]

set timing_library *string*
timing library file name

set noise_library *string*
noise library file name

set output_library *string*
output library file name

CCS to ECSM Options

set ccs2ecsm (true|false)
convert CCS library to ECSM library. If more than one input library provided,

it will scale these libraries to target PVT first then convert to an ECSM library. If indexes (slew/load) of one specific arc among different libraries in library list are different, ccs2ecsm will always use those in first library in list as desired indexes. [**false**]

set library_list <lib1> <lib2> ... <libN>
input library list

set process value
target process for scaling

set voltage value
target voltage for scaling

set temperature value
target temperature for scaling

set output_library string
output library file name

set capacitance_mode (first|second|ave|min|max)
set which method to use for conversion of CCS receiver capacitance to ECSM capacitance. First means select receiver_cap1; second means receiver_cap2; ave means average, i.e., (receiver_cap1+receiver_cap2)/2.0; min means minimum of receiver_cap1 and receiver_cap2; max means maximum of receiver_cap1 and receiver_cap2. [first].

set sample <vs1> <vs2>...<vsN>
desired voltage sample points. [0.05 0.10 0.20 0.30 0.40 0.50 0.60 0.70 0.80 0.90 0.95].

CCS to NLDM Options

set ccsldm (true|false)
convert CCS library to NLDM library. If more than one input library provided, it will scale these libraries to target PVT first then convert to an NLDM library. If indexes (slew/load) of one specific arc among different libraries in library list are different, ccs2nldm will always use those in first library in list as desired indexes. [**false**]

set library_list <lib1> <lib2> ... <libN>
input library list

set process value
target process for scaling

set voltage value
target voltage for scaling

set temperature value
target temperature for scaling

set output_library string
output library file name

EXAMPLES

The following example shows a command file called `f.cmd` and its usage:

```
set compact_ccs true
set input_library in.lib
set output_library out.lib
prompt> format_lib -f f.cmd
```


generate_qtm_model

Generate Quick Timing Model (QTM) from design CEL view.

SYNTAX

```
string generate_qtm_model  
[-block name_list]  
-clock_definitions filename_or_cellname_filename_pairs  
[-directory directory_name]
```

Data Types

<i>directory_name</i>	string

ARGUMENTS

`-block name_list`
Specify the list of names for the soft macros in current design that you want to create Quick Timing Model for. This is an optional option. By default, The Quick Timing Model for the current design is created.

`-clock_definitions filename_or_cellname_filename_pairs`
Specify the SDC file(s) which contain clock information for the design or the soft macros. When creating QTM for the current design, use this option without the option `-block`, and provide a SDC file name with this option. When creating QTMs for the soft macros inside current design, use this option together with the option `-block`, and specify a list of cellname-filename pairs with this option to provide one SDC file for each soft macro cell. This is a required option.

`-directory directory_name`
Specify the directory where QTM files will be written to. This is an optional option. By default, the files are written to the current working directory.

DESCRIPTION

This command creates Quick Timing Model (QTM) for the current design or the soft macro cells inside the current design. The QTMs are created based on the clock information provided in the given SDC files, and the delay information in the QTMs are based on the real delays in the design.

EXAMPLES

The following example creates QTM for current design, and the QTM file is written to directory my qtm dir.

```
prompt> generate_gtm_model -clock_definition clk.sdc -directory my_gtm_dir
```

The following command creates QTMs for soft macros A1 and A2 in current design.

```
prompt> generate_qtm_model -block {A1 A2} -  
clock_definition {A1 clk1.sdc A2 clk2.sdc}ffP
```

SEE ALSO

```
create_qtm_model(2)  
report_qtm_model(2)  
save_qtm_model(2)  
write_qtm_model(2)
```

get_adjusted_endpoints

Creates a collection of endpoints that have been adjusted in feasibility mode.

SYNTAX

```
status get_adjusted_endpoints
[-zero_path]
[-zero_wire_load]
[-io]
[-slack_threshold]
[-all]
[-scenarios scenarios]
```

Data Types

scenarios list

ARGUMENTS

- zero_path**
This option returns only endpoints that have been adjusted for zero_path violations, where zero_path margins have been added to the endpoint required times.
- zero_wire_load**
This option returns only endpoints that have been adjusted for zero_wire_load violations, where zero_wire_load margins have been added to the endpoint required times.
- io**
This option returns only endpoints that have been adjusted due to I/O margins that were added to endpoint required times.
- slack_threshold**
This option returns only endpoints that have been adjusted for slack_threshold, where slack threshold margins have been added to the endpoint required times.
- all**
This option returns all adjusted endpoints irrespective of the reasons for adjustment.
- scenarios *scenarios***
This option returns only adjusted endpoints of the given scenarios. The default is to return only adjusted endpoints of the current scenarios.

DESCRIPTION

Feasibility mode enables constraint relaxation for unachievable endpoints for reporting commands and the **place_opt** command. During feasibility mode unachievable endpoints are identified and required time on the unachievable endpoints are relaxed

based on the unachievable timing portion. Endpoints with zero path violations and zero wire load violations are considered unachievable endpoints.

This command allows you to obtain in a collection the list of endpoints that have been adjusted for the given specific reasons and scenarios. The default is to return only the adjusted endpoints of the current scenario.

Multicorner-Multimode Support

By default, this command uses information from the current scenario. You can select different scenarios by using the **-scenarios** option.

EXAMPLES

The following examples show how to use the **get_adjusted_endpoints** command.

```
prompt> get_adjusted_endpoints -zero_path -scenarios s1  
{Top/B_A/B/C1/a Top/B_A/C2/b Top/C_A/C2/b}
```

```
prompt> sizeof_collection \  
[get_adjusted_endpoints -zero_path -scenario s1]  
3
```

SEE ALSO

```
place_opt(2)  
set_feasibility_options(2)  
report_feasibility_options(2)
```

get_alternative_lib_cells

Creates a collection of equivalent library cells from loaded libraries, for a given cell or library cell. This collection can be used to replace or resize a specified cell in the current design. You can assign these library cells to a variable or pass them into another command.

SYNTAX

```
string get_alternative_lib_cells  
[-quiet]  
[-regexp [-nocase]]  
[-exact]  
[-filter expression]  
[-library libraries]  
pattern_or_objects
```

Data Types

<i>expression</i>	string
<i>libraries</i>	string
<i>pattern_or_objects</i>	string

ARGUMENTS

- quiet
Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.
- regexp
Uses the value of the *patterns_or_objects* argument as a real regular expression rather than as a simple wildcard pattern.
- nocase
Makes matches case-insensitive. You can use the **-nocase** option only when you also use the **-regexp** option.
- exact
Disables simple pattern matching. Use this option when searching for objects that contain the "*" (asterisk) wildcard as an actual character.
- filter *expression*
Filters the collection with the value of the *expression* argument. For library cells that match the specified patterns (or objects), the expression is evaluated based on the attributes of the library cells. If the expression evaluates to true, the library cell is included in the result. The **-regexp** and **-nocase** options are applied in the same way as in the **filter_collection** command. See the **filter_collection** man page for more information on how to use a **-filter** option in general.
- library *libraries*
Specifies a list of libraries from which to select equivalent library cells. The default is to select from all libraries in the link path. In this case,

lib_spec can be a list of library names or a collection of loaded libraries.

`pattern_or_objects`

Specifies cells or library cells for which a collection of equivalent library cells is required. Pattern matching is case sensitive unless you use the **-nocase** option. The default is * (asterisk).

DESCRIPTION

This command creates a collection of equivalent library cells from loaded libraries loaded for a given cell or library cell. The command returns a collection handle (identifier) if any library cells that are logically similar to the specified library cell match the *patterns_or_objects* option (if specified) and pass the filter (if specified). If no objects match the criteria, the command returns an empty string.

When the **-libraries** option is omitted, the command traverses the link path to find libraries. In this case, if a library is found with a min library relationship (which is determined from the setting in the **set_min_library** command), the min library is automatically included. When the **-libraries** option is used, the command considers only the libraries that are passed in.

If the command is unable to find any logically similar library cells for the library cell and the current design is not linked, the design automatically links.

Regular expression matching in the **get_alternative_lib_cells** command is the same as in the Tcl **regexp** command. When using **-regexp**, take be careful how you quote the *patterns_or_objects* option and filter *expression*. It is recommended that you use rigid quotation marks with curly braces around regular expressions.

Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search simply by adding *.** to the beginning or end of the expressions, as needed.

You can use the **get_alternative_lib_cells** command at the command prompt, or you can nest it as an argument to another command, such as **query_objects**. In addition, you can assign the **get_alternative_lib_cells** result to a variable.

When issued from the command prompt, **get_alternative_lib_cells** behaves as if you had called **query_objects** to display the objects in the collection. By default, the command displays a maximum of 100 objects. You can change this maximum by using the **collection_result_display_limit** variable.

The "implicit query" property of **get_alternative_lib_cells** provides a fast, simple way to display cells in a collection. However, if you want the flexibility provided by the options of the **query_objects** command (for example, if you want to display the object class), use **get_alternative_lib_cells** as an argument to **query_objects**.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES for XG Mode

The following example shows that, given a standard library cell, you can query the equivalent library cells.

```
prompt> set libcelsel\  
[get_lib_cells slow/AND2X2]  
{ "slow/AND2X2" }  
prompt> query_objects \  
[get_alternative_lib_cells $libcelsel]  
{ "slow/AND2X1", "slow/AND2X6", "slow/AND2X12" }
```

The following example shows that, given a cell, you can query the equivalent library cells that can be used in place of those cells.

```
prompt> set celsel [get_cell top/U21]  
{"top/U21"}  
prompt> query_objects \  
[get_alternative_lib_cells $celsel]  
{ "slow/AND2X1", "slow/AND2X2", "slow/AND2X6", "slow/AND2X12" }
```

SEE ALSO

```
collections(2)  
filter_collection(2)  
get_cells(2)  
get_lib_cells(2)  
query_objects(2)  
set_min_library(2)  
collection_result_display_limit(3)  
wildcards(3)
```

get_always_on_logic

Returns a collection of cells and nets on always-on paths in the design.

SYNTAX

```
collection get_always_on_logic  
[-cells]  
[-nets]  
[-all]
```

ARGUMENTS

-cells
Returns the cells on always-on paths.

-nets
Returns the nets on always-on paths.

-all
Returns the nets and cells on always-on paths. This is the default when no argument is specified.

DESCRIPTION

This command creates a collection of nets and cells on always-on paths. Always-on paths are paths that must be powered up the entire time that the system is operating. These paths consist of logic cones that terminate on always-on pins in the design. Examples of always-on pins are the enable pin of the isolation cell and special control pins of retention registers.

You can mark any instance pin with an always-on attribute by using the **set_attribute** command. You can create power down regions by using the **create_power_domain** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

SEE ALSO

```
create_power_domain(2)  
set_attribute(2)
```


get_attribute

Returns the value of an attribute on multiple objects.

SYNTAX

```
string get_attribute  
[-class class_name]  
[-quiet]  
object_spec  
attribute_name
```

Data Types

<i>class_name</i>	string
<i>object_spec</i>	string
<i>attribute_name</i>	string

ARGUMENTS

-class *class_name*
Specifies the class name of the *object_spec* option when *object_spec* is not a collection. Valid classes are **design**, **port**, **cell**, **net**, and so on. This option is required if *object_spec* is not a collection.

-quiet
Indicates that error and warning messages are not to be reported.

object_spec
Specifies objects from which to get the attribute value. The *object_spec* must be either a collection or names that is combined with the *class_name* to find the object. If *object_spec* is not a collection, you must also use the **-class** option.

attribute_name
Specifies the name of the attribute whose value is to be returned.

DESCRIPTION

This command returns the values of an attribute on objects. If the data type of the attribute is collection, the attribute values will be merged into one collection. Otherwise, a list of attribute values will be returned. If attribute of any objects cannot be got, a list of remaining available attribute values will be returned.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example shows that the first command defines an X attribute for cells:

```
prompt> define_user_attribute -type int -class cell X
Info:User-defined attribute 'X' on class 'cell'.
1
```

The following example sets the attribute to a value on all cells in this level of the hierarchy:

```
prompt> set_attribute [get_cells *] X 30
{"i1", "i2"}
```

The following example retrieves the value from one cell, combines it with the application attribute *full_name*, and creates a simple report:

```
prompt> foreach_in_collection sel [get_cells *] {\
echo -n "On '[get_attribute $sel full_name]'", "
echo "X = [get_attribute $sel X]"
}
On 'i1', X = 30
On 'i2', X = 30
```

The following example retrieves the attribute values from two objects, and the data type of attribute *preserved_cells* is collection.

```
prompt> get_attribute [get_mw_cels test1 test2] preserved_cells
{CORE REG_BLK STACK_BLK UPC_BLK}
```

The following example retrieves the attribute values from two objects, and the data type of attribute *preserved_cells* is collection.

```
prompt> get_attribute [get_cells test1 test2] allowable_orientation
{N FN FS S FW W E FE} {N FN FS SFW W E FE}
```

SEE ALSO

```
collections(2)
define_user_attribute(2)
foreach_in_collection(2)
list_attributes(2)
remove_attribute(2)
set_attribute(2)
```

get_bounds

Creates a collection of bounds from the current design.

SYNTAX

```
collection get_bounds
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-filter expression]
patterns | -of_objects objects
```

Data Types

<i>expression</i>	string
<i>patterns</i>	list
<i>objects</i>	list

ARGUMENTS

- quiet**
Suppresses warning and error messages, if no objects match. Does not suppress syntax error messages.
- regexp**
Views the *patterns* argument as a real regular expression rather than a simple wildcard pattern. This option also modifies the behavior of the =~ and !~ filter operators to behave as regular expressions rather than simple wildcard patterns.
- nocase**
Makes matches case-insensitive. If you use this option, you must also use the **-regexp** option.
- exact**
Considers wildcards to be plain characters, and does not interpret their meaning as wildcards.
- filter *expression***
Filters the collection with *expression*. The expression is evaluated based on the bound's attributes for any bounds that match the patterns option. If the expression evaluates to **true**, the bound is included in the result.
- patterns**
Matches bound names against *patterns*. If you do not use this option, the command uses the asterisk (*) as the default pattern. The *patterns* and **-of_objects** options are mutually exclusive; you can use only one.
- of_objects *objects***
Creates a collection of bounds containing the specified objects. The *objects* list should be a list of leaf cells or ports. The *patterns* and **-of_objects**

options are mutually exclusive; you can use only one.

DESCRIPTION

This command creates a collection of bounds from the current design that match certain criteria. The command returns a collection handle (identifier) if any bounds match the *patterns* value and passes the filter, if specified. If no objects match the criteria, the command returns an empty string.

You can use this command at the command prompt, or you can nest it as an argument to another command (for example, to the **report_bound** command). In addition, you can assign the **get_bounds** result to a variable.

When issued from the command prompt, the **get_bounds** command behaves as though you have called the **report_bound** command to report the objects in the collection. By default, it displays a maximum of 100 objects. You can change this maximum by using the **collection_result_display_limit** variable.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following examples get all bounds that have a name starting with **my_bound**.

```
prompt> get_bounds my_bound*
```

SEE ALSO

```
collections(2)
foreach_in_collection(2)
query_objects(2)
collection_result_display_limit(3)
```

get_buffers

Creates a collection of buffer cells from the libraries loaded in memory.

SYNTAX

```
collection get_buffers
[-filter expression]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-inverter]
[-inverting_buffers]
[-library lib_spec]
patterns
```

Data Types

<i>expression</i>	string
<i>lib_spec</i>	list
<i>patterns</i>	list

ARGUMENTS

- filter *expression***
Filters the collection with *expression*. For any buffer cells that match *patterns*, the expression is evaluated based on the buffer lib cell's attributes. If the expression evaluates to true, the buffer cell is included in the result. This option works the same as the **filter** command in dc_shell.
- quiet**
Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.
- regexp**
Uses the *patterns* argument as real regular expressions rather than simple wildcard patterns.
- nocase**
Makes matches case-insensitive.
- exact**
Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters.
- inverter**
This option specifies to get the inverter cells. By default only buffer cells will be returned. To include the library cells having both inverter and non-inverting outputs use **-inverting_buffers** option.
- inverting_buffers**
This option specifies to include inverting buffer cells along with the

inverter cells or buffer cells. By default this is false and only buffer cells will be returned (only inverter cells when `-inverter` is specified)
Inverting buffers are library cells which have both inverting and non-inverting outputs.

`-library lib_spec`

Specifies a list of libraries from which the buffer cells are to be collected. The default is to select from all libraries in the link path. In this case, `lib_spec` can be a list of library names, or collection of libraries loaded.

`patterns`

Matches buffer cell names against patterns. Patterns can include the wildcard characters `*` (asterisk) and `?` (question mark). For more details about using and escaping wildcards refer to the **wildcards** man page.

DESCRIPTION

The **get_buffers** command creates a collection of buffer cells from libraries currently loaded into `dc_shell-xg-t` that match certain criteria. The command returns a collection handle (identifier) if any buffer cells match the *patterns* or *objects* and pass the filter (if specified). If no objects match the criteria, the empty string is returned.

Regular expression matching is the same as in the Tcl **regexp** command. When using **-regexp**, take care in the way you quote the *patterns* and filter *expression*. Using rigid quoting with curly braces around regular expressions is recommended. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search simply by adding `".*"` to the beginning or end of the expressions as needed.

You can use the **get_buffers** command at the command prompt, or you can nest it as an argument to another command, such as **query_objects**. In addition, you can assign the **get_buffers** result to a variable.

When issued from the command prompt, **get_buffers** behaves as though **query_objects** has been called to display the objects in the collection. By default, a maximum of 100 objects is displayed. You can change this maximum using the **collection_result_display_limit** variable.

The implicit query property of **get_buffers** provides a fast, simple way to display buffer cells in a collection. However, if you want the flexibility provided by the **query_objects** options (for example, if you want to display the object class), use **get_buffers** as an argument to **query_objects**.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example queries all buffer cells that are in the current library.

```
prompt> get_buffers  
{"slow/BUF2X2", "slow/BUF2X4", "slow/BUF2X12"}
```

The following example queries all inverter cells that are in the misc_cmos library

```
prompt> get_buffers -inverter -library misc_cmos  
{"misc_cmos/INV2X2", "misc_cmos/INV2X4", "misc_cmos/INVX12"}
```

SEE ALSO

```
collections(2)  
filter_collection(2)  
get_lib_cells(2)  
query_objects(2)  
collection_result_display_limit(3)  
wildcards(3)
```

get_cell_sites

Creates a collection of cell sites from the current design.

SYNTAX

```
collection get_cell_sites
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-filter expression]
patterns
```

Data Types

<i>expression</i>	string
<i>patterns</i>	list

ARGUMENTS

- quiet**
Suppresses warning and error messages, if no objects match. Does not suppress syntax error messages.
- regexp**
Views the *patterns* argument as a real regular expression rather than a simple wildcard pattern. This option also modifies the behavior of the =~ and !~ filter operators to behave as regular expressions rather than simple wildcard patterns.
- nocase**
When combined with -regexp, this option makes matches case-insensitive. You can use -nocase only when you also use -regexp.
- exact**
Disables all pattern matching.
- filter *expression***
Filters the collection with *expression*. For any cell site that matches the *patterns* argument, the *expression* is evaluated based on the cell site's attributes. If the expression evaluates to true, the cell site is included in the result.
- patterns***
Matches cell site names against the specified *patterns*. The *patterns* list can include the wildcard character "*".

DESCRIPTION

The **get_cell_sites** command creates a collection of cell sites from the current design that matches certain criteria. The command returns a collection handle

(identifier) if any cell site matches the patterns and passes the filter (if specified). If no objects match the criteria, an empty string is returned.

You can use the **get_cell_sites** command at the command prompt, or you can nest it as an argument to another command (for example, **remove_cell_sites**). In addition, you can assign the **get_cell_sites** result to a variable.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example gets all the cell sites in the current design.

```
prompt> get_cell_sites  
{CELL_SITE#3690}
```

SEE ALSO

`remove_cell_sites(2)`

get_cells

Creates a collection of cells that meet the selection criteria.

SYNTAX

```
collection get_cells
[-hierarchical]
[-quiet]
[-regexp [-nocase] | -exact]
[-filter expression]
[patterns
 | -of_objects objects
 | -object_id object_id
 | -within region
 | -intersect region
 | -touching region
 | -at point]
[-all]
[-design_id design_id]
[-hsc separator]
```

Data Types

<i>expression</i>	string
<i>patterns</i>	list
<i>objects</i>	list
<i>object_id</i>	integer
<i>region</i>	list
<i>point</i>	string
<i>design_id</i>	integer

ARGUMENTS

-hierarchical

Searches for cells level-by-level, relative to the current instance. The full name of the object at a particular level must match the patterns. For example, if there is a cell block1/adder, a hierarchical search finds it by using "adder".
By default, the search is not hierarchical.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Uses the *patterns* argument as real regular expressions rather than as simple wildcard patterns.
The **-regexp** and **-exact** options are mutually exclusive. You can specify only one of these options.

-nocase

Makes the matches case-insensitive.

You can use this option only when you also use the **-regexp** option.

-exact

Disables simple pattern matching. Use this when searching for objects that contain the * (asterisk) wildcard character.

-filter *expression*

Filters the collection with the value of *expression*. For any cells that match the specified criteria, the expression is evaluated based on the cell's attributes. To see the list of cell attributes that you can use in the expression, use the **list_attributes -application -class cell** command. If the expression evaluates to true, the cell is included in the result.

patterns

Matches cell names against the specified patterns. Patterns can include the wildcard characters * (asterisk) and ? (question mark). For more details about using and escaping wildcards, see the **wildcards** man page. The *patterns*, **-of_objects**, **-object_id**, **-within**, **-intersect**, **-touching**, and **-at** arguments are mutually exclusive; you can specify only one. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

-of_objects *objects*

Creates a collection of cells connected to the specified objects. Each object can be a pin, net, bound, voltage area, plan group, or pin shape. The *patterns*, **-of_objects**, **-object_id**, **-within**, **-intersect**, **-touching**, and **-at** arguments are mutually exclusive; you can specify only one. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*. In addition, you cannot use the **-hierarchical** option with the **-of_objects** option.

-object_id *object_id*

Specifies the Milkyway object ID of the cell to include in the collection. To determine the *object_id* value, use the **get_attribute** command to return the value of the **object_id** attribute. The *patterns*, **-of_objects**, **-object_id**, **-within**, **-intersect**, **-touching**, and **-at** options are mutually exclusive; you can specify only one. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

-within *region*

Creates a collection containing all cells within the specified region. The region boundary can be rectangle or polygon. The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} or {{*llx lly urx ury*}}, which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically it is microns.

The *patterns*, **-of_objects**, **-object_id**, **-within**, **-intersect**, **-touching**, and **-at** options are mutually exclusive; you can specify only one. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern* argument.

In addition, you cannot use the **-hierarchical** option with the **-within** option.

-intersect *region*

Creates a collection containing all cells that intersect the specified region. The region boundary can be rectangle or polygon. The format for specifying a rectangle is `{{llx lly} {urx ury}}` or `{llx lly urx ury}`, which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is `{{x1 y1} {x2 y2} ... {xN yN} {x1 y1}}`, where each `{x y}` pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically it is microns.

The *patterns*, **-of_objects**, **-object_id**, **-within**, **-intersect**, **-touching**, and **-at** options are mutually exclusive; you can specify only one. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*. In addition, you cannot use the **-hierarchical** option with the **-intersect** option.

-touching *region*

Creates a collection containing all cells that touch the specified region. The region boundary can be rectangle or polygon. The format for specifying a rectangle is `{{llx lly} {urx ury}}` or `{llx lly urx ury}`, which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is `{{x1 y1} {x2 y2} ... {xN yN} {x1 y1}}`, where each `{x y}` pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically it is microns.

The *patterns*, **-of_objects**, **-object_id**, **-within**, **-intersect**, **-touching**, and **-at** options are mutually exclusive; you can specify only one. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*. In addition, you cannot use the **-hierarchical** option with the **-touching** option.

-at *point*

Creates a collection containing all cells at the specified point. The format for specifying a point is `{x y}`.

The coordinate unit is specified in the technology file; typically it is microns.

The *patterns*, **-of_objects**, **-object_id**, **-within**, **-intersect**, **-touching**, and **-at** options are mutually exclusive; you can specify only one. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*. In addition, you cannot use the **-hierarchical** option with the **-at** option.

-all

Includes physical-only cells in the collection. The following types of cells are considered physical-only cells:

- Standard cell filler
- Pad filler
- Corner
- Flip-chip pad (bump)
- Chip
- Cover
- Tap

Cells containing only PG ports
By default, physical-only cells are not included in the collection.

-design_id *design_id*

Specifies the design from which to create the collection of cells. The design is one of the currently open Milkyway designs. To determine the *design_id* value, use the **get_attribute** command to return the value of the **design_id** attribute. Use this option only when multiple Milkyway designs are open. By default, the command creates a collection of cells from the current design.

-hsc *separator*

Specifies the hierarchy separator character that is used by this command to create unambiguous names when an object name contains a slash (/). The valid values are / (slash), @ (at sign), ^ (caret), # (pound sign), . (period), and | (vertical bar). The default is /.

DESCRIPTION

This command creates a collection of cells that match the specified criteria. By default, the command creates the collection of cells from the current design, relative to the current instance.

The command returns a collection of cells if any cells match the *patterns*, *objects*, or *object_id* argument and pass the filter, if specified. If no objects match the criteria, an empty string is returned.

If the *patterns*, *objects*, and *object_id* arguments fail to match any objects and the current design is not linked, the design automatically links.

Regular expression matching is the same as in the Tcl **regexp** command.

When using the **-regexp** option, be careful about the way you quote the *patterns* and filter *expression* arguments; use rigid quoting with curly braces around regular expressions. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding *.** to the beginning or end of the expressions as needed.

When using the **-within**, **-touching**, or **-intersect** option, note that the behavior differs depending on the format of the specified region. A cell has both a boundary and a bounding box (bbox). In some cases, these attributes differ. When you specify a region in rectangle format, the tool searches cells by their bounding box. When you specify a region in polygon format, the tool searches cells by their boundary. Searching by bounding box is faster than searching by boundary.

You can use the **get_cells** command at the command prompt or you can nest it as an argument to another command, such as **query_objects**. In addition, you can assign the **get_cells** result to a variable.

When issued from the command prompt, **get_cells** behaves as though **query_objects** has been called to display the objects in the collection. By default, a maximum of 100 objects is displayed. You can change this maximum by using the **collection_result_display_limit** variable.

The implicit query property of **get_cells** provides a fast, simple way to display the cells in a collection. However, if you want the flexibility provided by the **query_objects** options (for example, if you want to display the object class), use the **get_cells** command as an argument to the **query_objects** command.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example queries the cells that begin with "o" and reference an FD2 library cell. Although the output looks like a list, it is only a display.

```
prompt> get_cells "o*" -filter "@ref_name == FD2"
{o_reg1 o_reg2 o_reg3 o_reg4}
```

The following example shows that, given a collection of pins, you can query the cells connected to those pins:

```
prompt> set pinsel [get_pins o*/CP]
{o_reg1/CP o_reg2/CP}

prompt> query_objects [get_cells -of_objects $pinsel]
{o_reg1 o_reg2}
```

The following example shows that, given a collection of nets, you can query the cells connected to those nets:

```
prompt> set netsel [get_nets tmp]
{tmp}

prompt> query_objects [get_cells -of_objects $netsel]
{b c}
```

The following example gets all cells within the specified rectangle:

```
prompt> get_cells -within {{62.101 91.301} {74.124 100.127}}
{STACK_BLK/C370/U30 STACK_BLK/U53}
```

The following example gets all cells that intersect the specified polygon:

```
prompt> get_cells \
-intersect {{30 20} {50 20} {50 30} {40 30} {40 40} {30 40} {30 20}}
{STACK_BLK/MEM_reg_9__0_ STACK_BLK/MEM_reg_1__0_}
```

SEE ALSO

`collections(2)`
`filter_collection(2)`
`get_pins(2)`
`list_attributes(2)`
`query_objects(2)`
`win_select_objects(2)`
`get_net_shapes(2)`
`collection_result_display_limit(3)`
`wildcards(3)`

get_clocks

Creates a collection of clocks from the current design.

SYNTAX

```
collection get_clocks
[-quiet]
[-regexp]
[-nocase]
[-filter expression]
patterns
```

Data Types

<i>expression</i>	string
<i>patterns</i>	list

ARGUMENTS

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns.

-nocase

Makes matches case-insensitive when combined with **-regexp**. Use **-nocase** only when you use **-regexp**.

-filter *expression*

Filters the collection with *expression*. For any clocks that match *patterns*, the expression is evaluated based on the clock's attributes. If the expression evaluates to true, the clock is included in the result. This option works the same as the **filter** command in dc_shell.

patterns

Matches clock names against patterns. Patterns can include the * (asterisk) and ? (question mark) wildcard characters. For more details about using and escaping wildcards refer to the **wildcards** man page.

DESCRIPTION

The **get_clocks** command creates a collection of clocks in the current design that match certain criteria. The command returns a collection handle (identifier) if any clocks match the *patterns* and pass the filter (if specified). If no objects match the criteria, the empty string is returned.

You can use the **get_clocks** command at the command prompt, or you can nest it as an argument to another command (for example, **query_objects**). In addition, you can assign the **get_clocks** result to a variable.

When issued from the command prompt, **get_clocks** behaves as though the **query_objects** command has been called to display the objects in the collection. By default, a maximum of 100 objects is displayed. You can change this maximum using the **collection_result_display_limit** variable.

The "implicit query" property of **get_clocks** provides a fast, simple way to display clocks in a collection. However, if you want the flexibility provided by the **query_objects** options (for example, if you want to display the object class), use **get_clocks** as an argument to **query_objects**.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example applies the **set_max_time_borrow** command to all clocks in the design matching *PHI**:

```
prompt> set_max_time_borrow 0 [get_clocks "PHI*"]
```

The following example sets the **clock_list** variable to a collection of clocks that have a period of 15:

```
prompt> set clock_list [get_clocks * -filter "@period==15"]
```

SEE ALSO

```
all_clocks(2)
collections(2)
create_clock(2)
query_objects(2)
report_clock(2)
wildcards(3)
collection_result_display_limit(3)
```

get_command_option_values

Queries current/default option values.

SYNTAX

```
get_command_option_values  
[-default | -current]  
-command command_name
```

ARGUMENTS

```
-default  
    get default option values if available  
  
-current  
    get current option values if available  
  
-command command_name  
    get option values for this command
```

DESCRIPTION

This command attempts to query a default or current value for each option (of the command) which has default and/or current-value-tracking enabled. Details of how the option value is queried depend on whether one of the optional options -current or -default is specified (see below).

A "Tcl array set compatible" (possibly empty) list of option names and values will be returned as the Tcl result - the even numbered entries in the list are the names of options which were enabled for default-value-tracking or current-value-tracking enabled (and had at least one of these values set to a not-undefined value); each odd numbered entry in the list is the default or current value of the option-name preceding it in the list. Any options which were enabled for neither default-value-tracking nor current-value-tracking will be omitted from the output list; similarly options which were enabled for default-value-tracking or current-value-tracking, but for which no (not-undefined) default or current value is set, will be omitted from the result list.

If neither -current nor -default is specified, then the default behavior is: for each command option which has either default-value-tracking or current-value-tracking (or both) enabled, the value returned is: the current value is returned if current-value-tracking is enabled and a (not-undefined) current value has been set; otherwise the default value is returned if default-value-tracking is enabled and a (not-undefined) default value has been set; otherwise the name/value pair for the option is not included in the result list.

If -current is specified, the value returned for an option is the current value if current-value-tracking is enabled, and a (not-undefined) current value has been set; otherwise the name/value pair for the option is omitted from the result list.

If -default is specified, the value returned for an option is the default value if

default-value-tracking is enabled, and a (not-undefined) default value has been set; otherwise the name/value pair for the option is omitted from the result list.

The result list from `get_command_option_values` includes option values of both dash options and positional options (assuming that both kinds of options of a command have been enabled for value-tracking).

The command will "throw" a Tcl error in a variety of situations, such as if an invalid command name was passed in via `-command`.

EXAMPLES

```
prompt> foo -bar1 10 -bar2 20
1
prompt> get_command_option_values -command foo
-bar1 10 -bar2 20
```

SEE ALSO

get_core_area

Creates a collection containing the core area of the current design.

SYNTAX

collection **get_core_area**

ARGUMENTS

This command has no arguments.

DESCRIPTION

This command returns a collection containing the core area of the current design. If the core area is not defined, the command returns an empty string.

The core area is a box that contains all rows. It represents the placeable area for standard cells. The core area is typically smaller than the cell boundary. Pads, I/O pins, and top-level power and ground rings are typically found outside the core area. Standard cells, macros, and wire tracks are typically found inside the core area. There should be only one core area in a design.

The core_area object is a first class object, and it has multiple attributes, such as bbox, cell_id, name and so on. You can use **list_attributes** and **report_attribute** to browse attributes. You can use **get_attribute** and **lindex** to get coordinates of the current design die area.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example gets the core area of the current design.

```
prompt> get_core_area
{unit_core}
```

The following example shows how to list the attributes of the core_area class by using **list_attributes** command.

```
prompt> list_attributes -application -class core_area
```

The following example shows how to get the attribute values of the core_area object by using **report_attribute** command.

```
prompt> report_attribute -application -class core_area [get_core_area]
```

The following example shows how to get coordinates of the current design die area,by using **get_attribute** command.

```
prompt> get_attribute [get_core_area] bbox  
{0.000 0.000} {300.480 396.620}
```

SEE ALSO

```
get_die_area(2)  
get_attribute(2)  
list_attributes(2)
```

get_coupling_capacitors

Reports coupling capacitors for the given net.

SYNTAX

```
int get_coupling_capacitors  
[-min]  
net
```

ARGUMENTS

-min
Reports coupling capacitors for the given net for min.

net
Reports coupling capacitors for the given net.

DESCRIPTION

Reports coupling capacitances for each individual aggressor. For each coupling the information is given in the format {victim:subnode_number (victim_net_name) aggressor:subnode_number (aggressor_net_name) CC_value type_of_filter}

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example produces detailed coupling capacitances for the net n9589.

```
prompt> get_coupling_capacitors n9589
```

```
{n9589:7 (n9589) n8661:7 (n8661) 6.59743e-  
05 not_filtered} {n9589:7 (n9589) n8661:8 (n8661) 6.59743e-05 not_filtered}
```

SEE ALSO

report_noise(2)

get_cts_scenario

Returns the name of the clock tree synthesis scenario or the empty string if a clock tree synthesis scenario is not defined.

SYNTAX

string **get_cts_scenario**

ARGUMENTS

The **get_cts_scenario** command has no arguments.

DESCRIPTION

This command returns the name of the clock tree synthesis scenario or the empty string if a clock tree synthesis scenario is not defined.

Multicorner-Multimode Support

This command uses information from the clock tree synthesis scenario.

EXAMPLES

The following example uses **get_cts_scenario** to get the clock tree synthesis scenario.

```
prompt> create_scenario MODE1
prompt> set_cts_scenario MODE1
MODE1
prompt> get_cts_scenario
MODE1
```

SEE ALSO

```
remove_cts_scenario(2)
set_cts_scenario(2)
```

get_design_lib_path

Returns the directory to which the specified library is mapped.

SYNTAX

```
status get_design_lib_path  
library_name
```

Data Types

```
library_name      string
```

ARGUMENTS

```
library_name  
    Specifies the library whose directory mapping is returned.
```

DESCRIPTION

This command returns the directory to which the specified library is mapped.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example maps the library named MY_LIB to the same directory as the WORK library.

```
prompt> define_design_lib MY_LIB\  
-path get_design_lib_path("WORK")
```

SEE ALSO

```
read_file(2)  
report_design_lib(2)
```


get_designs

Creates a collection of one or more designs loaded into the tool.

SYNTAX

```
collection get_designs
[-hierarchical]
[-quiet]
[-regex]
[-nocase]
[-exact]
[-filter expression]
patterns
```

Data Types

<i>expression</i>	string
<i>patterns</i>	list

ARGUMENTS

-hierarchical

Searches for designs inferred by the design hierarchy relative to the current instance. The full name of the object at a particular level must match the patterns. The use of this option does not force an auto link.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regex

Uses the *patterns* argument as real regular expressions rather than simple wildcard patterns.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters.

-filter *expression*

Filters the collection with *expression*. For any designs that match *patterns*, the expression is evaluated based on the design's attributes. If the expression evaluates to true, the design is included in the result.

patterns

Matches design names against patterns. Patterns can include the wildcard characters * (asterisk) and ? (question mark). For more details about using and escaping wildcards refer to the **wildcards** man page.

DESCRIPTION

The **get_designs** command creates a collection of designs from those currently loaded into the tool that match certain criteria. The command returns a collection if any designs match the *patterns* and pass the filter (if specified). If no objects match your criteria, the empty string is returned.

Regular expression matching is the same as in the Tcl **regexp** command. When using **-regexp**, take care in the way you quote the *patterns* and *filter expression*. Using rigid quoting with curly braces around regular expressions is recommended. Regular expressions are always anchored. The expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search simply by adding *.** to the beginning or end of the expressions as needed.

You can use the **get_designs** command at the command prompt, or you can nest it as an argument to another command, such as **query_objects**. In addition, you can assign the **get_designs** result to a variable.

When issued from the command prompt, **get_designs** behaves as though **query_objects** has been called to display the objects in the collection. By default, a maximum of 100 objects is displayed. You can change this maximum using the **collection_result_display_limit** variable.

The implicit query property of **get_designs** provides a fast, simple way to display designs in a collection. However, if you want the flexibility provided by the **query_objects** options (for example, if you want to display the object class), use **get_designs** as an argument to **query_objects**.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example queries the designs that begin with "mpu". Although the output looks like a list, it is just a display. A complete listing of designs is available using the **list_designs** command.

```
prompt> get_designs mpu*
{mpu_0_0 mpu_0_1 mpu_1_0 mpu_1_1}
```

The following example shows that, given a collection of designs, you can remove those designs:

```
prompt> remove_design [get_designs mpu*]
Removing design mpu_0_0...
Removing design mpu_0_1...
```

```
Removing design mpu_1_0...
Removing design mpu_1_1...
```

SEE ALSO

```
collections(2)
filter_collection(2)
list_designs(2)
query_objects(2)
remove_design(2)
collection_result_display_limit(3)
wildcards(3)
```

get_die_area

Creates a collection containing the die area of the current design.

SYNTAX

```
collection get_die_area
```

ARGUMENTS

This command has no arguments.

DESCRIPTION

This command returns a collection containing the die area of the current design. If the die area is not defined, the command returns an empty string.

The die area is also known as the cell boundary. It represents the silicon boundary of a chip and typically encloses all objects of a design, such as pads, I/O pins, cells, and so on. There should be only one die area in a design.

The die area object is a first class object that has multiple attributes, such as `bbox`, `cell_id`, `name`, and so on. You can use the **list_attributes** and **report_attribute** commands to browse attributes. You can use the **get_attribute** command and **lindex** to get coordinates of the current design die area.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example gets the die area of the current design.

```
prompt> get_die_area  
{top_die}
```

The following example shows how to use the **list_attributes** command to list the attributes of the `die_area` class.

```
prompt> list_attributes -application -class die_area
```

The following example shows how to use the **report_attribute** command to get the attribute values of the `die_area` object.

```
prompt> report_attribute -application\  
-class die_area [get_die_area]
```

The following example shows how to use the **get_attribute** command to get coordinates of the current design die area.

```
prompt> get_attribute [get_die_area] bbox  
{0.000 0.000} {300.480 396.620}
```

SEE ALSO

```
get_attribute(2)  
get_core_area(2)  
list_attributes(2)
```

get_dominant_scenarios

Returns a list of dominant scenarios.

SYNTAX

```
list get_dominant_scenarios  
[-scenarios scenario_list]  
[-distributed]  
[-setup_distributed]  
[-run_distributed]  
[-process_distributed]
```

Data Types

scenario_list list

ARGUMENTS

- scenarios *scenario_list***
Lists scenarios, which might be active or inactive, to analyze. By default, the command analyzes all active scenarios.
- distributed**
Specifies for analysis to be done in distributed mode. You must use the **set_mcmm_job_options** and/or **set_host_options** commands to set various parameters necessary for distributed operation, such as the host list or compute farm.
- setup_distributed**
Generates the necessary directories, data files, and scripts for a distributed-mode run; however, it does not actually start any slave jobs, and does not do actual dominant-scenario analysis. You must use the **set_mcmm_job_options** command to set various parameters necessary to do this, such as the working directory.
- run_distributed**
Executes a set of distributed analysis jobs previously set up by specifying the **-setup_distributed** option; however, it does not do actual dominant-scenario analysis. You must use the **set_mcmm_job_options** command to set various parameters necessary to do this, such as the working directory.
- process_distributed**
Performs dominant-scenario analysis based on the results of a previous distributed-mode run. You must use the **set_mcmm_job_options** command to set various parameters, such as the working directory. You can use the **-process_distributed** option with the **-run_distributed** option to re-run and re-analyze a previously-created distributed-mode run.

DESCRIPTION

This command identifies a set of essential or dominant scenarios. These are a subset

of the user-defined scenarios that are considered essential for concurrent multicorner, multimode optimization of a design. These scenarios have the worst slack (timing, DRC, and so on) for at least one of the constrained objects in the design (that is, an end-point for delay constraints or a net for DRC constraints).

If any of these scenarios is not active during optimization, the final optimized design's quality of results (QoR) might degrade when measured across all user-defined scenarios. The analysis is based on the current state of the design in terms of its current placement, routing, and clock-tree state.

You can use the **get_dominant_scenarios** command in conjunction with the **set_active_scenario** command (or use -apply option in threaded or distributed mode) to direct the tool's concurrent multi-scenario optimization to work on a subset of scenarios. This can improve the memory and run-time capacity of concurrent multi-scenario optimization.

get_dominant_scenarios will run in multi-CPU mode if the maximum number of processing threads has been set by the **-max_cores** option of the **set_host_options** command. On multi-core or multi-CPU machines, activating threaded operation can greatly reduce runtime.

If the **-distributed** option is given, **get_dominant_scenarios** will run in distributed mode. You must use the **set_mcm_job_options** and/or **set_host_options** commands to set various parameters necessary for distributed operation, such as the host list or compute farm.

Multicorner-Multimode Support

This command uses information from both active and inactive scenarios.

This command also uses information from the clock tree synthesis scenario.

The set of active scenarios before and after this command can be different, this is to save runtime by avoiding unexpected scenario activation/deactivation. Use **all_active_scenarios** to save activation status and **set_active_scenario** to restore original active scenarios.

EXAMPLES

The following example uses the command to identify the dominant (or essential) subset of scenarios and sets them to be active prior to optimization.

```
prompt> create_scenario scenel
prompt> read_sdc scenel.sdc
prompt> create_scenario scene2
prompt> read_sdc scene2.sdc
prompt> # Create more scenarios and read their SDC...
prompt> create_scenario scene8
prompt> read_sdc scene8.sdc
prompt> set_active_scenarios [get_dominant_scenarios]
prompt> all_active_scenarios
prompt> psynopt -area_recovery -power
```

SEE ALSO

`all_active_scenarios(2)`
`all_scenarios(2)`
`create_scenario(2)`
`current_scenario(2)`
`remove_scenario(2)`
`set_active_scenarios(2)`
`set_mcmc_job_options(2)`
`set_host_options(2)`

get_drc_errors

Returns a collection of error objects that match the specified criteria.

SYNTAX

```
collection get_drc_errors
[-error_view mw_error_view]
[-error_id error_id]
[-type error_type]
[-bbox {llx lly urx ury}]
[-quiet]
[-regex [-nocase]]
[-exact]
[-filter expression]
```

Data Types

<i>mw_error_view</i>	list or collection of one item
<i>error_type</i>	list or collection
<i>llx</i>	float
<i>lly</i>	float
<i>urx</i>	float
<i>ury</i>	float
<i>expression</i>	string

ARGUMENTS

-error_view *mw_error_view*
Specifies the error view from which to query error objects.
If you do not specify this option, the top-level design is used. You can specify only one error view; specifying more than one error view causes an error.

-error_id *error_id*
Returns the error objects that match the specified error IDs.

-type *error_type*
Returns the error objects of the specified type.

-bbox {*llx lly urx ury*}
Returns the error objects in the specified area.

-quiet
Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed. By default, these messages are displayed.

-regex
Uses the *error_ids* argument as real regular expressions rather than as simple wildcard patterns.

-nocase
Makes the matches case-insensitive. By default, the matches are case-

sensitive. You can use the **-nocase** option only when you also use the **-regexp** option.

-exact

Disables simple pattern matching. Use this when searching for objects that contain the * (asterisk) wildcard character.

-filter *expression*

Filters the collection with the value of *expression*. For any error objects that match the specified criteria, the expression is evaluated based on the error object's attributes. To see the list of error object attributes that you can use in the expression, use the **list_attributes -application -class drc_error** command. If the expression evaluates to true, the error object is included in the result.

DESCRIPTION

This command returns a collection of error objects that match the specified criteria. If you do not specify any options, the command returns all error objects in the current top-level design.

EXAMPLES

The following example queries all errors in the short error type class from the current top-level design:

```
prompt> set errors [get_drc_errors -type "short"]
{1280 1281 1282 1283 1284 1285 1286}
```

SEE ALSO

```
create_drc_error(2)
create_drc_error_type(2)
create_open_drc_error(2)
create_open_locator_drc_error(2)
create_short_drc_error(2)
create_spacing_drc_error(2)
list_drc_error_types(2)
report_drc_error_type(2)
```

get_edit_groups

Returns a collection of edit groups.

SYNTAX

```
collection get_edit_groups
[-filter expression]
[-of_objects collection]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-object_id obj_id]
[-design_id cell_id]
[patterns]
```

ARGUMENTS

-filter *expression*
Filters the collection with the specified expression.
For any edit groups that match the specified patterns (or objects) the expression is evaluated based on the edit group's attributes.
If the expression evaluates to true, the edit group is included in the result.
This option works the same as the filter command.

-of_objects *collection*
Get edit groups of these objects.

-quiet
Suppresses warning and error messages if no objects match.
Syntax error messages are not suppressed.

-regexp
Uses the patterns argument as a real regular expression rather than simple wildcard patterns.

-nocase
Makes the matches case-insensitive.

-exact
Disables simple pattern matching.
For use when searching for objects that contain the * wildcard character.

-object_id *obj_id*
Specifies the object ID of one edit group.

-design_id *cell_id*
Specifies the design ID to look in instead of the current design.

patterns
Match edit group names against patterns (default = *).

RETURNS

groups
Collection of matching edit groups

DESCRIPTION

Creates a collection of edit groups from the current design

EXAMPLES

Get all edit groups in the design

```
> set gr [get_edit_groups]
```

Get edit groups matching regular expression

```
> set gr [get_edit_groups -regexp "[Pp].*"]
```

SEE ALSO

`create_edit_group(2)`

get_error_view_property

Returns the value of the specified error view property.

SYNTAX

```
string get_error_view_property  
-writer  
| -version  
| -ignore_type_name_property  
| -run_set  
| -areas  
| -excluded_areas  
| -command  
[-error_view mw_error_view]
```

Data Types

mw_error_view list or collection of one item

ARGUMENTS

-writer

Returns the value of the Writer property. This is the product name of the error view writer.

You can specify only one property to return. The following options are mutually exclusive: **-writer**, **-version**, **-ignore_type_name_property**, **-run_set**, **-areas**, **-excluded_areas**, and **-command**.

-version

Returns the value of the Version property. This is the product version of the error view writer.

You can specify only one property to return. The following options are mutually exclusive: **-writer**, **-version**, **-ignore_type_name_property**, **-run_set**, **-areas**, **-excluded_areas**, and **-command**.

-ignore_type_name_property

Returns the value of the IgnoreTypeNameProperty property. If this property is true, the TypeName property of the error type is ignored. This TypeName property is used to store type names that are longer than can normally be stored in the database. If value is false, the IgnoreTypeNameProperty property can be deleted.

You can specify only one property to return. The following options are mutually exclusive: **-writer**, **-version**, **-ignore_type_name_property**, **-run_set**, **-areas**, **-excluded_areas**, and **-command**.

-run_set

Returns the value of the Runset property. This is the name of the runset file input to the DRC check.

You can specify only one property to return. The following options are mutually exclusive: **-writer**, **-version**, **-ignore_type_name_property**, **-run_set**, **-areas**, **-excluded_areas**, and **-command**.

-areas

Returns the value of the Areas property. These are the areas of the design that were included in the DRC check.

You can specify only one property to return. The following options are mutually exclusive: **-writer**, **-version**, **-ignore_type_name_property**, **-run_set**, **-areas**, **-excluded_areas**, and **-command**.

-excluded_areas

Returns the value of the ExcludedAreas property. These are the areas of the design that were excluded from the DRC check.

You can specify only one property to return. The following options are mutually exclusive: **-writer**, **-version**, **-ignore_type_name_property**, **-run_set**, **-areas**, **-excluded_areas**, and **-command**.

-command

Returns the value of the Command property. This is the command string that was used to invoke the DRC or LVS check that produced the error view.

You can specify only one property to return. The following options are mutually exclusive: **-writer**, **-version**, **-ignore_type_name_property**, **-run_set**, **-areas**, **-excluded_areas**, and **-command**.

-error_view *mw_error_view*

Specifies the error view from which to query the property value.

If you do not specify this option, the properties are queried from the top-level design. You can specify only one error view; specifying more than one error view causes an error.

DESCRIPTION

This command returns the value of the specified error view property.

SEE ALSO

`set_error_view_property(2)`

get_flat_cells

Creates a collection of leaf cells that match certain criteria in the current design.

SYNTAX

```
collection get_flat_cells
[-quiet]
[-regex]
[-nocase]
[-exact]
[-filter expression]
patterns
  | -of_objects objects
  | -object_id object_id
[-all]
```

Data Types

<i>expression</i>	string
<i>patterns</i>	list
<i>objects</i>	list
<i>object_id</i>	integer

ARGUMENTS

-quiet
Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regex
Uses the *patterns* argument as real regular expressions rather than as simple wildcard patterns.

-nocase
Makes the matches case-insensitive.

-exact
Disables simple pattern matching. Use this when searching for objects that contain the * (asterisk) wildcard character.

-filter *expression*
Filters the collection with the value of *expression*. For any cells that match the specified criteria, the expression is evaluated based on the cell's attributes. If the expression evaluates to true, the cell is included in the result.

patterns
Matches cell full names against patterns. Patterns can include the wildcard characters * (asterisk) and ? (question mark). For more details about using and escaping wildcards, see the **wildcards** man page. The *patterns*, **-of_objects**, and **-object_id** options are mutually exclusive; you can choose

only one.

-of_objects *objects*

Creates a collection of cells connected to the specified objects. In this case, each object is a named cell, pin, net, bound, voltage area, plan group, power domain or a collection of these objects. The *patterns*, **-of_objects**, and **-object_id** options are mutually exclusive; you can choose only one.

-object_id *object_id*

Specifies the Milkyway object ID of the leaf cell to include in the collection. To determine the *id* value, use the **get_attribute** command to return the value of the **object_id** attribute. The *patterns*, **-of_objects**, and **-object_id** options are mutually exclusive; you can specify only one.

-all

Includes physical-only cells in the collection.

When a cell has one of the following types, it is physical-only:

- Standard Filler
- Pad Filler
- Corner
- Flip Chip Pad (Bump)
- Chip
- Cover
- Tap
- Cells containing only PG ports

DESCRIPTION

This command creates a collection of leaf cells that match certain criteria in the current design. Leaf cells are those processed during placement and route. They are shown and can be selected in GUI Layout window.

The command returns a collection of cells if any cell matches *patterns*, *objects*, or *object_id* and pass the filter (if specified). If no objects match the criteria, an empty string is returned.

When using *patterns*, the command searches all leaf cells to match *patterns* on full name regardless of their hierarchical level.

Regular expression matching is the same as in the Tcl **regexp** command. When using **-regexp**, take care in the way you quote the *patterns* and filter *expression*; use rigid quoting with curly braces around regular expressions. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding *".*"* to the beginning or end of the expressions as needed.

You can use the **get_flat_cells** command at the command prompt, or you can nest it as an argument to another command, such as **query_objects** and **change_selection**. In addition, you can assign the **get_flat_cells** result to a variable.

When issued from the command prompt, **get_flat_cells** behaves as though **query_objects** has been called to display the objects in the collection. By default, a maximum of 100 objects is displayed. You can change this maximum by using the

collection_result_display_limit variable.

The implicit query property of **get_flat_cells** provides a fast, simple way to display cells in a collection. However, if you want the flexibility provided by the **query_objects** options (for example, if you want to display the object class), use **get_flat_cells** as an argument to the **query_objects** command.

For information about collections and the querying of objects, see the **collections** man page.

get_flat_cells and **get_cells** returns the same type database objects. The collections returned by both commands are interchangeable and can be passed along to any command that accept cell objects.

get_flat_cells differs from **get_cells** in several ways. **get_cells** performs the pattern search based on the logical hierarchy of the design and are consistent with the commands used for synthesis and timing analysis that have been in Design Compiler, Primetime, and SDC. **get_flat_cells** searches all leaf cells to match *patterns* on full name relative to the current design, it does not depend on the current instance. It provides pattern search in the way that is consistent with the place-and-route view of the design, and thus is simpler for many tasks through ICC flow. The user is free to use whichever command is easier for them depending on the type of search they are doing. **get_flat_cells** is not part of the SDC format standard and should not be used in timing constraint scripts if the user wants to apply them across other tools, or using `read_sdc`.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example queries all soft macro cells in the current design. Although the output looks like a list, it is only a display.

```
prompt> get_flat_cells * -filter {is_soft_macro==true}
{I_ORCA_TOP/I_BLENDER_1 I_ORCA_TOP/I_BLENDER_2 I_ORCA_TOP/I_BLENDER_3 I_ORCA_TOP/
I_BLENDER_4 I_ORCA_TOP/I_BLENDER_5}
```

The following example queries the leaf cells whose full name begins with *BLOCK/*, equivalently all leaf cells any number of logical levels below the hierarchical cell *BLOCK*.

```
prompt> get_flat_cells BLOCK/*
{BLOCK/SB/U1 BLOCK/SB/U2 BLOCK/SB/U3 BLOCK/SB/U4 BLOCK/U5 BLOCK/U6 BLOCK/U7 BLOCK/
U8 BLOCK/U9}
```

As a comparison to the above example, the following example finds only cells that are direct children of the hierarchical cell *BLOCK*.

```
prompt> get_cells BLOCK/*  
{BLOCK/SB BLOCK/U9 BLOCK/U5 BLOCK/U8 BLOCK/U7 BLOCK/U6}
```

The following example shows that, given a collection of nets, you can select the leaf cells connected to those nets. You can see that these cells are selected in GUI Layout View if there is one opens for the current design.

```
prompt> set netsel [get_flat_nets r1_2_]  
{r1_2_}  
prompt> change_selection [get_flat_cells -of_object $netsel]
```

SEE ALSO

```
collections(2)  
filter_collection(2)  
get_cells(2)  
get_flat_pins(2)  
get_flat_nets(2)  
query_objects(2)  
change_selection(2)  
collection_result_display_limit(3)  
wildcards(3)
```

get_flat_nets

Creates a collection of top nets of hierarchical net groups and meet the specified criteria in the current design.

SYNTAX

```
collection get_flat_nets
[-filter expression]
[-quiet]
[-regexp]
[-nocase]
[-exact]
patterns
  | -of_objects objects
  | -object_id object_id
[-all]
```

Data Types

<i>patterns</i>	list
<i>objects</i>	list
<i>object_id</i>	integer

ARGUMENTS

-filter *expression*
Filters the collection with *expression*. For any nets that match the specified criteria, the expression is evaluated based on the net's attributes. If the expression evaluates to true, the net is included in the result.

-quiet
Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp
Uses the *patterns* argument as real regular expressions rather than simple wildcard patterns.

-nocase
Makes matches case-insensitive.

-exact
Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters.

patterns
Matches full names of top nets of hierarchical net groups against patterns. Patterns can include the wildcard characters * (asterisk) and ? (question mark). For more details about using and escaping wildcards refer to the **wildcards** man page. The *patterns*, **-of_objects**, and **-object_id** arguments are mutually exclusive; you can specify only one.

-of_objects *objects*

Creates a collection of top nets of hierarchical net groups that are connected to the specified objects. Each object is either a named pin, port, terminal, net, cell, shape, via, or a collection of these objects. The *patterns*, **-of_objects**, and **-object_id** arguments are mutually exclusive; you can specify only one.

-object_id *object_id*

Specifies the Milkyway object ID of the net to include in the collection. To determine the *id* value, use the **get_attribute** command to return the value of the **object_id** attribute. The *patterns*, **-of_objects**, and **-object_id** arguments are mutually exclusive; you can specify only one.

-all

Includes power and ground nets.

DESCRIPTION

The **get_flat_nets** command creates a collection of top nets of hierarchical net groups in the current design.

The command returns a collection if any nets match *patterns*, *objects*, or *object_id* and pass the filter (if specified). If no objects matched the criteria, the empty string is returned.

When using *patterns*, the command searches all top nets of hierarchical net groups to match *patterns* on full name.

Regular expression matching is the same as in the Tcl **regexp** command. When using **-regexp**, take care in the way you quote the *patterns* and filter *expression*. Use rigid quoting with curly braces around regular expressions. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding *.** to the beginning or end of the expressions as needed.

You can use the **get_flat_nets** command at the command prompt, or you can nest it as an argument to another command, such as **query_objects** and **change_selection**. In addition, you can assign the **get_flat_nets** result to a variable.

When issued from the command prompt, **get_flat_nets** behaves as though **query_objects** has been called to display the objects in the collection. By default, a maximum of 100 objects is displayed. You can change this maximum using the **collection_result_display_limit** variable.

The implicit query property of **get_flat_nets** provides a fast, simple way to display nets in a collection. However, if you want the flexibility provided by the **query_objects** options (for example, if you want to display the object class), use **get_flat_nets** as an argument to **query_objects**.

For information about collections and the querying of objects, see the **collections** man page.

get_flat_nets and **get_nets** returns the same type database objects. The collections returned by both commands are interchangeable and can be passed along to any command

that accept net objects.

get_flat_nets differs from **get_nets** in several ways. **get_nets** performs the pattern search based on the logical hierarchy of the design and are consistent with the commands used for synthesis and timing analysis that have been in Design Compiler, Primitime, and SDC. **get_flat_nets** searches top nets of hierarchical net groups to match *patterns* on full name relative to the current design, it does not depend on the current instance. It provides pattern search in the way that is consistent with the place-and-route view of the design, and thus is simpler for many tasks through ICC flow. The user is free to use whichever command is easier for them depending on the type of search they are doing. **get_flat_nets** is not part of the SDC format standard and should not be used in timing constraint scripts if the user wants to apply them across other tools, or using `read_sdc`.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example queries the top nets whose full name begins with *BLOCK/*. Although the output looks like a list, it is just a display.

```
prompt> get_flat_nets BLOCK/*  
{BLOCK/SB/n76}
```

The following example shows that there is no top net whose full name begins with *BLOCK/* and ends with *n1*.

```
prompt> get_flat_nets BLOCK/*n1  
Warning: No nets matched 'BLOCK/*n1' (SEL-004)
```

As a comparison to the above example, the following example shows that there is a net whose full name begins with *BLOCK/* and ends with *n1*.

```
prompt> get_nets BLOCK/*n1  
{BLOCK/n1}
```

The following selects top nets of hierarchy net groups that connect to cells, you can see that these nets are selected in GUI Layout View if there is one opens for the current design:

```
prompt> set cellsel [get_flat_cells I_ORCA_TOP/I_SDRAM_IF/U23276]  
{I_ORCA_TOP/I_SDRAM_IF/U23276}
```

```
prompt> change_selection [get_flat_nets -of_object $cellsel]
```

SEE ALSO

`collections(2)`
`filter_collection(2)`
`get_nets(2)`
`get_flat_cells(2)`
`get_flat_pins(2)`
`query_objects(2)`
`collection_result_display_limit(3)`
`wildcards(3)`

get_flat_pins

Creates a collection of pins of leaf cells that match the specified criteria in the current design.

SYNTAX

```
collection get_flat_pins
[-filter expression]
[-quiet]
[-regexp]
[-nocase]
[-exact]
patterns
  | -of_objects objects
  | -object_id object_id
[-all]
```

Data Types

<i>expression</i>	string
<i>patterns</i>	list
<i>objects</i>	list
<i>object_id</i>	integer

ARGUMENTS

-filter *expression*
Filters the collection with *expression*. For any pins that match the specified criteria, the expression is evaluated based on the pin's attributes. If the expression evaluates to true, the pin is included in the result.

-quiet
Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp
Uses the *patterns* argument as real regular expressions rather than simple wildcard patterns.

-nocase
Makes matches case-insensitive.

-exact
Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters.

patterns
Matches leaf cell pin full names against patterns. Patterns can include the wildcard characters * (asterisk) and ? (question mark). For more details about using and escaping wildcards, refer to the **wildcards** man page. The *patterns*, **-of_objects**, and **-object_id** arguments are mutually exclusive; you can specify only one.

-of_objects *objects*

Creates a collection of leaf cell pins connected to the specified objects. Each object is a named leaf cell or net or a collection of these objects. The *patterns*, **-of_objects**, and **-object_id** arguments are mutually exclusive; you can specify only one. In addition, you cannot use **-hierarchical** with **-of_objects**.

-object_id *object_id*

Specifies the Milkyway object ID of the leaf cell pin to include in the collection. To determine the *id* value, use the **get_attribute** command to return the value of the **object_id** attribute. The *patterns*, **-of_objects**, and **-object_id** arguments are mutually exclusive; you can specify only one.

-all

Includes power and ground pins.

DESCRIPTION

The **get_flat_pins** command creates a collection of pins of leaf cells that match certain criteria in the current design. Leaf cells are those processed during placement and route. They are shown and can be selected in GUI Layout window.

The command returns a collection if any pins match the *patterns*, *objects*, or *object_id* and pass the filter (if specified). If no objects match the criteria, the empty string is returned.

When using *patterns*, the command searches all pins of leaf cells to match *patterns* on full name regardless of their hierarchical level.

When used with **-of_objects**, **get_flat_pins** searches for pins connected to any leaf cells or nets specified in *objects*.

Regular expression matching is the same as in the Tcl **regexp** command. When using **-regexp**, take care in the way you quote the *patterns* and filter *expression*. Use rigid quoting with curly braces around regular expressions. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can expand the search by adding *.** to the beginning or end of the expressions as needed.

You can use the **get_flat_pins** command at the command prompt, or you can nest it as an argument to another command, such as **query_objects** and **change_selection**. In addition, you can assign the **get_flat_pins** result to a variable.

When issued from the command prompt, **get_flat_pins** behaves as though **query_objects** has been called to display the objects in the collection. By default, a maximum of 100 objects is displayed. You can change this maximum using the **collection_result_display_limit** variable.

The implicit query property of **get_flat_pins** provides a fast, simple way to display pins in a collection. However, if you want the flexibility provided by the **query_objects** options (for example, if you want to display the object class), use **get_flat_pins** as an argument to **query_objects**.

For information about collections and the querying of objects, see the **collections**

man page.

get_flat_pins and **get_pins** returns the same type database objects. The collections returned by both commands are interchangeable and can be passed along to any command that accept pin objects.

get_flat_pins differs from **get_pins** in several ways. **get_pins** performs the pattern search based on the logical hierarchy of the design and are consistent with the commands used for synthesis and timing analysis that have been in Design Compiler, Primetime, and SDC. **get_flat_pins** searches all pins of leaf cells to match *patterns* on full name relative to the current design, it does not depend on the current instance. It provides pattern search in the way that is consistent with the place-and-route view of the design, and thus is simpler for many tasks through ICC flow. The user is free to use whichever command is easier for them depending on the type of search they are doing. **get_flat_pins** is not part of the SDC format standard and should not be used in timing constraint scripts if the user wants to apply them across other tools, or using `read_sdc`.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example queries the *I* pins of leaf cells that begin with *I_CLK*. Although the output looks like a list, it is just a display.

```
prompt> get_flat_pins I_CLK*/I
{I_CLK_SOURCE_SDRAM_CLK/I I_CLK_SOURCE_SYS_2x_CLK/I I_CLK_SOURCE_SYS_CLK/
I I_CLK_SOURCE_PCLK/I}
```

The following example queries *ZN* pins of all leaf cells any number of logical levels below the hierarchical cell *BLOCK*.

```
prompt> get_flat_pins BLOCK/*/ZN
{BLOCK/SB/U1/ZN BLOCK/SB/U2/ZN BLOCK/SB/U3/ZN BLOCK/SB/U4/ZN BLOCK/U5/ZN BLOCK/U6/
ZN BLOCK/U7/ZN BLOCK/U8/ZN BLOCK/U9/ZN}
```

As a comparison to the above example, the following example finds *ZN* pins of cells that are direct children of the hierarchical cell *BLOCK*.

```
prompt> get_pins BLOCK/*/ZN
{BLOCK/U9/ZN BLOCK/U5/ZN BLOCK/U8/ZN BLOCK/U7/ZN BLOCK/U6/ZN}
```

The following example selects pins of leaf cells whose name begin with *O_1* you can see that these pins are selected in GUI Layout View if there is one opens for the current design:

```
prompt> set csel [get_flat_cells O_1*]
```

```
{0_10__reg 0_11__reg 0_12__reg 0_13__reg 0_1__reg}
```

```
prompt> change_selection [get_flat_pins -of_objects $csel]
```

SEE ALSO

```
collections(2)  
filter_collection(2)  
get_pins(2)  
get_flat_cells(2)  
get_flat_nets(2)  
query_objects(2)  
collection_result_display_limit(3)  
wildcards(3)
```

get_floorplan_data

Returns the value of the specified floorplan data attribute on the specified module.

SYNTAX

```
string get_floorplan_data  
-module module_object  
[-view one_level | logical | physical]  
[-quiet]  
attribute_name
```

Data Types

<i>module_object</i>	collection
<i>attribute_name</i>	string

ARGUMENTS

-module *module_object*
Specifies a single module object from which to get the floorplan attribute value. This is a required option.
The *module_object* must be a collection consisting of one object.
Valid objects are the current Milkyway design, hierarchical cells, soft macros, hard macros, and plan groups.

-view one_level | logical | physical
Specifies the view in which the attribute value is calculated.
When you specify **one_level** (the default), the attribute value is calculated by traversing only one level of hierarchy in the specified module.
When you specify **logical** or **physical**, the attribute value is calculated by traversing through all levels of hierarchy in the specified module. The difference between **logical** and **physical** is that soft macros are traversed in the physical view, but not in the logical view.
This option is ignored if you specify a plan group (plan groups have only a logical view).

-quiet
Suppresses all output messages from the execution of this command.

DESCRIPTION

This command returns the value of a floorplan data attribute on the specified module and view.

The return value is a string.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example returns the value of the **number_of_standard_cell** floorplan data attribute on all hierarchical cells in the current design, and combines it with the **full_name** application attribute to create a simple report.

```
prompt> foreach_in_collection sel \  
    [get_cells * -filter {is_hierarchical == true}] {echo -  
n "On '[get_attribute $sel full_name]', "  
echo "number_of_standard_cell = [get_floorplan_data -  
module $sel number_of_standard_cell -view one_level]"  
}  
On 'I_CLOCK_GEN', number_of_standard_cell = 10  
On 'I_RESET_BLOCK', number_of_standard_cell = 15  
On 'I_ORCA_TOP', number_of_standard_cell = 127
```

SEE ALSO

```
collections(2)  
foreach_in_collection(2)  
list_floorplan_data(2)  
floorplan_data_attributes(3)
```

get_fp_trace_mode

Determines whether a design is in trace mode.

SYNTAX

```
int get_fp_trace_mode
```

EXAMPLE

```
prompt> get_fp_trace_mode  
Design is not in trace mode  
0
```

```
prompt> set_fp_trace_mode
```

```
:  
:
```

```
prompt> get_fp_trace_mode  
Design is in trace mode  
1
```

```
prompt> set_fp_trace_mode -end
```

```
:  
:
```

```
prompt> get_fp_trace_mode  
Design is not in trace mode  
0
```

SEE ALSO

```
set_fp_trace_mode(2)  
get_fp_trace_mode(2)
```

get_fp_wirelength

Calculates the virtual route wirelength as a minimum length needed to connect all pins.

SYNTAX

```
status get_fp_wirelength
[-specified_nets object_list
 | -internal_nets
 | -interface_nets]
```

Data Types

object_list list

ARGUMENTS

-specified_nets *object_list*

Reports specified nets. Use this option if you want to estimate the wirelength on only the specified nets. The **-specified_nets**, **-internal_nets**, and **-interface_nets** are mutually exclusive; use only one.

-internal_nets

Reports only internal nets. This option shows the wirelength of internal nets for all plan groups. All of the pins of international nets are inside plan groups. If you use this option and the command does not find a plan group in the design, no processing is done. The **-specified_nets**, **-internal_nets**, and **-interface_nets** are mutually exclusive; use only one.

-interface_nets

Reports only interface nets. This option shows the wirelength of interface nets for all plan groups. Interface nets have at least 1 pin inside the plan group and 1 pin outside the plan group. If you use this option and the command does not find a plan group in the design, no processing is done. The **-specified_nets**, **-internal_nets**, and **-interface_nets** are mutually exclusive; use only one.

DESCRIPTION

This command calculates the virtual route wirelength as a minimum length needed to connect all pins. You can use it to evaluate overall placement quality. The command produces vertical, horizontal, and total wirelength statistics for top-level signal nets in the design.

It uses pin locations to calculate the minimum length of nets, with no layer-based information and no consideration of congestion. If a top-level port does not have a pin, the command issues a warning and skips the corresponding net. If a port inside the child cell does not have a pin, the command issues a warning and uses the coordinates of the center of the corresponding cell instance instead of the coordinates of the missing pin.

If you use the command without any options, it processes all nets in the design.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example processes all nets in a sample design.

```
prompt> get_fp_wirelength

INFO: Checking variables ...
INFO: Checking variables completed.
INFO: Initialization ...
INFO: Initialization completed.
INFO: Reading nets ...
INFO: Net VDD is a power/ground net. It is skipped.
INFO: Net VSS is a power/ground net. It is skipped.
INFO: Reading nets completed.
INFO: Started estimation of wire length ...
VR: Init Virtual Routing - Level 1
VR: Virtual Routing Completed
Number of nets successfully processed: 3746 Estimation
of wire length completed successfully.
Virtual Wire Length in X direction: 682600.8200
Virtual Wire Length in Y direction: 742716.0860
Total Virtual Wire Length: 1425316.9060

Elapsed =      0:00:01, CPU =      0:00:00
```

SEE ALSO

`place_fp_pins(2)`

get_generated_clocks

Creates a collection of generated clocks.

SYNTAX

```
collection get_generated_clocks
[-quiet]
[-regexp]
[-nocase]
[-filter expression]
[-exact]
patterns
```

Data Types

<i>expression</i>	string
<i>patterns</i>	list

ARGUMENTS

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed. This option is available only in XG mode.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also modifies the behavior of the `=~` and `!~` filter operators to compare with real regular expressions rather than simple wildcard patterns.

-nocase

Makes matches case-insensitive when combined with **-regexp**. Use **-nocase** only when you also use **-regexp**.

-filter *expression*

Filters the collection with *expression*. For any generated clocks that match *patterns* the expression is evaluated based on the generated clock's attributes. If the expression evaluates to true, the generated clock is included in the result.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the `*` (asterisk) and `?` (question mark) wildcard characters. The **-exact** and **-regexp** options are mutually exclusive.

patterns

Matches generated clock names against patterns. Patterns can include the `*` (asterisk) and `?` (question mark) wildcard characters.

DESCRIPTION

The **get_generated_clocks** command creates from the current design a collection of generated clocks that match certain criteria. The command returns a collection handle (identifier) if any generated clocks match the *patterns* and pass the filter (if specified). If no objects match the criteria, the empty string is returned.

To create a generated clock in the design, use the **create_generated_clock** command. To remove a generated clock from the design, use the **remove_generated_clock** command. To show information about clocks and generated clocks in the design, use the **report_clock** command.

Use the **get_generated_clocks** command at the command prompt or nest it as an argument to another command, such as **query_objects**. You can also assign the **get_generated_clocks** result to a variable.

When issued from the command prompt, **get_generated_clocks** behaves as though **query_objects** had been called to display the objects in the collection. By default a maximum of 100 objects is displayed. Change this maximum using the **collection_result_display_limit** variable.

The "implicit query" property of **get_generated_clocks** provides a fast, simple way to display cells in a collection. However, if you want the flexibility provided by the **query_objects** options, such as if you want to display the object class, use **get_generated_clocks** as an argument to **query_objects**.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following command applies the **set_clock_latency** command on all generated clocks in the design matching *GEN**:

```
prompt> set_clock_latency 2.0 [get_generated_clocks "GEN*"]
```

The following command removes all the generated clocks in the design matching *GEN1**:

```
prompt> remove_generated_clock [get_generated_clocks "GEN1*"]
```

SEE ALSO

```
collections(2)
create_generated_clock(2)
query_objects(2)
remove_generated_clock(2)
report_clock(2)
collection_result_display_limit(3)
```

get_ilm_objects

Returns a collection of nets, cells, or pins that are part of the interface logic models for the current design.

SYNTAX

```
collection get_ilm_objects  
[-type net | pin | cell]
```

ARGUMENTS

```
-type net | pin | cell
```

Specifies the type of object to be returned as a collection of objects. Valid values are **net**, **pin**, or **cell**. The command returns a collection of objects of the specified type that belong to the interface logic models on the current design.

DESCRIPTION

This command returns a collection of nets, cells, or pins that have been identified as belonging to interface logic models by using the **create_ilm** command. The **get_ilm_objects** command takes into consideration that the Interface Logic Model (ILM) preserves the hierarchy of the original design. Hence the collection contains nets, pins, and cells of all levels of hierarchy. You can use the command to review the objects that have been identified as belonging to the interface logic models on the current design.

For a discussion of ILM creation and the associated commands, see the **create_ilm** command man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example returns a collection of all interface logic cells.

```
prompt> get_ilm_objects -type cell
```

The following example returns a collection of all interface logic nets.

```
prompt> get_ilm_objects -type net
```

The following example returns a collection of all interface logic pins.

```
prompt> get_ilm_objects -type pin
```

SEE ALSO

`create_ilm(2)`

get_ilms

Creates a collection of interface logic models (ILMs) defined in the current design.

SYNTAX

```
collection get_ilms  
[-quiet]  
[-reference]  
[-filter expression]  
patterns
```

Data Types

<i>expression</i>	string
<i>patterns</i>	list

ARGUMENTS

-quiet

Suppresses warning and error messages if no objects match. It does not suppress syntax error messages. The default is to show warning and error messages.

-reference

Returns reference cells for the ilm blocks. The default is to return the instances.

-filter *expression*

Filters the collection with the *expression* value. For any ILMs that match the *patterns* value, the expression is evaluated based on the attributes of the ILM. If the expression evaluates to **true**, the ILM is included in the result. By default, the collection is not filtered.

patterns

Matches the placement keepout names against the *patterns* list. The *patterns* list can include the * (asterisk) wildcard character.

DESCRIPTION

This command creates a collection of ilm cells/blocks from the current design that match certain criteria. It returns a collection handle (identifier) if any ilms match a *patterns* list value and pass the filter (if specified). If no objects match the criteria, the command returns an empty string.

You can use this command at the command prompt, or you can nest it as an argument to another command. You can also assign the **get_ilms** result to a variable.

When you issue the **get_ilms** command from the command prompt, by default, the display shows a maximum number of **100** objects. You can use the **collection_result_display_limit** variable to change the maximum number.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following examples get all ILMs that have a name starting with my_ilm.

```
prompt> get_ilms my_ilm*
```

The following examples get all reference of ILM blocks that have a name starting with JE.

```
prompt> get_ilms -reference JE*  
{JE1 JE2}
```

The following example queries the ilms that begin with je and refer- ences a JE1 library cell.

```
prompt> get_ilms "je*" -filter "@ref_name == JE1"  
{je1_sm}
```

The following example shows that given a collection of ILM cell instance, you can query the pins connected to those cells:

```
prompt> get_pins -of_objects [get_ilms je1_sm]  
{je1_sm/je1_mod_in2ff1_in je1_sm/clock je1_sm/je1_ff4_out2mod_out}
```

SEE ALSO

```
collections(2)  
foreach_in_collection(2)  
query_objects(2)  
report_design(2)  
collection_result_display_limit(3)
```

get_layer_attribute

Queries layer attribute.

SYNTAX

```
string get_layer_attribute  
[-quiet]  
[-layer layer]  
[attribute]
```

ARGUMENTS

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-layer *layer*

Specify the layer to query attributes. The layer could be specified by a collection or by layer name.

attribute

Specify which attribute to query. The name of the attributes can be queried by: `prompt>list_attributes -application -class layer`

DESCRIPTION

This command retrieves the attribute value of certain layer. The return value is a Tcl string.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example queries the layer number.

```
> get_layer_attribute -layer M1 layer_number  
3
```

SEE ALSO

`get_layers(2)`
`get_attribute(2)`

get_layers

Creates a collection of one or more layers.

SYNTAX

```
collection get_layers
[-filter expression]
[-quiet]
[-regexp [-nocase]] | [-exact]
[-include_system]
[patterns]
[-exact]
```

Data Types

patterns string

ARGUMENTS

-filter *expression*

Filters the collection with the value of *expression*. For any layers that match, the expression is evaluated based on the layer's attributes. If the expression evaluates to true, the layer is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* option as real regular expressions rather than simple wildcard patterns. This option modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than with simple wildcard patterns. The **-regexp** and **-exact** options are mutually exclusive; you can use only one.

-nocase

Makes matches case-insensitive. You can use this option only when you also use the **-regexp** option.

-include_system

Includes the system layers in the collection. Without this option, only user layers that are defined in the technology file can be included in the collection. Layer numbers from 1 to 187 are for user layers, and the layer numbers great than 187 are for system layers.

patterns

Matches layer names against patterns. Patterns can include the wildcard characters * (asterisk) and ? (question mark) or regular expressions, depending on whether or not you also use the **-regexp** option.

-exact

Disables simple pattern matching. You can use this option when searching for

objects that contain the * (asterisk) and ? (question mark) wildcard characters. The **-regex** and **-exact** options are mutually exclusive; you can use only one.

DESCRIPTION

This command creates a collection of layers defined in the technology file for the current library, or a collection that also contains system layers if you use the **-include_system** option.

You can use the **get_layers** command at the command prompt or as an argument nested in another command (for example, in the **query_objects** command). You can also assign its result to a variable.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example returns a layer for the name M1R.

```
prompt> get_layers M1R  
{M1R}
```

The following example returns layers for routing.

```
prompt> get_layers -filter is_routing_layer==TRUE  
{METAL5 METAL METAL2 METAL3 METAL4 M6 M7 M8 PA}
```

SEE ALSO

collections(2)
filter_collection(2)
get_layer_attribute(2)
query_objects(2)

get_lib_attribute

Returns the value of an attribute on a list of library objects.

SYNTAX

```
list get_lib_attribute  
object_list  
attribute_name
```

Data Types

<i>object_list</i>	list
<i>attribute_name</i>	string

ARGUMENTS

object_list
Lists the library objects for which the attribute value is to be returned.

attribute_name
Specifies the name of the attribute whose value is to be returned.

DESCRIPTION

This command searches *object_list* for the specified attribute and returns a list of attribute values. If the attribute is not found on any of the specified objects, the command returns an empty list.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows how to find the cell named INV and obtain its area value.

```
prompt> get_lib_attribute sample/INV area  
Performing get_lib_attribute on library object  
'sample/INV'.  
{1.0}
```

The following example shows an error issued because the the command does not find a cell named INV1.

```
prompt> get_attribute sample/INV1 area  
Error: The 'INV1' object does not exist in the 'sample' technology library. (UIL-54)
```

```
{}
```

The following example shows the `lib_udg_attr` value returned for the user-defined group named `lib_udg1` in the library named `sample`.

```
prompt> get_lib_attribute\  
sample/lib_udg1 lib_udg_attr  
Performing get_lib_attribute on library object 'sample/lib_udg1'.  
{Test}
```

The following example shows the area values returned for gates named `G1` and `G2` in the library named `tech_lib`.

```
prompt> get_lib_attribute  
{sample/G1, sample/G2} area  
Performing get_lib_attribute on library object 'sample/G1'.  
Performing get_lib_attribute on library object 'sample/G2'.  
{1.0, 2.0}
```

SEE ALSO

`set_lib_attribute(2)`

get_lib_cells

Creates a collection of library cells from the libraries loaded into memory.

SYNTAX

```
collection get_lib_cells
[-filter expression]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-scenario scenario_name]
[patterns]
[-of_objects objects]
```

Data Types

<i>expression</i>	string
<i>scenario_name</i>	string
<i>patterns</i>	list
<i>objects</i>	list

ARGUMENTS

-filter *expression*
Filters the collection with *expression*. For any library cells that match *patterns*, the expression is evaluated based on the library cell's attributes. If the expression evaluates to true, the library cell is included in the result.

-quiet
Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp
Uses the *patterns* argument as real regular expressions rather than simple wildcard patterns.

-nocase
Makes matches case-insensitive.

-exact
Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters.

-scenario *scenario_name*
Creates a collection of library cells that belong to the libraries which are used in the specified scenario. The result will be based on the following options: pattern, -exact, -nocase, and -regexp.

patterns
Matches library cell names against patterns. Patterns can include the

wildcard characters * (asterisk) and ? (question mark). For more details about using and escaping wildcards refer to the **wildcards** man page.

-of_objects *objects*

Creates a collection of library cells that are referenced by the specified cells or own the specified library pins. In this case, each object is either a named library pin, netlist cell, library pin collection, or a netlist cell collection. The **-of_objects** and *patterns* arguments are mutually exclusive. You must specify one, but not both.

DESCRIPTION

The **get_lib_cells** command creates a collection of library cells from libraries currently loaded into memory that match certain criteria. If no objects match the criteria, the empty string is returned.

Regular expression matching is the same as in the Tcl **regexp** command. When using **-regexp**, take care in the way you quote the *patterns* and filter *expression*. Using rigid quoting with curly braces around regular expressions is recommended. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search simply by adding *.** to the beginning or end of the expressions as needed.

You can use the **get_lib_cells** command at the command prompt, or you can nest it as an argument to another command, such as **query_objects**. In addition, you can assign the **get_lib_cells** result to a variable.

When issued from the command prompt, **get_lib_cells** behaves as though **query_objects** has been called to display the objects in the collection. By default, a maximum of 100 objects is displayed. You can change this maximum using the **collection_result_display_limit** variable.

The implicit query property of **get_lib_cells** provides a fast, simple way to display library cells in a collection. However, if you want the flexibility provided by the **query_objects** options (for example, if you want to display the object class), use **get_lib_cells** as an argument to **query_objects**.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

By default, this command uses information from all active scenarios. You can select different scenarios by using the **-scenario** option.

EXAMPLES

The following example queries all library cells that are in the *misc_cmos* library and begin with "AN2". Although the output looks like a list, it is just a display.

```
prompt> get_lib_cells misc_cmos/AN2*  
{misc_cmos/AN2 misc_cmos/AN2P}
```

The following example shows one way to determine the library cell used by a particular cell instance:

```
prompt> get_lib_cells -of_objects [get_cells o_reg1]  
{misc_cmos/FD2}
```

SEE ALSO

```
collections(2)  
filter_collection(2)  
get_cells(2)  
query_objects(2)  
collection_result_display_limit(3)  
wildcards(3)
```

get_lib_pins

Creates a collection of library cell pins from libraries loaded into memory.

SYNTAX

```
collection get_lib_pins
[-filter expression]
[-quiet]
[-regexp]
[-nocase]
[-exact]
patterns | -of_objects objects
```

Data Types

<i>expression</i>	string
<i>patterns</i>	list
<i>objects</i>	list

ARGUMENTS

- filter** *expression*
Filters the collection with *expression*. For any library cell pins that match *patterns*, the expression is evaluated based on the library cell pin's attributes. If the expression evaluates to true, the lib_pin is included in the result.
- quiet**
Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.
- regexp**
Uses the *patterns* argument as real regular expressions rather than simple wildcard patterns.
- nocase**
Makes matches case-insensitive.
- exact**
Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters.
- patterns**
Matches library cell pin names against patterns. Patterns can include the wildcard characters * (asterisk) and ? (question mark). For more details about using and escaping wildcards refer to the **wildcards** man page.
- of_objects** *objects*
Creates a collection of library cell pins referenced by the specified netlist pins or owned by the specified library cells. In this case, each object is either a named library cell, netlist pin, library cell collection, or a netlist pin collection. The **-of_objects** and *patterns* arguments are mutually

exclusive; you must specify one, but not both.

DESCRIPTION

The **get_lib_pins** command creates a collection of library cell pins from the libraries currently loaded into memory that match certain criteria. If no objects matched the criteria, the empty string is returned.

Regular expression matching is the same as in the Tcl **regexp** command. When using **-regexp**, take care in the way you quote the *patterns* and *filter expression*. Using rigid quoting with curly braces around regular expressions is recommended. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search simply by adding *.** to the beginning or end of the expressions as needed.

You can use the **get_lib_pins** command at the command prompt, or you can nest it as an argument to another command, such as **query_objects**. In addition, you can assign the **get_lib_pins** result to a variable.

When issued from the command prompt, **get_lib_pins** behaves as though **query_objects** has been called to display the objects in the collection. By default, a maximum of 100 objects is displayed. You can change this maximum using the **collection_result_display_limit** variable.

The implicit query property of **get_lib_pins** provides a fast, simple way to display library cell pins in a collection. However, if you want the flexibility provided by the **query_objects** options (for example, if you want to display the object class), use **get_lib_pins** as an argument to **query_objects**.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command uses information from all active scenarios.

EXAMPLES

The following example queries all pins of the AN2 library cell in the misc_cmos library. Although the output looks like a list, it is just a display.

```
prompt> get_lib_pins misc_cmos/AN2/*
{misc_cmos/AN2/A misc_cmos/AN2/B misc_cmos/AN2/Z}
```

The following example shows one way to find out how the library pin is used by a particular pin in the netlist:

```
prompt> get_lib_pins -of_objects o_reg1/Q
{misc_cmos/FD2/Q}
```

SEE ALSO

`collections(2)`
`filter_collection(2)`
`get_libs(2)`
`get_lib_cells(2)`
`query_objects(2)`
`collection_result_display_limit(3)`
`wildcards(3)`

get_libs

Creates a collection of libraries loaded into memory.

SYNTAX

```
collection get_libs
[-filter expression]
[-quiet]
[-regex]
[-nocase]
[-exact]
[-scenario scenario_name]
[-of_objects objects]
[patterns]
```

Data Types

<i>expression</i>	string
<i>objects</i>	list
<i>patterns</i>	list

ARGUMENTS

- filter** *expression*
Filters the collection with *expression*. For any libraries that match *patterns*, or *objects*, the expression is evaluated based on the library's attributes. If the expression evaluates to true, the library is included in the result.
- quiet**
Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.
- regex**
Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, this option modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns.
The **-regex** and **-exact** options are mutually exclusive.
- nocase**
Makes matches case-insensitive when used with **-regex**. You must use this option with the **-regex** option.
- exact**
Disables simple pattern matching. This option is used when searching for objects that contain the * (asterisk) and ? (question mark) wildcard characters.
The **-exact** and **-regex** options are mutually exclusive.
- scenario** *scenario_name*
Creates a collection of libraries that are used in the specified scenario.

The result is based on the following options: *pattern*]fP, **-exact**, **-nocase**, and **-regex**.

-of_objects *objects*

Creates a collection of libraries that contain the specified objects. Each object is either a named library cell or a library cell collection.

The **-of_objects** and *patterns* arguments are mutually exclusive; you must specify one, but not both.

patterns

Matches library names against patterns. Patterns can include the ***** (asterisk) and **?** (question mark) wildcard characters, based on the **-regex** option. For more details about using and escaping wildcard characters refer to the **wildcards** man page.

The *patterns* and **-of_objects** arguments are mutually exclusive; you must specify one, but not both.

DESCRIPTION

The **get_libs** command creates a collection of libraries from those currently loaded into memory that match certain criteria. The command returns a collection handle or identifier if any libraries match the *patterns* and pass the filter, if specified. If no objects match the criteria, the empty string is returned.

When using the **-regex** option, use care in the way you quote the *patterns* and filter *expression*. It is best to use rigid quoting with curly braces around regular expressions.

Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can expand the search by adding **.*** (dot asterisk) to the beginning or end of the expressions.

You can use the **get_libs** command at the command prompt, or you can nest it as an argument to another command, such as **query_objects**. You can also assign the **get_libs** result to a variable.

When issued from the command prompt, **get_libs** behaves as though **query_objects** has been called to display the objects in the collection. By default, a maximum of 100 objects is displayed. You can change this maximum using the **collection_result_display_limit** variable.

The "implicit query" property of **get_libs** provides a fast, simple way to display cells in a collection. However, if you want the flexibility provided by the **query_objects** options (for example, if you want to display the object class), use **get_libs** as an argument to **query_objects**.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

By default, this command uses information from all active scenarios. You can select

different scenarios by using the **-scenario** option.

EXAMPLES

The following example queries all loaded libraries. Use the **list_libraries** command to get a complete listing of the libraries.

```
prompt> get_libs *  
{"misc_cmos", "misc_cmos_io"}
```

The following example removes a library collection. You cannot remove libraries if they are referenced by a design.

```
prompt> remove_lib [get_libs misc*]  
Removing library misc_cmos...  
Removing library misc_cmos_io...
```

SEE ALSO

```
collections(2)  
filter_collection(2)  
query_objects(2)  
collection_result_display_limit(3)  
wildcards(3)
```

get_license

Obtains a license for a feature.

SYNTAX

```
status get_license  
feature_list
```

Data Types

```
feature_list      list
```

ARGUMENTS

```
feature_list  
    Specifies a list of features to be obtained. If more than one feature is  
    specified, they must be enclosed in braces ({}).  
    By looking at your key file, you can determine all of the features licensed  
    at your site.
```

DESCRIPTION

Obtains a license for the named features. These features are checked out by the current user until the **remove_license** command is used or until the program is exited.

The **list_licenses** command provides a list of the features that you are currently using.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

This example obtains the multivoltage feature:

```
prompt> get_license {Galaxy-MV}
```

SEE ALSO

```
check_license(2)  
license_users(2)  
list_licenses(2)  
remove_license(2)
```

get_location

Gets the location of the specified objects.

SYNTAX

```
list get_location  
object_list  
[-rp_group rp_group]
```

Data Types

<i>object_list</i>	list
<i>rp_group</i>	string

ARGUMENTS

object_list
Lists objects whose location is to be returned.

-rp_group *rp_group*
Specifies the relative placement group within which to get the lattice position location.

DESCRIPTION

This command returns a list containing the x and y location of the objects specified. The input objects can be cells, ports, pins, or, when the **-rp_group** option is used, RP groups. The x and y values are absolute values. It means they are specified with respect to (0,0).

When getting the location of an RP group within another RP group the command can obtain multiple locations when the RP group is hierarchically instantiated multiple times. In such a case, the returned list of positions contains a sublist giving each of the locations for that RP group.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows how **get_location** can be used to get the location of an object

```
prompt> set cell_location\  
[get_location [get_cell U11]]  
prompt> set x_loc [lindex $cell_location 0]  
prompt> set y_loc [lindex $cell_location 1]
```

```
prompt> set cell_rp_location\  
[get_location\  
[get_cell U11]\  
-rp_group my_group]  
prompt> set x_rp_loc [lindex $cell_rp_location 0]  
prompt> set y_rp_loc [lindex $cell_rp_location 1]
```

SEE ALSO

```
all_rp_groups(2)  
get_cells(2)  
get_ports(2)
```

get_magnet_cells

Returns a collection of cells that can be pulled closer to magnet cells. Magnet cells can be specified as an argument to this command.

SYNTAX

```
collection get_magnet_cells
[-stop_by_sequential_cells]
[-exclude_buffers]
[-logical_level level]
[-stop_points object_list]
magnet_objects
```

Data Types

<i>level</i>	integer
<i>object_list</i>	collection
<i>magnet_objects</i>	collection

ARGUMENTS

-stop_by_sequential_cells

Specifies for the magnetizing of cells to stop when a sequential cell is encountered during the traversal of levels. By default, all the cells in the logical levels specified by the *magnet_objects* argument are magnetized by this command.

-exclude_buffers

Prevents consideration of buffers and inverters when calculating the level. They will be pulled like normal cells but will be skipped over for level determination. By default, buffers and inverters are considered to be one level of logic.

-logical_level *level*

Specifies the number of logical levels to magnetize. By default, the *level* value is 1, which means that only the first level cells are magnetized. The **-logical_level** and **-stop_points** options are mutually exclusive; you can use only one.

-stop_points *object_list*

Identifies the cells that lie on a timing path between the objects specified in *magnet_objects* and those that are specified in the *object_list*. These objects can be named in a list of any pin, port, or cell objects. The **-logical_level** and **-stop_points** options are mutually exclusive; you can use only one.

magnet_objects

Specifies the objects to become magnets. Any objects can be a fixed macro cell, a pin of a fixed macro cell, or an I/O pin. Vias, blockages, and IO pads are not allowed to be magnets.

DESCRIPTION

This command allows you obtain a collection of cells that are affected by magnet placement. You can use the returned collection to create bounds before running the **create_placement** command. You can also use it to highlight cells affected by magnet placement in the GUI.

If there are a large number of high fanout nets, magnet placement might pull a large number of cells resulting in long run-time. In order to avoid such cases, you can set the **magnet_placement_fanout_limit** environment variable to some reasonable value. Its default value is 1000.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example gets a collection of cells that are affected by magnet placement on a cell named INST_1.

```
prompt> get_magnet_cells [get_cells "INST_1"]
```

The following example obtains a collection of cells that would get pulled when performing magnet placement on a cell named INST_2. Two levels of cells are pulled and buffers, if any exist, are skipped over.

```
prompt> get_magnet_cells\  
-exclude_buffers [get_cells "INST_2"]
```

SEE ALSO

```
create_placement(2)  
legalize_placement(2)  
magnet_placement(2)  
set_cell_location(2)  
set_dont_touch_placement(2)
```


get_message_info

Returns information about diagnostic messages.

SYNTAX

```
Integer get_message_info
[-error_count | -warning_count | -info_count |
-limit l_id
  | -occurrences o_id
  | -suppressed s_id]
```

Data Types

<i>l_id</i>	string
<i>o_id</i>	string
<i>s_id</i>	string

ARGUMENTS

-error_count
Information returned is the number of error messages issued so far.

-warning_count
Information returned is the number of warning messages issued so far.

-info_count
Information returned is the number of informational messages issued so far.

-limit *l_id*
Information returned is the current user-specified limit for a given message id. The limit was set with **set_message_info**.

-occurrences *o_id*
Information returned is the number of occurrences of a given message id.

-suppressed *s_id*
Information returned is the number of times a message was suppressed by usage of **suppress_message** or due to exceeding a user-specified limit.

DESCRIPTION

The **get_message_info** command retrieves information about error, warning, and informational messages. For example, if the following message is generated, information about it is recorded.

Error: unknown command 'wrong_command' (CMD-005)

It is useful to be able to retrieve recorded information about generated diagnostic messages. For example, you can stop a script after a certain number of errors have occurred, or monitor the number of messages issued by a single command. For an example that shows using **get_message_info** in a procedure, see the EXAMPLES section. You can also find out how many times a specific message has occurred, or how many times it has been suppressed. Also, you can find out if a limit has been set for a

particular message id.

EXAMPLES

The following example uses **get_message_info** to count the number of errors that occurred during execution of a specific command, and to return from the procedure if the error count exceeds a given amount.

```
prompt> proc
do_command {limit} {      set current_errors [get_message_info -error_count]
    command
    set new_errors [get_message_info -error_count]
    if {[expr $new_errors - $current_errors] > $limit} {      return -
code error "Too many errors"
}
    ...
}
```

SEE ALSO

```
set_message_info(2)
print_message_info(2)
suppress_message(2)
```

get_mw_cels

Creates a collection of one or more Milkyway designs.

SYNTAX

```
collection get_mw_cels
[-hierarchical]
[-quiet]
[-regexp [-nocase]] | [-exact]
[-filter expression]
patterns
[-exact]
```

Data Types

<i>expression</i>	string
<i>patterns</i>	list

ARGUMENTS

-hierarchical

Searches for Milkyway designs inferred by the design hierarchy relative to the current instance. The full name of the object at a particular level must match the patterns. This option does not force an auto link.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. This option modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. The **-regexp** and **-exact** options are mutually exclusive; you can use only one.

-nocase

Makes matches case-insensitive. You can use this option only when you also use the **-regexp** option.

-filter *expression*

Filters the collection with the value of *expression*. For any Milkyway designs that match, the expression is evaluated based on the Milkyway design's attributes. If the expression evaluates to **true**, the Milkyway design is included in the result.

patterns

Matches Milkyway design names against patterns. Patterns can include the wildcard characters * (asterisk) and ? (question mark) or regular expressions, based on the **-regexp** option. Patterns can also include collections of Milkyway celtypes.

-exact

Disables simple pattern matching. You can use this when searching for objects that contain the * (asterisk) and ? (question mark) wildcard characters. The **-regexp** and **-exact** options are mutually exclusive; you can use only one.

DESCRIPTION

This command creates a collection of Milkyway designs from those currently loaded into the tool that match certain criteria. It returns a collection if any Milkyway designs match the *patterns* value and pass the filter (if specified). If no objects match the criteria, the command returns an empty string.

Regular expression matching is the same as for the Tcl **regexp** command. When using the **-regexp** option, take care in the way you quote the *patterns* and filter *expression* options. Using rigid quoting with curly braces around regular expressions is recommended. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search simply by adding **.*** to the beginning or end of the expressions as needed.

You can use the **get_mw_cels** command at the command prompt, or you can use it as an argument nested in another command (for example, in the **query_objects** command), or you can assign its result to a variable.

When issued from the command prompt, the **get_mw_cels** command behaves as though the **query_objects** command has been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum by using the **collection_result_display_limit** variable.

The "implicit query" property of the **get_mw_cels** command provides a fast, simple way to display Milkyway designs in a collection. However, if you want the flexibility provided by the **query_objects** options (for example, if you want to display the object class), use the **get_mw_cels** command as an argument to the **query_objects** command.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example queries mw_cels that begin with *mpu*. Although the output looks like a list, it is just a display. You can use the **list_mw_cels** command to see a complete list of Milkyway designs.

```
prompt> get_mw_cels mpu*
{"mpu_0_0", "mpu_0_1", "mpu_1_0", "mpu_1_1"}
```

The following example shows that, given a collection of Milkyway designs, you can remove those Milkyway designs.

```
prompt> remove_mw_cel [get_mw_cels mpu*]  
Removing mw_cel mpu_0_0...  
Removing mw_cel mpu_0_1...  
Removing mw_cel mpu_1_0...  
Removing mw_cel mpu_1_1...
```

SEE ALSO

```
collections(2)  
filter_collection(2)  
list_mw_cels(2)  
query_objects(2)  
remove_mw_cel(2)  
collection_result_display_limit(3)
```

get_net_shapes

Creates a collection by selecting net shapes from the current design.

SYNTAX

```
collection get_net_shapes
[-within region
  | -intersect region
  | -touching region
  | -at point]
[-filter expression]
[-quiet]
[-type {vw | hw | path}]
[patterns | -of_objects net_list]
```

Data Types

<i>region</i>	list
<i>point</i>	string
<i>expression</i>	string
<i>patterns</i>	list
<i>net_list</i>	list

ARGUMENTS

-within *region*

Creates a collection containing all net shapes within the specified region. The region boundary can be rectangle or polygon. The valid format of a rectangle is `{{llx lly} {urx ury}}` or `{llx lly urx ury}`, which specifies the lower-left and upper-right corners of the rectangle. The valid format of a polygon is: `{{x1 y1} {x2 y2} ... {xN yN} {x1 y1}}`, and each `{x y}` pair specifies one point of the input polygon. Besides, a list of one polygon is also supported as input for this option, with the format: `{{{x1 y1} {x2 y2} ... {xN yN} {x1 y1}}}`. Pay attention that the valid polygon is rectilinear, so the adjacent points have one same coordinate. The coordinate unit is specified in technology file (usually it is micron). The **-within**, **-intersect**, **-touching**, and **-at** options are mutually exclusive; you can specify only one.

-intersect *region*

Creates a collection containing all net shapes intersect the specified region. The region boundary can be rectangle or polygon. The valid format of a rectangle is `{{llx lly} {urx ury}}` or `{llx lly urx ury}`, which specifies the lower-left and upper-right corners of the rectangle. The valid format of a polygon is: `{{x1 y1} {x2 y2} ... {xN yN} {x1 y1}}`, and each `{x y}` pair specifies one point of the input polygon. Besides, a list of one polygon is also supported as input for this option, with the format: `{{{x1 y1} {x2 y2} ... {xN yN} {x1 y1}}}`. Pay attention that the valid polygon is rectilinear, so the adjacent points have one same coordinate. The coordinate unit is specified in technology file (usually it is micron). The **-within**, **-intersect**, **-touching**, and **-at** options are mutually exclusive; you can specify only one.

-touching region

Creates a collection containing all net shapes touching the specified region. The region boundary can be rectangle or polygon. The valid format of a rectangle is `{{llx lly} {urx ury}}` or `{llx lly urx ury}`, which specifies the lower-left and upper-right corners of the rectangle. The valid format of a polygon is: `{{x1 y1} {x2 y2} ... {xN yN} {x1 y1}}`, and each `{x y}` pair specifies one point of the input polygon. Besides, a list of one polygon is also supported as input for this option, with the format: `{{{x1 y1} {x2 y2} ... {xN yN} {x1 y1}}}`. Pay attention that the valid polygon is rectilinear, so the adjacent points have one same coordinate. The coordinate unit is specified in technology file (usually it is micron). The **-within**, **-intersect**, **-touching**, and **-at** options are mutually exclusive; you can specify only one.

-at point

Creates a collection containing all net shapes that overlap the point specified by the coordinates `{x y}`. The coordinate unit is specified in technology file (usually it is micron). The **-within**, **-intersect**, **-touching**, and **-at** options are mutually exclusive; you can specify only one.

-filter expression

Filters the collection with the value of *expression*, which must be a Boolean expression based on bound attributes. You can determine the net shape attributes by using the **list_attributes** command. If the expression evaluates to **true**, the net shape is included in the collection. By default, this option is off.

-quiet

Suppresses the reporting of non-terminal (warning and informational) messages.

-type {vw | hw | path}

Specifies the type of net shape, where **vw** is for vertical wire, **hw** is for horizontal wire, and **path** is the actual path. By default, this option is off.

patterns

Matches the net shape names against the patterns in the current design. You can specify the patterns by using the following formats:

- `*` (asterisk) allows all net shapes.
 - `HWIRE#*` allows all horizontal wire net shapes.
 - `HWIRE# object_id` allows all horizontal wires with a specified object ID.
 - `VWIRE#*` allows all vertical wire net shapes.
 - `VWIRE# object_id` allows all vertical wires with a specified object ID.
- The *patterns* and **-of_objects** options are mutually exclusive. If you do not specify either the *patterns* and **-of_objects** options, the command defaults to using `*` as the pattern. By default, this option is off.

-of_objects net_list

Creates a collection containing the net shapes connected to the specified nets. The *patterns* and **-of_objects** options are mutually exclusive. If you do

not specify either the *patterns* and *-of_objects* options, the command defaults to using *** as the pattern. By default, this option is off.

DESCRIPTION

This command creates a collection of net shapes by selecting net shapes from the current design that meet the selection criteria. It returns a collection handle if one or more net shapes meet the selection criteria. If no net shapes match the selection criteria, it returns an empty string.

When using **-within**, **-touching**, or **-intersect**, note that the behavior will be different for the different format of the specified region. A shape object has its boundary and bbox, sometimes they are different. When the specified region is in rectangle format, we will search objects by their bbox; when the specified region is in polygon format, we will search objects by their boundary. And searching by bbox is faster than searching by boundary.

You can use the **get_net_shapes** command at the command prompt, use it as an argument nested in another command, or assign its result to a variable.

See the **collections** command man page for information about working with collections.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example get all net shapes within specified rectangle.

```
prompt> get_net_shapes * -within {{4 20} {15 30}}
{"HWIRE#7264", "VWIRE#6766"}
```

The following example gets net shapes intersect specified polygon:

```
prompt>get_net_shapes -intersect \
{{30 20} {50 20} {50 30} {40 30} {40 40} {30 40} {30 20}}
{HWIRE#39936}
```

The following example creates a net shape collection.

```
prompt> get_net_shapes -of_objects n300 \
-intersect {{4 20} {15 30}}
{"VWIRE#6766"}
```

The following example creates a net shape collection.

```
prompt> get_net_shapes -filter {route_type=="P/G Std. Cell Pin Conn"}
```



```
{PATH#1134336 PATH#1134337 PATH#1134338 PATH#1134339 PATH#1134340}
```

SEE ALSO

```
collections(2)  
create_net_shape(2)  
remove_net_shape(2)  
get_user_shapes(2)  
filter_collection(2)  
query_objects(2)  
collection_result_display_limit(3)
```

get_nets

Creates a collection of nets that meet the specified criteria.

SYNTAX

```
collection get_nets
[-hierarchical]
[-filter expression]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-top_net_of_hierarchical_group]
[-segments]
[-boundary_type boundary_type]
patterns
  | -of_objects objects
  | -object_id object_id
[-all]
[-design_id design_id]
```

Data Types

<i>expression</i>	string
<i>boundary_type</i>	string
<i>patterns</i>	list
<i>objects</i>	list
<i>object_id</i>	integer
<i>design_id</i>	integer

ARGUMENTS

-hierarchical

Searches for nets level-by-level relative to the current instance. The full name of the object at a particular level must match the patterns. For example, if there is a net named block1/muxsel, a hierarchical search finds it using muxsel. You cannot use the **-hierarchical** option with the **-of_objects** option.

-filter *expression*

Filters the collection with the value of the *expression* argument. For any nets that match the specified criteria, the expression is evaluated based on the net's attributes. If the expression evaluates to true, the net is included in the result.

-quiet

Suppresses messages if no objects match. Syntax error messages are not suppressed.

-regexp

Uses the value of the *patterns* option as real regular expressions rather than as simple wildcard patterns.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. You can use this when searching for objects that contain the * and ? wildcard characters.

-top_net_of_hierarchical_group

Keeps only the top net of a hierarchical group. When more than one hierarchical net of the same group is specified (local nets at various hierarchical levels of the same physical net), only the net closest to the top of the hierarchy is saved with the collection. In the case of multiple nets at the same level, the first net specified is kept. This option is best used in combination with the **-segments** option and with a single net.

-segments

Returns all global segments for given nets. This modifies the initial search that matches *patterns* or *objects*. It is applied before the **-top_net_of_hierarchical_group** is determined. For each net, all global segments that are part of that net are added to the resulting collection. Global net segments are all those physically connected across all hierarchical boundaries. This option is best used with a single net. When you use the **-segments** option with the **-top_net_of_hierarchical_group** option, you can isolate the highest net segment of a physical net.

-boundary_type *boundary_type*

Specifies what to do when getting nets of boundary pins. This option requires that you also use the **-of_objects** option. The valid values for the *boundary_type* argument are **lower**, **upper**, and **both**. The **lower** value specifies the net inside the hierarchical block. The **upper** value specifies the net outside the hierarchical block. The **both** value specifies both nets. The option has no meaning for pins which are non-hierarchical and not on ILM cells.

Note that **upper** is less useful than the other values because getting pins of a hierarchical net returns the pin outside the hierarchical block, and a subsequent execution of the **get_nets** command would get the upper hierarchical net. Using **upper** on a hierarchical pin is the same as entirely omitting the option.

patterns

Matches net names against patterns. Patterns can include the wildcard characters * (asterisk) and ? (question mark). For more details about using and escaping wildcards, see the **wildcards** man page. The *patterns*, **-of_objects**, and **-object_id** arguments are mutually exclusive; you can use only one.

-of_objects *objects*

Creates a collection of nets connected to the specified objects. Each object is either a named pin, port, cell, shape, via, pin_guide, physical_bus, pin collection, port collection, cell collection, shape collection, via collection, pin_guide collection or physical_bus collection. The *patterns*, **-of_objects**, and **-object_id** arguments are mutually exclusive; you can specify only one. You cannot use the **-hierarchical** option with the **-of_objects** option.

-object_id *object_id*
 Specifies the Milkyway object ID of the net to include in the collection. To determine the *id* value, use the **get_attribute** command to return the value of the **object_id** attribute. The *patterns*, **-of_objects**, and **-object_id** arguments are mutually exclusive; you can use only one.

-all
 Includes power and ground nets.

-design_id *design_id*
 Specifies the Milkyway design ID of the design from which to create the collection of nets. The design is one of the currently open Milkyway designs (CELS). To determine the *id* value, use the **get_attribute** command to return the value of the **design_id** attribute. Use this option only when multiple Milkyway designs are open. If you do not use the **-design_id** option, the command creates a collection of cells from the current design.

DESCRIPTION

This command creates a collection of nets in the current design or in the design specified by the **-design_id** option (if you use it) relative to the current instance that match certain criteria. The command returns a collection if any nets match *patterns*, *objects*, or *object_id* and pass the filter (if specified). If no objects matched the criteria, the command returns an empty string.

If *patterns*, *objects*, and *object_id* fail to match any objects and the current design is not linked, the design automatically links.

Regular expression matching is the same as in the Tcl **regexp** command. When using **get_nets -regexp**, take care in how you quote the *patterns* and **-filter** options. Use rigid quoting with curly braces around regular expressions. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding the characters *.** to the beginning or end of the expressions, as needed.

You can use the **get_nets** command at the command prompt, or you can nest it as an argument to another command, such as **query_objects**. You can also assign the **get_nets** result to a variable.

When issued from the command prompt, **get_nets** behaves as though **query_objects** has been called to display the objects in the collection. By default, the command displays a maximum of 100 objects. You can change this maximum by using the **collection_result_display_limit** variable.

The implicit query property of **get_nets** provides a fast, simple way to display nets in a collection. However, if you want the flexibility provided by the **query_objects** options (for example, if you want to display the object class), use the **get_nets** command as an argument of the **query_objects** command.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example queries the nets that begin with NET in a block named block1. Although the output looks like a list, it is a display.

```
prompt> get_nets block1/NET*  
{ "block1/NET1QNX", "block1/NET2QNX" }
```

The following example shows that with a collection of pins, you can query the nets connected to those pins.

```
prompt> current_instance block1  
block1  
prompt> set pinsel [get_pins {o_reg1/QN o_reg2/QN}]  
{ "o_reg1/QN", "o_reg2/QN" }  
prompt> query_objects [get_nets -of_objects $pinsel]  
{ "NET1QNX", "NET2QNX" }
```

The following example shows that with a collection of cells, you can query the nets connected to those cells.

```
prompt> current_instance block1  
block1  
prompt> set cellsel [get_cells {o_reg1 o_reg2}]  
{ "o_reg1", "o_reg2" }  
prompt> query_objects [get_nets -of_objects $cellsel]  
{ "NET1QX", "NET1QNX", "NET1DX", "NET2QX", "NET2QNX", "NET2DX" }
```

SEE ALSO

```
collections(2)  
filter_collection(2)  
get_pins(2)  
query_objects(2)  
collection_result_display_limit(3)  
wildcards(3)
```

get_new_bounds

Creates a collection of bounds from the current design.

SYNTAX

```
collection get_new_bounds
[-quiet]
[-within rectangle]
[-filter expression]
[patterns]
```

Data Types

<i>rectangle</i>	string
<i>expression</i>	Boolean
<i>patterns</i>	list

ARGUMENTS

- quiet**
Suppresses the reporting of non-terminal (warning and informational) error messages.
- within *rectangle***
Creates a collection containing all bounds within the specified rectangle.
- filter *expression***
Filters the collection with the *expression* value, which must be a Boolean expression based on bound attributes.
- patterns***
Creates a collection containing the bounds whose names match the specified patterns. You can specify the patterns by using the following formats:
- *** (asterisk) allows all bounds names.
 - *hw** allows all horizontal bounds.
 - *hw object_id* allows all horizontal bounds with a specified object ID.
 - *vw** allows all vertical bounds.
 - *vw object_id* allows all vertical bounds with a specified object ID.
- By default, the command uses *** as the pattern.

DESCRIPTION

This command creates a collection of bounds that meet the selection criteria. It returns a collection handle if one or more bounds meet the selection criteria. If no bounds match the selection criteria, it returns an empty string. You can also use the **get_new_bounds** command as an argument to another command or assign its result to

a variable.

See the **collections** man page for information about working with collections.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates a new bound named *MB0* and creates a collection containing all bounds within the rectangle specified by coordinates {10 10 20 20}. It filters the collection with the expression value **type==soft**.

```
prompt> create_bound -name MB0 -bounding_box {10 10 20 20  
{"MB0"}
```

```
prompt> get_new_bounds -within {0 0 30 30} -filter "type==soft"  
{"MB0"}
```

SEE ALSO

`remove_bounds(2)`

get_object_fixed_edit

Gets the fixed state for the specified objects.

SYNTAX

```
fixed get_object_fixed_edit  
objects
```

Data Types

objects collection

ARGUMENTS

objects
List of objects for which to get the fixed state.

DESCRIPTION

This command gets the fixed state for the specified objects. The state is returned as a list of Boolean numbers.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example inverts the fixed state of all selected objects. (The **invert_list** function inverts the elements of a Boolean list.)

```
prompt> set_object_fixed_edit [get_selection]\  
[invert_list [get_object_fixed_edit [get_selection]]]
```

SEE ALSO

`set_object_fixed_edit(2)`

get_object_name

Returns the name of the object in a single-object collection.

SYNTAX

```
string get_object_name  
collection
```

Data Types

```
collection      string
```

ARGUMENTS

```
collection  
    Specifies the name of the collection that contains the single object whose  
    name is requested.
```

DESCRIPTION

This command returns the name of the object in a single-object collection. The command works only for collections that contain a single object. If the collection contains more than one object, the command issues an error message.

EXAMPLES

The following example returns the name `top` as the object contained in the collection returned by the **current_design** command.

```
prompt> get_object_name [current_design]  
Current design is 'top'.  
top
```

The following example issues an error message because the collection returned by the **all_outputs** command contains more than one object.

```
prompt> get_object_name [all_outputs]  
Error: Can't get object name of multiple objects;  
       Only a single object collection accepted.  (UITCL-100)
```

The following example gets names of objects in a multi-object collection, by using the **query_objects** command.

```
prompt> query_objects [all_outputs]  
{a, b, c, OP, Z}
```

SEE ALSO

`dc_shell(1)`
`collections(2)`
`query_objects(2)`

get_object_snap_type

Returns the snap type for the specified object class.

SYNTAX

```
string get_object_snap_type  
{-class snap_class | -enabled}
```

Data Types

snap_class string

ARGUMENTS

```
-class snap_class
    Specifies the object class to get the snap type for. One of:
    soft_macro           Soft Macro
    plan_group           Plan Group and Move Bound
    movebound            Movebound
    hard_macro           Hard Macro
    standard_cell        Standard Cell
    routing              Net shape, Via and Via Array
    shape                Polygon
    pin                  Top Level Pin, Soft Macro Pin, Pin Guide
    voltage_area         Voltage area
    placement_blockage   Placement Blockage
    wiring_keepout       Wiring Keepout
    rp_group             Relative Placement Group
    This option and the -enabled option are mutually exclusive; you must specify
    one of these options.

-enabled
    Return a Boolean value indicating whether snapping is enabled, instead of
    returning the snap type.
    This option and the -class option are mutually exclusive; you must specify
    one of these options.
```

DESCRIPTION

This command either returns the snap type for the specified object class (*-class* option) or a Boolean value indicating whether snapping is enabled (*-enabled* option).

The snap type is the one currently being used for all editing commands to snap objects of that class. Valid values for snap type are:

litho	Design grid (also known as minimum grid)
row	Row edge
row_tile	Row edge and unit tile
row_site	Row edge and placement site
mid_row	Mid row (mostly for soft macro placement)
mid_row_tile	Mid row and unit tile

mid_row_site	Mid row and placement site
flip_chip	Flip-chip grid (for flip chips)
unit_tile	Unit tile
wiretrack	Wire track (for pins, wires, paths, and terminals)
halfwiretrack	Mid wire track (for pins, wires, paths, and terminals)
user	User grid

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example stores the current snap type for soft macros in the type variable:

```
rompt> set type [get_object_snap_type -class soft_macro]
```

The following example enables snapping if snapping is disabled:

```
rompt> if {![get_object_snap_type -enabled]} \
  {set_object_snap_type -enabled 1}
```

SEE ALSO

```
move_objects(2)
resize_objects(2)
rotate_objects(2)
align_objects(2)
distribute_objects(2)
expand_objects(2)
flip_objects(2)
set_object_snap_type(2)
```

get_path_groups

Creates a collection of path groups from the current design.

SYNTAX

```
collection get_path_groups
[-quiet]
[-regexp]
[-nocase]
[-filter expression]
patterns
```

Data Types

<i>expression</i>	string
<i>patterns</i>	list

ARGUMENTS

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as a real regular expression rather than as a simple wildcard pattern. It also modifies the behavior of the `=~` and `!~` filter operators so that they compare with real regular expressions rather than with simple wildcard patterns.

-nocase

Makes matches case-insensitive when combined with the **-regexp** option. You can use the **-nocase** option only when you also use **-regexp**.

-filter *expression*

Filters the collection with the value of the *expression* argument. For any path groups that match the *patterns* option, the expression is evaluated based on the attributes of the path group. If the expression evaluates to true, the path group is included in the result.

patterns

Matches path group names against *patterns*. The *patterns* option can include the `*` (asterisk) and `?` (question mark) wildcard characters.

DESCRIPTION

This command creates a collection of path groups from the current design that match certain criteria. The command returns a collection handle (identifier) if any path groups match the value of the *patterns* option and pass the filter (if specified). If no objects match the criteria, the command returns an empty string.

Path groups control the optimization cost function and also affect timing reports.

You can use the **get_path_groups** command at the command prompt, or you can nest it as an argument to another command, such as **query_objects**. In addition, you can assign the **get_path_groups** result to a variable.

When issued from the command prompt, **get_path_groups** behaves as though **query_objects** has been called to display the objects in the collection. By default, a maximum of 100 objects is displayed. You can change this maximum by using the **collection_result_display_limit** variable.

The implicit query property of **get_path_groups** provides a fast, simple way to display cells in a collection. However, if you want the flexibility provided by the **query_objects** option (for example, if you want to display the object class), use **get_path_groups** as an argument to **query_objects**.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command uses information from the current scenario only.

SEE ALSO

```
collections(2)
foreach_in_collection(2)
group_path(2)
query_objects(2)
report_path_group(2)
collection_result_display_limit(3)
```

get_physical_buses

Returns a collection of physical buses from the current design. You can assign these physical buses to a variable or pass them into another command.

SYNTAX

```
collection get_physical_buses
[-quiet]
[-regex]
[-nocase]
[-exact]
[-filter expression]
patterns
  | -of_objects objects
  | -object_id id
```

Data Types

<i>expression</i>	string
<i>patterns</i>	list
<i>objects</i>	list
<i>id</i>	integer

ARGUMENTS

-quiet
Suppresses warning and error message if no objects match. Does not suppress syntax error messages.

-regex
Views the *patterns* option as a real regular expression rather than as a simple wildcard pattern. The **-regex** option also modifies the behavior of the =~ and !~ filter operators to behave as regular expressions rather than as simple wildcard patterns.

-nocase
Makes matches case-insensitive.

-exact
Disables simple pattern matching. Use this when searching for objects that contain the * (asterisk) wildcard character.

-filter *expression*
Filters the collection with the value of the *expression* argument. For any physical bus that match the *patterns* option, the *expression* is evaluated based on the attributes of the physical bus. If the *expression* evaluates to true, the command includes the physical bus in the result.

patterns
Matches physical bus names against the value of the *patterns* option. The *patterns* list can include the wildcard character * (asterisk). The *patterns*, **-of_objects**, and **-object_id** arguments are mutually exclusive; you must

specify only one.

-of_objects *objects*

Creates a collection of physical bus connected to the specified objects. In this case, each object is a net collection. The *patterns*, **-of_objects**, and **-object_id** arguments are mutually exclusive; you must specify only one.

-object_id *id*

Creates a collection of a physical bus with its Milkyway object ID specified. The *patterns*, **-of_objects**, and **-object_id** arguments are mutually exclusive; you must specify only one.

DESCRIPTION

This command creates a collection of physical buses from the current design that match specific criteria. The command returns a collection handle (identifier) if any physical bus matches the *patterns* option value and passes the filter, if specified. If no objects match the criteria, the command returns an empty string.

You can use the **get_physical_buses** command at the command prompt, nest it as an argument to another command (for example, **report_attributes**), or assign the **get_physical_buses** result to a variable.

When issued from the command prompt, **get_physical_buses** behaves as if you called the **query_objects** command to report the objects in the collection. By default, the command displays a maximum of 100 objects. You can change this maximum by using the **collection_result_display_limit** variable.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example returns a collection with one physical bus object whose name is prefixed with UPC.

```
prompt> get_physical_buses UPC*
{UPC_DATA}
```

The following example returns a collection with one physical bus object in which net UPC_DATA[0] is an element.

```
prompt> get_physical_buses -of_objects [get_nets UPC_DATA[0]]
{UPC_DATA}
```


SEE ALSO

`collections(2)`
`create_physical_bus(2)`
`query_objects(2)`
`remove_physical_bus(2)`
`report_physical_bus(2)`
`update_physical_bus(2)`
`collection_result_display_limit(3)`

get_physical_lib_cells

Creates a collection of library cells from libraries loaded into the tool.

SYNTAX

```
collection get_physical_lib_cells
[-filter expression]
[-quiet]
[-regexp]
[-nocase]
[-exact]
patterns | -of_objects objects
[-hsc separator]
```

Data Types

<i>expression</i>	string
<i>patterns</i>	list
<i>objects</i>	list
<i>separator</i>	string

ARGUMENTS

-filter *expression*

Filters the collection with the value of the *expression* argument. For any library cells that match the *patterns* option or the *objects* argument, the expression is evaluated based on the attributes of the library cell. If the expression evaluates to true, the command includes the library cell in the result.

-quiet

Suppresses information, warning, and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* option value as real regular expressions rather than as simple wildcard patterns. This option also modifies the behavior of the =~ and !~ filter operators to compare them with real regular expressions rather than with simple wildcard patterns. The fb-regexp and **-exact** options are mutually exclusive; you can use only one.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. Use this when searching for objects that contain the * (asterisk) and ? (question mark) wildcard characters. The fb-regexp and **-exact** options are mutually exclusive; you can use only one.

patterns

Matches library cell names against patterns. Patterns can include the * (asterisk) and ? (question mark) wildcard characters or regular expressions,

get_physical_lib_cells

based on the **-regexp** option. Patterns can also include collections of type `lib_cell`. The *patterns* and **-of_objects** options are mutually exclusive; you must specify one, but not both.

-of_objects *objects*

Creates a collection of library cells that are referenced by the specified cells or own the specified library pins. In this case, each object is either a named library pin or netlist cell, or a library pin collection or a netlist cell collection. **-of_objects** and *patterns* are mutually exclusive; you must specify only one.

-hsc *separator*

Specifies the hierarchy separator character that is used to create unambiguous names whenever a design uses a / (slash) within an object name. You can use the following characters as hierarchy separator characters: / (slash), @ (at sign), ^ (caret), # (pound sign), . (period), and | (vertical bar). The default is / (slash).

DESCRIPTION

This command creates a collection of library cells from libraries currently loaded into the tool that match certain criteria. The command returns a collection if any library cells match the value of the *patterns* option or the *objects* argument and pass the filter, if specified. If no objects match the criteria, the tool returns an empty string.

Regular expression matching in the **get_physical_lib_cells** command is the same as in the Tcl **regexp** command. When using **-regexp**, take be careful how you quote the *patterns* and filter expression. It is recommended that you use rigid quotation marks with curly braces around regular expressions.

Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search simply by adding *.** to the beginning or end of the expressions, as needed.

You can use the **get_physical_lib_cells** command at the command prompt, nest it as an argument to another command (for example, **query_objects**), or assign the **get_physical_lib_cells** result to a variable.

When issued from the command prompt, **get_physical_lib_cells** behaves as if you had called **query_objects** to display the objects in the collection. By default, the command displays a maximum of 100 objects. You can change this maximum by using the **collection_result_display_limit** variable.

The "implicit query" property of **get_physical_lib_cells** provides a fast, simple way to display cells in a collection. However, if you want the flexibility provided by the options of the **query_objects** command (for example, if you want to display the object class), use **get_physical_lib_cells** as an argument to **query_objects**.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example queries all library cells that are in the misc_cmos library and begin with AN2. Although the output looks like a list, it is just a display.

```
prompt> get_physical_lib_cells misc_cmos/AN2*  
{"misc_cmos/AN2", "misc_cmos/AN2P"}
```

The following example shows one way to find out the name of the library cell used by a specific cell.

```
prompt> get_physical_lib_cells\  
-of_objects [get_cells o_reg1  
{"misc_cmos/FD2"}
```

SEE ALSO

```
collections(2)  
filter_collection(2)  
get_cells(2)  
query_objects(2)  
collection_result_display_limit(3)
```

get_physical_lib_pins

Creates a collection of library cell pins from libraries.

SYNTAX

```
collection get_physical_lib_pins  
[-filter expression]  
[-quiet]  
[-regexp]  
[-nocase]  
[-exact]  
patterns | -of_objects objects  
[-hsc separator]
```

Data Types

<i>expression</i>	string
<i>patterns</i>	list
<i>objects</i>	list

ARGUMENTS

-filter *expression*

Filters the collection with *expression*. For any library cell pins that match *patterns* (or *objects*), the expression is evaluated based on the library cell pin's attributes. If the expression evaluates to true, the lib_pin is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. The **-regexp** and **-exact** arguments are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. The **-exact** and **-regexp** arguments are mutually exclusive.

patterns

Matches library cell pin names against patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the **-regexp** option. Patterns can also include collections of type lib_pin. The *patterns* and **-of_objects** arguments are mutually exclusive. You must specify one, but not both.

-of_objects *objects*

Creates a collection of library cell pins referenced by the specified netlist pins or owned by the specified library cells. In this case, each object is either a named library cell or netlist pin, or a library cell collection or netlist pin collection. The **-of_objects** and *patterns* arguments are mutually exclusive. You must specify one, but not both.

-hsc separator

Specifies the hierarchy separator character that is used to create unambiguous names whenever a design uses a slash (/) within object names. You can use the following characters as hierarchy separator characters: slash (/), at sign (@), caret (^), pound sign (#), period (.), and vertical bar (|), with the slash (/) being the default.

DESCRIPTION

The **get_physical_lib_pins** command creates a collection of library cell pins from currently loaded libraries that match certain criteria. The command returns a collection if any library cell pins match *patterns* or *objects* and pass the filter (if specified). If no objects matched the criteria, the empty string is returned.

Regular expression matching is the same as in the **regexp** Tcl command. When using **-regexp**, take care in the way you quote the *patterns* and filter *expression*. Using rigid quoting with curly braces around regular expressions is recommended. Regular expressions are always anchored. The expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding *".*"* to the beginning or end of the expressions as needed.

You can use the **get_physical_lib_pins** command at the command prompt, or you can nest it as an argument to another command, such as **query_objects**. In addition, you can assign the **get_physical_lib_pins** result to a variable.

When issued from the command prompt, **get_physical_lib_pins** behaves as though **query_objects** had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed. You can change this maximum using the **collection_result_display_limit** variable.

The implicit query property of **get_physical_lib_pins** provides a fast, simple way to display cells in a collection. However, if you want the flexibility provided by the **query_objects** options (for example, if you want to display the object class), use **get_physical_lib_pins** as an argument to **query_objects**.

For information about collections and the querying of objects, see the **collections** command man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example queries all pins of the AN2 library cell in the misc_cmos library. Although the output looks like a list, it is just a display.

```
prompt> [get_physical_lib_pins misc_cmos/AN2/*  
{"misc_cmos/AN2/A", "misc_cmos/AN2/B", "misc_cmos/AN2/Z"}
```

The following example shows one way to determine how the library pin is used by a particular pin in the netlist.

```
prompt> get_physical_lib_pins -of_objects o_reg1/Q  
{"misc_cmos/FD2/Q"}
```

SEE ALSO

```
collections(2)  
filter_collection(2)  
get_physical_libs(2)  
get_physical_lib_cells(2)  
query_objects(2)  
collection_result_display_limit(3)
```

get_physical_libs

Creates a collection of libraries.

SYNTAX

```
collection get_physical_libs
[-filter expression]
[-quiet]
[-regex [-nocase]]
[-exact]
patterns | -of_objects objects
```

Data Types

<i>expression</i>	string
<i>patterns</i>	list
<i>objects</i>	list

ARGUMENTS

-filter *expression*

Filters the collection with the value of the *expression* argument. For any libraries that match the *patterns* option or *objects* argument, the expression is evaluated based on the library's attributes. If the expression evaluates to true, the library is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regex

Uses the value of the *patterns* argument as a real regular expression rather than as a simple wildcard pattern. This option also modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions, rather than with simple wildcard patterns. The **-regex** and **-exact** options are mutually exclusive; you can use only one.

-nocase

Makes matches case-insensitive when you use it with the **-regex** option. You can use the **-nocase** option only when you also use the **-regex** option.

-exact

Disables simple pattern matching. You can use this option when searching for objects that contain the * (asterisk) and ? (question mark) wildcard characters. The **-regex** and **-exact** options are mutually exclusive; you can use only one.

patterns

Matches library names against patterns. Patterns can include the wildcard characters the * (asterisk) and ? (question mark) wildcard characters or regular expressions, depending on the use of the **-regex** option. Patterns can also include collections of lib type. The *patterns* and **-of_objects** options

are mutually exclusive; you must use only one.

-of_objects *objects*

Creates a collection of libraries that contain the specified objects. In this case, each object is either a named library cell or a library cell collection. The *patterns* and **-of_objects** options are mutually exclusive; you must use only one.

DESCRIPTION

This command creates a collection of libraries from those currently loaded that match certain criteria. The command returns a collection if any libraries match the *patterns* option and pass the filter (if specified). If no objects match the criteria, the command returns an empty string.

Regular expression matching is the same as in the **regexp** Tcl command. When using the **-regexp** option, be careful how you use quotation marks in the *patterns* option and *expression* argument. It is recommended that you use curly braces around regular expressions. Regular expressions are always anchored. The expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search simply by adding *.** to the beginning or end of an expression, as needed.

You can use the **get_physical_libs** command at the command prompt or nest it as an argument to another command (such as **query_objects**). You can also assign the **get_physical_libs** result to a variable.

When issued from the command prompt, **get_physical_libs** behaves as if you had called **query_objects** to display the objects in the collection. By default, a maximum of 100 objects is displayed. You can change the maximum by using the **collection_result_display_limit** variable.

The implicit query property of **get_physical_libs** provides a fast, simple way to display cells in a collection. However, if you want the flexibility provided by the **query_objects** options (for example, if you want to display the object class), use **get_physical_libs** as an argument to **query_objects**.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example queries all loaded libraries. Although the output looks like a list, it is just a display. You can use the **list_libraries** command to get a complete listing of libraries.

```
prompt> get_physical_libs *
```

```
{"misc_cmos", "misc_cmos_io"}
```

The following example shows that you can remove the libraries of a given library collection, but you cannot remove libraries if they are referenced by a design.

```
prompt> remove_reference_library [get_physical_libs misc*]  
Removing library misc_cmos...  
Removing library misc_cmos_io...
```

SEE ALSO

```
collections(2)  
filter_collection(2)  
query_objects(2)  
collection_result_display_limit(3)
```

get_pin_guides

Retrieves a collection of pin guides existing in the current design.

SYNTAX

```
collection get_pin_guides
[-filter expression]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-object_id object_id]
[patterns]
```

Data Types

<i>expression</i>	string
<i>object_id</i>	integer
<i>patterns</i>	string

ARGUMENTS

-filter *expression*

Filters the collection with the value of the *expression* argument. For pin guides that match the specified patterns (or objects), the expression is evaluated based on the attributes of the pin guides. If the expression evaluates to true, the pin guide is included in the result. This option works the same way as the **filter_collection** command works. The **-regexp** and **-nocase** options are applied in the same way as in the **filter_collection** command. See the **filter_collection** man page for more information on how to use a **-filter** option in general.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Uses the value of the *patterns* argument as a real regular expression rather than as a simple wildcard pattern.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. Use this option when searching for objects that contain the "*" (asterisk) wildcard as an actual character.

-object_id *object_id*

Specifies the object ID of one pin guide.

patterns

Matches pin guide names against patterns. Pattern matching is case sensitive

unless you use the **-nocase** option. The default is * (asterisk).

DESCRIPTION

This command retrieves a collection of pin guides existing in the current design. It returns a collection of all pin guides if no arguments are given to the command. The absence of any arguments is equivalent to using the command **get_pin_guides ***.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets all pin guides in the design.

```
prompt> set pin_guides [get_pin_guides]
```

The following example gets pin guides that match the regular expression shown.

```
prompt> set pin_guides [get_pin_guides -regexp "[Pp].*"]
```

SEE ALSO

```
create_pin_guide(2)  
filter_collection(2)  
report_pin_guides(2)
```

get_pin_shapes

Creates a collection of pin shapes that matches the criteria.

SYNTAX

```
collection get_pin_shapes
[-within region
 | -intersect region
 | -touching region
 | -at point]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-filter expression]
[patterns | -of_objects objects]
```

Data Types

<i>region</i>	list
<i>point</i>	string
<i>expression</i>	string
<i>patterns</i>	list
<i>objects</i>	list

ARGUMENTS

-within *region*

Creates a collection containing all pin shapes within the specified region. The region boundary can be rectangle or polygon. The valid format of a rectangle is `{{llx lly} {urx ury}}` or `{llx lly urx ury}`, which specifies the lower-left and upper-right corners of the rectangle. The valid format of a polygon is: `{{x1 y1} {x2 y2} ... {xN yN} {x1 y1}}`, and each `{x y}` pair specifies one point of the input polygon. Besides, a list of one polygon is also supported as input for this option, with the format: `{{{x1 y1} {x2 y2} ... {xN yN} {x1 y1}}}`. Pay attention that the valid polygon is rectilinear, so the adjacent points have one same coordinate. The coordinate unit is specified in technology file (usually it is micron). The **-within**, **-intersect**, **-touching**, and **-at** options are mutually exclusive; you can specify only one.

-intersect *region*

Creates a collection containing all pin shapes intersect the specified region. The region boundary can be rectangle or polygon. The valid format of a rectangle is `{{llx lly} {urx ury}}` or `{llx lly urx ury}`, which specifies the lower-left and upper-right corners of the rectangle. The valid format of a polygon is: `{{x1 y1} {x2 y2} ... {xN yN} {x1 y1}}`, and each `{x y}` pair specifies one point of the input polygon. Besides, a list of one polygon is also supported as input for this option, with the format: `{{{x1 y1} {x2 y2} ... {xN yN} {x1 y1}}}`. Pay attention that the valid polygon is rectilinear, so the adjacent points have one same coordinate. The coordinate unit is specified in technology file (usually it is micron).

The **-within**, **-intersect**, **-touching**, and **-at** options are mutually exclusive; you can specify only one.

-touching *region*

Creates a collection containing all pin shapes touching the specified region. The region boundary can be rectangle or polygon. The valid format of a rectangle is `{{llx lly} {urx ury}}` or `{llx lly urx ury}`, which specifies the lower-left and upper-right corners of the rectangle. The valid format of a polygon is: `{{x1 y1} {x2 y2} ... {xN yN} {x1 y1}}`, and each `{x y}` pair specifies one point of the input polygon. Besides, a list of one polygon is also supported as input for this option, with the format: `{{{x1 y1} {x2 y2} ... {xN yN} {x1 y1}}}`. Pay attention that the valid polygon is rectilinear, so the adjacent points have one same coordinate. The coordinate unit is specified in technology file (usually it is micron). The **-within**, **-intersect**, **-touching**, and **-at** options are mutually exclusive; you can specify only one.

-at *point*

Creates a collection containing all pin shapes that overlap the point specified by the coordinates `{x y}`. The coordinate unit is specified in technology file (usually it is micron). The **-within**, **-intersect**, **-touching**, and **-at** options are mutually exclusive; you can specify only one.

-quiet

Suppresses the reporting of non-terminal (warning and informational) messages.

-regexp

Uses the *patterns* argument as real regular expressions rather than as simple wildcard patterns. By default, this option is off.

-nocase

Makes the matches case-insensitive. By default, this option is off.

-exact

Disables simple pattern matching. Use this when searching for objects that contain the `*` (asterisk) wildcard character. By default, this option is off.

-filter *expression*

Filters the collection with the value of *expression*, which must be a Boolean expression based on pin shape attributes. You can determine the pin shape attributes using the **list_attributes** command. If the expression evaluates to **true**, the pin shape is included in the collection. By default, this option is off.

patterns

Matches the pin shape names against the patterns in the current design. The *patterns* and **-of_objects** options are mutually exclusive. If you do not specify either the *patterns* and **-of_objects** options, the command defaults to using `*` as the pattern.

-of_objects *objects*

Creates a collection containing the pin shapes connected to the specified objects. Each object is a cell or pin. The *patterns* and **-of_objects** options are mutually exclusive. If you do not

specify either the *patterns* and *-of_objects* options, the command defaults to using *** as the pattern. By default, this option is off.

DESCRIPTION

This command creates a collection of pin shapes by selecting pin shapes from the current design that meet the selection criteria. It returns a collection handle if one or more pin shapes meet the selection criteria. If no pin shapes match the selection criteria, it returns an empty string.

When using **-within**, **-touching**, or **-intersect**, note that the behavior will be different for the different format of the specified region. A *pin_shape* object has its boundary and *bbox*, sometimes they are different. When the specified region is in rectangle format, we will search objects by their *bbox*; when the specified region is in polygon format, we will search objects by their boundary. And searching by *bbox* is faster than searching by boundary.

You can use the **get_pin_shapes** command at the command prompt, use it as an argument nested in another command, or assign its result to a variable.

See the **collections** command man page for information about working with collections.

EXAMPLES

The following example shows that in a collection of pin, you can query the pin shapes connected to the specified pin. You can also fetch the pin shape on metal6 layer using the **-filter** option.

```
prompt> set pin [get_pins tc4/sram_target_2/sram_Q[4]]
{tc4/sram_target_2/sram_Q[4]}
prompt> get_pin_shapes -of_objects $pin
{tc4/sram_target_2/sram_Q[4]}
prompt> get_pin_shapes -filter {layer =~ "M6"} -of_objects $pin
{tc4/sram_target_2/sram_Q[4]}
```

The following example shows that in a soft macro, you can select the pin shapes in it.

```
prompt> get_pin_shapes -of_objects \
tc4st_bus_1/tc4stbus_sram_target_2/sram_interface_16b
{tc4st_bus_1/tc4stbus_sram_target_2/sram_interface_16b/sram_Q[4] tc4st_bus_1/
tc4stbus_sram_target_2/sram_interface_16b/test_so_new tc4st_bus_1/
tc4stbus_sram_target_2/sram_interface_16b/test_so_new 1}
```

The following example fetches an EEQ pin shape with its name.

```
prompt> get_pin_shapes {"tc4st_bus_1/tc4stbus_sram_target_2/sram_interface_16b/
test_so_new 1"}
{tc4st_bus_1/tc4stbus_sram_target_2/sram_interface_16b/test_so_new 1}
```

The following example gets all pin shapes within a specified rectangle.

```
prompt> get_pin_shapes * \  
-within {{4 20} {10 30}}  
{I_BD_70/gnd I_BD_70/vdd I_BD_70/reset}
```

The following example gets all pin shapes intersect specified polygon:

```
prompt>get_pin_shapes -intersect \  
{{30 20} {50 20} {50 30} {40 30} {40 40} {30 40} {30 20}}  
{STACK_BLK/MEM_reg_1__0_/VSS STACK_BLK/MEM_reg_9__0_/VDD}
```

SEE ALSO

```
collections(2)  
filter_collection(2)  
get_pins(2)  
query_objects(2)  
collection_result_display_limit(3)
```


get_pins

Creates a collection of pins that match the specified criteria.

SYNTAX

```
collection get_pins
[-hierarchical]
[-filter expression]
[-quiet]
[-regexp [-nocase] | -exact]
[patterns
 | -of_objects objects [-leaf]
 | -object_id object_id]
[-all]
[-hsc separator]
```

Data Types

<i>expression</i>	string
<i>patterns</i>	list
<i>objects</i>	collection
<i>object_id</i>	integer

ARGUMENTS

-hierarchical

Searches for pins level-by-level, relative to the current instance. The full name of the object at a particular level must match the patterns. The search is similar to that of the UNIX **find** command. For example, if there is a pin named block1/adder/D[0], a hierarchical search finds it by using adder/D[0]. You cannot use the **-hierarchical** option if you use the **-of_objects** option.

-filter *expression*

Filters the collection with the value of the *expression* argument. For any pins that match the specified criteria, the expression is evaluated based on the pin's attributes. If the expression evaluates to true, the pin is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Uses the value of the *patterns* argument as a real regular expression rather than as a simple wildcard pattern. This option also modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions, rather than with simple wildcard patterns. The **-regexp** and **-exact** options are mutually exclusive; you can use only one.

-nocase

Makes matches case-insensitive when you use it with the **-regexp** option. You

can use the **-nocase** option only when you also use the **-regexp** option.

-exact

Disables simple pattern matching. You can use this option when searching for objects that contain the * (asterisk) and ? (question mark) wildcard characters.

The **-regexp** and **-exact** options are mutually exclusive; you can use only one.

patterns

Matches pin names against patterns. Patterns can include the wildcard characters * (asterisk) and ? (question mark). For more details about using and escaping wildcards, see the **wildcards** man page.

The *patterns*, **-of_objects**, and **-object_id** arguments are mutually exclusive; you can specify only one.

-of_objects *objects*

Creates a collection of pins connected to the specified objects. Each object can be a cell, net, pin_guide or pin_shape.

The *patterns*, **-of_objects**, and **-object_id** arguments are mutually exclusive; you can specify only one. In addition, you cannot use **-hierarchical** if you use the **-of_objects** option.

-leaf

You can use this option only if you also use the **-of_objects** option. For any nets in the *objects* argument, only pins on leaf cells connected to those nets are included in the collection. In addition, hierarchical boundaries are crossed to find pins on leaf cells.

-object_id *object_id*

Specifies the Milkyway object ID of the pin to include in the collection. To determine the *object_id* value, use the **get_attribute** command to return the value of the **object_id** attribute.

The *patterns*, **-of_objects**, and **-object_id** arguments are mutually exclusive; you can specify only one.

-all

Includes power and ground pins.

-hsc *separator*

Specifies the hierarchy separator character that is used for this command to create unambiguous names when an object name contains a slash (/).

You can use the following characters as hierarchy separator characters: / (slash), @ (at sign), ^ (caret), # (pound sign), . (period), and | (vertical bar). The default is /.

DESCRIPTION

The **get_pins** command creates a collection of pins in the current design, relative to the current instance, that match certain criteria. The command returns a collection if any pins match the *patterns*, *objects*, or *object_id* argument and pass the filter, if specified. If no objects match the criteria, the command returns an empty string.

When you use the **get_pins** command with the **-of_objects** option, it searches for pins connected to any cells or nets specified in the *objects* argument.

There are two variations of pins that are considered for net objects. By default, the command considers only pins connected to the net at the same hierarchical level. When you use the **-leaf** option, the command considers only pins connected to the net that are on leaf cells. In this case, hierarchical boundaries are crossed to find pins on leaf cells. The **-leaf** option has no effect on the pins of cells.

If the *patterns*, *objects*, and *object_id* arguments do not match any objects, and the current design is not linked, the design automatically links.

Regular expression matching is the same as in the **regexp** Tcl command. When using the **-regexp** option, be careful how you use quotation marks in the *patterns* and *expression* arguments. You should use curly braces ({}) around regular expressions. Regular expressions are always anchored. The expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding *.** to the beginning or end of an expression, as needed.

You can use the **get_pins** command at the command prompt or nest it as an argument to another command, such as **query_objects**. You can also assign the **get_pins** result to a variable.

When issued from the command prompt, **get_pins** behaves as if you had called the **query_objects** command to display the objects in the collection. By default, a maximum of 100 objects is displayed. You can change the maximum by using the **collection_result_display_limit** variable.

The implicit query property of **get_pins** provides a fast, simple way to display pins in a collection. However, if you want the flexibility provided by the **query_objects** options (for example, if you want to display the object class), use **get_pins** as an argument to **query_objects**.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example queries the CP pins of cells that begin with *o*. Although the output looks like a list, it is just a display.

```
prompt> get_pins o*/CP
{o_reg1/CP o_reg2/CP o_reg3/CP o_reg4/CP}
```

The following example shows that given a collection of cells, you can query the pins connected to those cells:

```
prompt> set csel [get_cells o_reg1]
{o_reg1}
```

```
prompt> query_objects [get_pins -of_objects $csel]  
{o_reg1/D o_reg1/CP o_reg1/CD o_reg1/Q o_reg1/QN}
```

The following example shows the difference between getting local pins of a net and leaf pins of net. In this example, NET1 is connected to i2/aP and reg1/QN. Cell i2 is hierarchical. Within cell i2, port a is connected to U1/A and U2/A.

```
prompt> get_pins -of_objects [get_nets NET1]  
{i2/a reg1/QN}
```

```
prompt> get_pins -leaf -of_objects [get_nets NET1]  
{i2/U1/A i2/U2/A reg1/QN}
```

The following example shows how to create a clock using a collection of pins:

```
prompt> create_clock -period 8 -name CLK [get_pins o_reg*/CP]  
1
```

SEE ALSO

```
collections(2)  
create_clock(2)  
filter_collection(2)  
get_cells(2)  
query_objects(2)  
collection_result_display_limit(3)  
wildcards(3)
```

get_placement_area

Returns a list of coordinates for the current core placement area.

SYNTAX

```
string get_placement_area
```

ARGUMENTS

None.

DESCRIPTION

Returns a list of coordinates for the current core placement area. Returns a list of float numbers in the order of lower-left corner coordinates and upper-right corner coordinates. If the core area is not defined in the current design, the command returns an error message.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows how to get the core placement area coordinates in the current design.

```
prompt> get_placement_area
```

SEE ALSO

```
create_placement(2)
```

get_placement_blockages

Creates a collection of placement blockages from the current design.

SYNTAX

```
collection get_placement_blockages
[-within region
  | -touching region
  | -intersect region]
[-filter expression]
[-quiet]
[-type hard | soft | pin | hard_macro | partial]
[patterns]
```

Data Types

<i>region</i>	list
<i>expression</i>	string
<i>patterns</i>	list

ARGUMENTS

-within *region*

Creates a collection containing all blockages within the specified region. The region boundary can be rectangle or polygon. The valid format of a rectangle is `{{llx lly} {urx ury}}` or `{llx lly urx ury}`, which specifies the lower-left and upper-right corners of the rectangle. The valid format of a polygon is: `{{x1 y1} {x2 y2} ... {xN yN} {x1 y1}}`, and each `{x y}` pair specifies one point of the input polygon. Note that the valid polygon is rectilinear, so the adjacent points have one same coordinate. The coordinate unit is specified in technology file (usually it is microns). The **-within**, **-intersect**, and **-touching** options are mutually exclusive; you can specify only one.

-touching *region*

Creates a collection containing all blockages touching the specified region. The region boundary can be rectangle or polygon. The valid format of a rectangle is `{{llx lly} {urx ury}}` or `{llx lly urx ury}`, which specifies the lower-left and upper-right corners of the rectangle. The valid format of a polygon is: `{{x1 y1} {x2 y2} ... {xN yN} {x1 y1}}`, and each `{x y}` pair specifies one point of the input polygon. Note that the valid polygon is rectilinear, so the adjacent points have one same coordinate. The coordinate unit is specified in technology file (usually it is microns). The **-within**, **-intersect**, and **-touching** options are mutually exclusive; you can specify only one.

-intersect *region*

Creates a collection containing all blockages that intersect the specified region. The region boundary can be rectangle or polygon. The valid format of a rectangle is `{{llx lly} {urx ury}}` or `{llx lly urx ury}`, which specifies the lower-left and upper-right corners of the rectangle. The valid format of a polygon is `{{x1 y1} {x2 y2} ... {xN yN} {x1 y1}}`, and

each {x y} pair specifies one point of the input polygon. Note that the valid polygon is rectilinear, so the adjacent points have one same coordinate. The coordinate unit is specified in the technology file. It is in microns usually.

The **-within**, **-intersect**, and **-touching** options are mutually exclusive; you can specify only one.

-filter *expression*

Filters the collection with *expression*. For any placement blockages that match the *patterns* argument, the *expression* is evaluated based on the attributes of the placement blockages. If the *expression* evaluates to *true*, the placement blockage is included in the result.

-quiet

Suppresses warning and error messages.

-type hard | soft | pin | hard_macro | partial

Indicates the type of placement blockage. By default, this command matches all kinds of placement blockages.

patterns

Specifies the placement blockages. The *patterns* can be a collection handle of blockages or other formats, similar to the following:

- * (asterisk) indicates all placement blockages
- xx* indicates any placement blockage whose name begins with xx
- pb#77 indicates one placement blockage that does not yet have a name and its *object_id* is 77
- pb#* indicates any placement blockage that does not yet have a name. If this argument is not specified, this command uses * (asterisk) as the *pattern*.

DESCRIPTION

This command creates a collection of placement blockages that meet the selection criteria. It returns a collection of placement blockages if one or more blockages meet the selection criteria. If no blockages match the selection criteria, it returns an empty string. You can use the **get_placement_blockages** command as an argument to another command or assign its result to a variable. Refer to the example below for details.

See the **collections** man page for information about working with collections.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following examples show some uses of this command to create placement blockages:

```
prompt> get_placement_blockages * -within {{2 2} {25 25}}  
{"PB#5389"}
```

```
prompt> get_placement_blockages * -filter "area > 1000"  
{"PB#4683"}
```

SEE ALSO

```
collections(2)  
create_placement_blockage(2)  
get_attribute(2)  
get_route_guides(2)  
query_objects(2)  
remove_placement_blockage(2)  
remove_route_guide(2)
```


get_plan_groups

Returns a collection of plan groups from the current design.

SYNTAX

```
collection get_plan_groups
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-filter expression]
patterns | -of_objects objects
```

Data Types

<i>expression</i>	string
<i>patterns</i>	list
<i>objects</i>	list

ARGUMENTS

-quiet

Suppresses warning and error messages, if no objects match. Does not suppress syntax error messages.

-regexp

Views the *patterns* argument as a real regular expression rather than a simple wildcard pattern. This option also modifies the behavior of the =~ and !~ filter operators to behave as regular expressions rather than simple wildcard patterns.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. For use when searching for objects that contain the * wildcard character.

-filter *expression*

Filters the collection with *expression*. For any plan group that match the *patterns* option, the expression is evaluated based on the plan group's attributes. If the expression evaluates to *true*, the plan group is included in the result.

patterns

Matches plan group names against *patterns*. The *patterns* list can include the wildcard character "*".

-of_objects *objects*

Creates a collection of plan groups connected to the specified objects. In this case, each object is a cell collection. The **-of_objects** and *patterns* arguments are mutually exclusive; you must specify one, but not both.

DESCRIPTION

The **get_plan_groups** command creates a collection of plan group from the current design that match certain criteria. The command returns a collection handle (identifier) if any plan group match the *patterns* and pass the filter (if specified). If no objects match the criteria, the empty string is returned.

You can use the **get_plan_groups** command at the command prompt, or you can nest it as an argument to another command (for example, **report_attributes**). In addition, you can assign the **get_plan_groups** result to a variable.

When issued from the command prompt, **get_plan_groups** behaves as though **report_attributes** has been called to report the objects in the collection. By default, a maximum of 100 objects are displayed. You can change this maximum using the variable **collection_result_display_limit**.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following examples get all the plan groups with the name starting with timer.

```
prompt> get_plan_groups timer*
```

SEE ALSO

```
collections(2)
foreach_in_collection(2)
query_objects(2)
create_plan_groups(2)
remove_plan_groups(2)
collection_result_display_limit(3)
```

get_polygon_area

Calculate the area of the input polygon.

SYNTAX

```
double get_polygon_area  
polygon
```

Data Types

polygon list

ARGUMENTS

polygon

A list of points that represents a polygon and the format is like this: {{x1 y1} {x2 y2} ... {xN yN} {x1 y1}}. The coordinate unit is specified in technology file (usually it is micron).

DESCRIPTION

This command returns the area covered by the input polygon. The input for this command is a list of points which represents a polygon; and the returned value represents the area of the input polygons. Before this command is used, the library should be opened.

EXAMPLES

The following command returns a value represents the area of the input polygons.

```
prompt>get_polygon_area {{5 5} {20 5} {20 20} {15 20} {15 10} \  
{10 10} {10 15} {5 15} {5 5}}  
150.0
```

SEE ALSO

```
convert_to_polygon(2)  
resize_polygon(2)
```

get_ports

Creates a collection of ports from the current design.

SYNTAX

```
collection get_ports
[-quiet]
[-regex]
[-nocase]
[-exact]
[-filter expression]
patterns
  | -of_objects objects
  | -object_id object_id
[-all]
```

Data Types

<i>expression</i>	string
<i>patterns</i>	list
<i>objects</i>	list
<i>object_id</i>	integer

ARGUMENTS

-quiet
Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regex
Uses the *patterns* argument as real regular expressions rather than simple wildcard patterns.

-nocase
Makes matches case-insensitive.

-exact
Disables simple pattern matching. This is used when searching for objects that contain the * (asterisk) and ? (question mark) wildcard characters.

-filter *expression*
Filters the collection with *expression*. For any ports that match the specified criteria, the expression is evaluated based on the port's attributes. If the expression evaluates to true, the cell is included in the result.

patterns
Matches port names against patterns. Patterns can include the wildcard characters * (asterisk) and ? (question mark). For more details about using and escaping wildcards refer to the **wildcards** man page. The *patterns*, **-of_objects**, and **-object_id** arguments are mutually exclusive; you can specify only one of these arguments.

-of_objects *objects*

Creates a collection of ports connected to the specified objects. Each object can be a named net, terminal, bound, net collection, terminal collection, bound collection. The *patterns*, **-of_objects**, and **-object_id** arguments are mutually exclusive; you can specify only one of these arguments.

-object_id *object_id*

Specifies the Milkyway object ID of the port to include in the collection. To determine the *id* value, use the **get_attribute** command to return the value of the **object_id** attribute. The *patterns*, **-of_objects**, and **-object_id** arguments are mutually exclusive; you can specify only one of these arguments.

-all

Includes power and ground ports.

DESCRIPTION

The **get_ports** command creates a collection of ports in the current design that match certain criteria. The command returns a collection if any ports match *patterns*, *objects*, or *object_id* and pass the filter (if specified). If no objects match the criteria, the empty string is returned.

Regular expression matching is the same as in the Tcl **regexp** command. When using **-regexp**, take care in the way you quote the *patterns* and filter *expression*. Use rigid quoting with curly braces around regular expressions. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can expand the search by adding *".*"* to the beginning or end of the expressions as needed.

You can use the **get_ports** command at the command prompt, or you can nest it as an argument to another command, such as **query_objects**. In addition, you can assign the **get_ports** result to a variable.

When issued from the command prompt, **get_ports** behaves as though **query_objects** has been called to display the objects in the collection. By default, a maximum of 100 objects is displayed. You can change this maximum using the **collection_result_display_limit** variable.

The implicit query property of **get_ports** provides a fast, simple way to display ports in a collection. However, if you want the flexibility provided by the **query_objects** options (for example, if you want to display the object class), use **get_ports** as an argument to **query_objects**.

For information about collections and the querying of objects, see the **collections** man page. In addition, refer to the man pages for the **all_inputs** and **all_outputs** commands, which also create collections of ports.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example queries all input ports beginning with *mode*. Although the output looks like a list, it is just a display.

```
prompt> get_ports "mode*" -filter {@port_direction == in}
{"mode[0]", "mode[1]", "mode[2]"}
```

The following example sets the driving cell for ports beginning with *in* to an FD2.

```
prompt> set_driving_cell -cell FD2 -library my_lib [get_ports in*]
```

The following example reports ports connected to nets matching the pattern *bidir**.

```
prompt> report_port [get_ports -of_objects [get_nets "bidir*"]]
```

The following example get the ports connected to terminals matching the pattern "CC*".

```
prompt> get_ports -of_objects [get_terminals "CC*"]
{CC CCEN}
```

SEE ALSO

```
all_inputs(2)
all_outputs(2)
collections(2)
filter_collection(2)
query_objects(2)
collection_result_display_limit(3)
find_allow_only_non_hier_ports(3)
wildcards(3)
```

get_power_domains

Creates a collection of power domains that meet the specified criteria.

SYNTAX for UPF Mode

```
collection get_power_domains

[-filter expression]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[[patterns] [-hierarchical] | -of_objects objects]
```

Data Types for UPF Mode

<i>expression</i>	string
<i>patterns</i>	list
<i>objects</i>	list

SYNTAX for Non-UPF Mode

```
collection get_power_domains

[-filter expression]
[-quiet]
[-regexp]
[-nocase]
[patterns]
[-of_objects objects]
```

Data Types for Non-UPF Mode

<i>expression</i>	string
<i>patterns</i>	list
<i>cells</i>	collection

ARGUMENTS

-filter *expression*
Filters the collection with *expression*. For any power domains that match *patterns*, the expression is evaluated based on the power domain's attributes. If the expression evaluates to true, the power domain is included in the result.

-quiet
Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Uses the *patterns* argument as real regular expressions rather than simple wildcard patterns.

-nocase

Makes matches case-insensitive.

-exact

This option is available only in UPF mode. Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters.

-hierarchical

This option is available only in UPF mode. Searches for power domains in the current scope and all of its descendant scopes. If you specify this option, you can specify only simple names in the *patterns* argument; the name cannot contain hierarchy delimiter characters.

If you do not specify this option, the tool searches for power domains only in the current scope and you can use hierarchy delimiter characters in the *patterns* argument to specify hierarchical power domain names relative to the current scope.

You cannot specify this option when you use the **-of_objects** option.

patterns

Matches power domain names against patterns. Patterns can include the wildcard characters * (asterisk) and ? (question mark). For more details about using and escaping wildcards, refer to the **wildcards** man page.

The *patterns* argument and **-of_objects** option are mutually exclusive. If you do not specify either, the tool uses "".

-of_objects *objects*

Returns a collection of power domains associated with the specified objects. In UPF mode, the objects can be cells, supply nets, supply ports, or power switches. In non-UPF mode, the objects must be cells.

The *patterns* argument and **-of_objects** option are mutually exclusive. If you do not specify either, the tool uses "".

DESCRIPTION

The **get_power_domains** command creates a collection of power domains in the current design, that match certain criteria. If no power domains match the criteria, an empty string is returned. In UPF mode, if no power domains match the criteria and the design is not linked, the tool automatically links the design.

Regular expression matching is the same as in the Tcl **regexp** command. When using **-regexp**, take care in the way you quote the patterns and filter expression. Use rigid quoting with curly braces around regular expressions. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding ".*" to the beginning or end of the expressions as needed.

You can use the **get_power_domains** command at the command prompt, or you can nest it as an argument to another command, such as **report_power_domain**. In addition, you can assign the **get_power_domains** result to a variable.

When issued from the command prompt, **get_power_domains** behaves as though **query_objects** has been called to display the objects in the collection. By default, a maximum of 100 objects is displayed. You can change this maximum by using the **collection_result_display_limit** variable.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following non-UPF mode example creates power domains, then queries them.

```
prompt> create_power_domain TOP_DOMAIN
1
prompt> create_power_domain SUB_DOMAIN -power_down \
        -power_down_ctrl [get_nets pd_ctrl] \
        -power_down_ack [get_nets pd_ack] \
        -object_list [get_cells mid1]
1
prompt> get_power_domains
{TOP_DOMAIN SUB_DOMAIN}
prompt> get_power_domains TOP*
{TOP_DOMAIN}
```

The following UPF mode example gets all power domains under the current scope.

```
prompt> get_power_domains -hierarchical {PD1 SUB_INST/PD2}
```

SEE ALSO

```
create_power_domain(2)
remove_power_domain(2)
report_power_domain(2)
```

get_power_switches

Creates a collection of power switches that meet the specified criteria. This command is supported only in UPF mode.

SYNTAX

```
collection get_power_switches
[-filter expression]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[patterns] [-hierarchical]
    | -of_objects objects
```

Data Types

<i>expression</i>	string
<i>patterns</i>	list
<i>objects</i>	list

ARGUMENTS

-filter *expression*
Filters the collection with *expression*. For any power switches that match *patterns*, the expression is evaluated based on the power switch's attributes. If the expression evaluates to true, the power switch is included in the result.

-quiet
Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp
Uses the *patterns* argument as real regular expressions rather than simple wildcard patterns.

-nocase
Makes matches case-insensitive.

-exact
Disables simple pattern matching. This is used when searching for objects that contain the * (asterisk) and ? (question mark) wildcard characters.

patterns
Matches power switch names against patterns. Patterns can include the * (asterisk) and ? (question mark) wildcard characters. For more details about using and escaping wildcards, refer to the **wildcards** man page. The *patterns* argument and **-of_objects** option are mutually exclusive. If you do not specify either, the tool uses "".

-hierarchical

Searches for power switches in the current scope and all of its descendant scopes. If you specify this option, you can specify only simple names in the *patterns* argument; the name cannot contain hierarchy delimiter characters. If you do not specify this option, the tool searches for power switches only in the current scope and you can use hierarchy delimiter characters in the *patterns* argument to specify hierarchical power switches relative to the current scope. This option is mutually exclusive with the **-of_objects** option.

-of_objects *objects*

Returns a collection of power switches associated with the specified objects. The objects can be power domains or supply nets. The *patterns* argument and **-of_objects** option are mutually exclusive.

DESCRIPTION

The **get_power_switches** command creates a collection of power switches in the current design, that match certain criteria. If no power switches match the criteria and the design is not linked, the tool automatically links the design. If there are still no supply nets that match the criteria, an empty string is returned.

Regular expression matching is the same as in the Tcl regexp command. When using **-regexp**, use care in the way you quote the patterns and filter expression. Use rigid quoting with curly braces around regular expressions. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can broaden the search by adding *".*"* to the beginning or end of the expressions as needed.

You can use the **get_power_switches** command at the command prompt, or you can nest it as an argument to another command, such as **report_supply_net**. In addition, you can assign the **get_power_switches** result to a variable.

When issued from the command prompt, **get_power_switches** behaves as though **query_objects** has been called to display the objects in the collection. By default, a maximum of 100 objects is displayed. You can change this maximum by using the **collection_result_display_limit** variable.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example gets all power switches under the current scope:

```
prompt> get_power_switches -hierarchical
{SW1 SUB_INST/SW2}
```

SEE ALSO

`create_power_switch(2)`
`remove_power_switch(2)`
`report_power_switch(2)`

get_route_guides

Creates a collection of route guides from the current design.

SYNTAX

```
collection get_route_guides  
[-within region | -touching region]  
[-filter expression]  
[-quiet]  
[patterns]
```

Data Types

<i>region</i>	list
<i>expression</i>	string
<i>patterns</i>	list

ARGUMENTS

-within *region*

Creates a collection containing all route guides within the specified region. The region boundary can be rectangle or polygon. The valid format of a rectangle is `{{llx lly} {urx ury}}` or `{llx lly urx ury}`, which specifies the lower-left and upper-right corners of the rectangle. The valid format of a polygon is: `{{x1 y1} {x2 y2} ... {xN yN} {x1 y1}}`, and each `{x y}` pair specifies one point of the input polygon. Note that the valid polygon is rectilinear, so the adjacent points have one same coordinate. The coordinate unit is specified in technology file (usually it is microns). The **-within** and **-touching** options are mutually exclusive; you can specify only one.

-touching *region*

Creates a collection containing all route guides touching the specified region. The region boundary can be rectangle or polygon. The valid format of a rectangle is `{{llx lly} {urx ury}}` or `{llx lly urx ury}`, which specifies the lower-left and upper-right corners of the rectangle. The valid format of a polygon is: `{{x1 y1} {x2 y2} ... {xN yN} {x1 y1}}`, and each `{x y}` pair specifies one point of the input polygon. Note that the valid polygon is rectilinear, so the adjacent points have one same coordinate. The coordinate unit is specified in technology file (usually it is microns). The **-within** and **-touching** options are mutually exclusive; you can specify only one.

-filter *expression*

Filters the collection with *expression*. For any route guides that match the patterns option, the expression is evaluated based on the attributes of the route guides. If the expression evaluates to *true*, the route guide is included in the result.

-quiet

Suppresses warning and error messages.

patterns

Specifies the route guides. The patterns can be a collection handle of route guides or other formats, similar to the following:

- * (asterisk) or rg* indicates all route guides
 - rg7789 indicates one route guide whose object_id is 7789.
- If this argument is not specified, the command uses * (asterisk) as the pattern.

DESCRIPTION

This command creates a collection of route guides that meet the selection criteria. It returns a collection of route guides if one or more route guides meet the selection criteria. If no route guides match the selection criteria, it returns an empty string. You can use the **get_route_guides** command as an argument to another command or assign its result to a variable. Refer to the example below for more information.

See the **collections** man page for information about working with collections.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following examples show some uses of this command to create route guide collections:

```
prompt> get_route_guides * -within {{2 2} {25 25}}
{"RG5389"}
```

```
prompt> get_route_guides * \
-touching {{10 5} {20 5} {20 15} {10 15} {10 5}}
{"RG5938"}
```

```
prompt>get_route_guides * -filter "area > 1000"
{"RG4683"}
```

SEE ALSO

collections(2)
create_route_guide(2)
create_placement_blockage(2)
get_attribute(2)
get_placement_blockages(2)
query_objects(2)
remove_placement_blockage(2)
remove_route_guide(2)

get_route_mode_options

Gets the mode value for running zroute.

SYNTAX

```
status get_route_mode_options  
[-zroute]
```

ARGUMENTS

-zroute
Returns the mode value for running zroute.

DESCRIPTION

This command gets the value which specifies whether zroute is selected as the default router.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command returns the value of the mode for running zroute.

```
prompt> get_route_mode_options -zroute
```

SEE ALSO

set_route_mode_options(2)

get_route_zrt_common_options

Gets the values of the specified common route options.

SYNTAX

```
status get_route_zrt_common_options  
[-name pattern]
```

Data Types

pattern string

ARGUMENTS

-name *pattern*
 Specifies the option to return the value for.
 By default, the command returns the value for all common route options.

DESCRIPTION

This command gets the value of the specified options.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command returns the values of all common route options.

```
prompt> get_route_zrt_common_options
```

SEE ALSO

```
set_route_zrt_common_options(2)  
report_route_zrt_common_options(2)
```


get_route_zrt_detail_options

Gets the values of the specified droute options.

SYNTAX

```
status get_route_zrt_detail_options  
[-name pattern]
```

Data Types

pattern string

ARGUMENTS

-name *pattern*
 Specifies the option to return the value for.
 By default, the command returns the value for all droute options.

DESCRIPTION

This command gets the value of the specified options.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command returns the values of all droute options.

```
prompt> get_route_zrt_detail_options
```

SEE ALSO

```
set_route_zrt_detail_options(2)  
report_route_zrt_detail_options(2)
```

get_route_zrt_global_options

Gets the values of the specified groute options.

SYNTAX

```
status get_route_zrt_global_options  
[-name pattern]
```

Data Types

pattern string

ARGUMENTS

-name *pattern*
 Specifies the option to return the value for.
 By default, the command returns the value for all groute options.

DESCRIPTION

This command gets the value of the specified options.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command returns the values of all groute options.

```
prompt> get_route_zrt_global_options
```

SEE ALSO

```
set_route_zrt_global_options(2)  
report_route_zrt_global_options(2)
```

get_route_zrt_track_options

Gets the values of the specified track assignment options.

SYNTAX

```
status get_route_zrt_track_options  
[-name pattern]
```

Data Types

pattern string

ARGUMENTS

-name *pattern*
 Specifies the option to return the value for.
 By default, the command returns the value for all track assignment options.

DESCRIPTION

This command gets the value of the specified options.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command returns the values of all track assignment options.

```
prompt> get_route_zrt_track_options
```

SEE ALSO

```
set_route_zrt_track_options(2)  
report_route_zrt_track_options(2)
```

get_routing_blockages

Creates a collection of routing blockages from the current design.

SYNTAX

```
collection get_routing_blockages
[-within region
 | -intersect region
 | -touching region
 | -at point]
[-filter expression]
[-quiet]
[-type via | metal]
[patterns]
```

Data Types

<i>region</i>	list of points
<i>point</i>	point
<i>expression</i>	string
<i>patterns</i>	string

ARGUMENTS

-within *region*

Creates a collection containing all routing blockages within the specified region. The region boundary can be rectangle or polygon. The valid format of a rectangle is `{{llx lly} {urx ury}}` or `{llx lly urx ury}`, which specifies the lower-left and upper-right corners of the rectangle. The valid format of a polygon is: `{{x1 y1} {x2 y2} ... {xN yN} {x1 y1}}`, and each `{x y}` pair specifies one point of the input polygon. Note that the valid polygon is rectilinear, so the adjacent points have one same coordinate. The coordinate unit is specified in technology file (usually it is microns). The **-within**, **-intersect**, **-touching**, and **-at** options are mutually exclusive; you can specify only one.

-intersect *region*

Creates a collection containing all routing blockages intersect the specified region. The region boundary can be rectangle or polygon. The valid format of a rectangle is `{{llx lly} {urx ury}}` or `{llx lly urx ury}`, which specifies the lower-left and upper-right corners of the rectangle. The valid format of a polygon is: `{{x1 y1} {x2 y2} ... {xN yN} {x1 y1}}`, and each `{x y}` pair specifies one point of the input polygon. Note that the valid polygon is rectilinear, so the adjacent points have one same coordinate. The coordinate unit is specified in technology file (usually it is microns). The **-within**, **-intersect**, **-touching**, and **-at** options are mutually exclusive; you can specify only one.

-touching *region*

Creates a collection containing all routing blockages touching the specified region. The region boundary can be rectangle or polygon. The valid format of a rectangle is `{{llx lly} {urx ury}}` or `{llx lly urx ury}`, which specifies

the lower-left and upper-right corners of the rectangle.
The valid format of a polygon is: `{{x1 y1} {x2 y2} ... {xN yN} {x1 y1}}`, and each `{x y}` pair specifies one point of the input polygon. Note that the valid polygon is rectilinear, so the adjacent points have one same coordinate. The coordinate unit is specified in technology file (usually it is microns). The **-within**, **-intersect**, **-touching**, and **-at** options are mutually exclusive; you can specify only one.

-at point

Creates a collection containing all routing blockages that overlap the point specified by the coordinates `{x y}`. The coordinate unit is specified in technology file (usually it is microns). The **-within**, **-intersect**, **-touching**, and **-at** options are mutually exclusive; you can specify only one.

-filter expression

Filters the collection with *expression*, which must be a Boolean expression based on the following routing blockage attributes:

Attribute names	Description
bbox	The bounding box of the routing blockage
points	Coordinates of the rectangle or polygon
name	Name of the routing blockage
layer on	The name of the layer that the routing blockage is on
layer_number is on	The number of the layer that the routing blockage is on
area	The area of the rectangle of the routing blockage

The tool evaluate the expression for any routing blockages that matches the *patterns* argument. If the expression evaluates to true, the routing blockage is included in the result.

-quiet

Suppresses warning and error messages.

-type via | metal

This option to identify what type routing blockage do you want to query. If you specify **-type via**, the command returns the routing blockages located on the via1Blockage-via14Blockage and polyContBlockage layers. If you specify **-type metal**, the command returns the routing blockages located on the metal1Blockage-metal15Blockage and polyBlockage layers. If you do not specify this option, the command returns all routing blockages in the current design that meet the selection criteria.

patterns

Matches the routing blockage names in the current design against the specified patterns. You can specify the patterns by using the following formats:

- A collection of routing blockages
 - An asterisk (*), which indicates all routing blockages
 - Routing blockage names, which are in the format `RB_object_id`
- If you do not specify this argument, the command uses asterisk (*).

DESCRIPTION

This command returns a collection of routing blockages of the current design that meet the selection criteria. You can use this command to query the routing blockages inside the Milkyway database by area, by point, or by region. You can also use this command to query routing blockages by filtering their attributes.

EXAMPLES

The following example returns the routing blockages within the rectangle with the lower-left corner at {2 2} and the upper-right corner at {25 25}.

```
prompt> get_routing_blockages * -within {{2 2} {25 25}}
{"RB_5389"}
```

The following example returns the routing blockages intersect the polygon.

```
prompt> get_routing_blockages \
-intersect {{10 5} {20 5} {20 15} {10 15} {10 5}}
{"RB_4306"}
```

The following example returns the routing blockages whose area is larger than 1000.

```
prompt> get_routing_blockages * -filter ,Área > 1000"
{"RB_4683"}
```

The following example returns all routing blockages.

```
prompt> get_routing_blockages "RB_*"
{"RB_4683 RB_5389"}
```

The following example shows how to use **get_attribute** to query the value of the routing blockage attributes.

```
prompt> get_attribute [get_routing_blockages RB_4683] layer
via3Blockage
prompt> get_attribute [get_routing_blockages RB_4683] bbox
{0.000 0.000}{100.000 100.000}
prompt> get_attribute [get_routing_blockages RB_4683] area
10000.000000
```

To get a complete report of the attribute values of all routing blockages, use the **report_attribute** command, as shown in the following example.

```
prompt> report_attribute -application [get_routing_blockages RB_4683]
Design  Object    Type   Attribute Name  Value
-----
CORE    RB_4683    float   area            10000.000000
CORE    RB_4683    string  bbox            {0.000 0.000} {100.000 100.000}
CORE    RB_4683    int     cell_id         3
CORE    RB_4683    string  full_name       CORE/RB_4683
CORE    RB_4683    string  layer           via3Blockage
CORE    RB_4683    int     layer_number    217
CORE    RB_4683    string  name            RB_4683
```

CORE	RB_4683	string	object_class	route_guide
CORE	RB_4683	int	object_id	1234
CORE	RB_4683	string	object_type	RECTANGLE

SEE ALSO

`collections(2)`
`create_routing_blockage(2)`
`get_attribute(2)`
`remove_routing_blockage(2)`
`report_attribute(2)`

get_rp_group_keepouts

Creates a collection of the specified relative placement group keepouts.

SYNTAX

```
collection get_rp_group_keepouts
[-quiet]
[patterns]
[-exact]
[-regexp]
[-filter expression]
[-of_objects objects]
```

Data Types

patterns list

ARGUMENTS

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

patterns

Matches relative placement group keepout names against the specified patterns. The format of each pattern string is `design_name::rp_group_name::keepout_name` If you do not specify this argument, the command returns a collection contain all relative placement group keepouts in the current design.

DESCRIPTION

The `get_rp_group_keepouts` command creates a collection of relative placement group keepouts that match the specified patterns. This command is supported only for designs that do not contain multiply-instantiated designs.

If no objects match the criteria, the empty string is returned. If you omit the *patterns* argument, the command returns all the relative placement group keepouts in the current design.

By default, a maximum of 100 objects is displayed when using the command at the command prompt. You can change this maximum by using the `collection_result_display_limit` variable.

EXAMPLES

The following example queries the keepouts in relative placement. Although the output looks like a list, it is not. The output is just a display.

```
get_rp_group_keepouts
938
```



```
prompt> get_rp_group_keepouts ripple::grp_ripple::keepout1
{ripple::grp_ripple::keepout1}
prompt> get_rp_group_keepouts
{ripple::grp_ripple::keepout1 ripple::grp_ripple::keepout2 ripple::grp_ripple::keepout3}
```

SEE ALSO

```
create_rp_group(2)
add_to_rp_group(2)
remove_from_rp_group(2)
write_rp_groups(2)
```

get_rp_groups

Creates a collection of relative placement groups.

SYNTAX

```
collection get_rp_groups
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-ignored]
[-top]
[-filter expression]
[patterns | -of_objects objects]
```

Data Types

<i>expression</i>	string
<i>patterns</i>	list
<i>objects</i>	list

ARGUMENTS

- quiet
Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.
- regexp
Uses the *patterns* argument as regular expressions rather than simple wildcard patterns.
- nocase
Makes the matches case-insensitive.
- exact
Disables simple pattern matching. For use when searching for objects that contain the * (asterisk) wildcard character.
- ignored
Specifies that only ignored relative placement groups should be collected.
- top
Specifies that only top level relative placement groups should be collected.
- filter *expression*
Filters the collection with the value of *expression*. For any relative placement groups that match the specified criteria, the expression is evaluated based on the relative placement group's attributes. If the expression evaluates to true, the relative placement group is included in the result.

patterns

Matches relative placement group names against patterns (or regular expressions if you use the **-regexp** option). The format of each pattern string can be in either *design_name::rp_group_name* or *rp_group_name*. Both *design_name* and *rp_group_name* can include the wildcard characters * (asterisk) and ? (question mark). For details about using and escaping wildcard characters see the **wildcards** man page. The *patterns* and **-of_objects** options are mutually exclusive; you can choose only one.

-of_objects *objects*

Creates a collection of relative placement groups containing the specified objects. In this case, each object is either a cell, an instantiated or included relative placement group or a relative placement group keepout. The *patterns* and **-of_objects** options are mutually exclusive; you can use only one. If you do not specify *patterns* or **-of_objects**, the command returns a collection containing all relative placement groups in the current design.

DESCRIPTION

The **get_rp_groups** command creates a collection of relative placement groups, that match certain criteria. This command is supported only for designs that do not contain multiply-instantiated designs. If no objects match the criteria, the empty string is returned.

Regular expression matching is the same as in the Tcl **regexp** command. When using **-regexp**, take care in the way you quote the *patterns*; use rigid quoting with curly braces around regular expressions. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding *".*"* to the beginning or end of the expressions as needed.

You can use the **get_rp_groups** command at the command prompt, or you can nest it as an argument to another command, such as **set_rp_group_options**. In addition, you can assign the **get_rp_groups** result to a variable.

By default, a maximum of 100 objects is displayed when using the command at the command prompt. You can change this maximum by using the **collection_result_display_limit** variable.

The implicit query property of **get_rp_groups** provides a fast, simple way to display relative placement groups in a collection.

Relative placement usage is available with Design Compiler Ultra.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example queries the relative placement groups starting with **g** and in a design starting with **r**. Although the output looks like a list, the output is just a display.

```
prompt> get_rp_groups r*::g*
{ripple::grp_ripple}
```

The following example sets the utilization attribute of relative placement groups using **get_rp_groups**:

```
prompt> set_rp_group_options [get_rp_groups r*::g*]\
-utilization 0.95
{ripple::grp_ripple}
```

The following example queries the relative placement groups that have 8 rows:

```
prompt> get_rp_groups -filter rows==8
{mul::grp_mul}
```

The following example queries the relative placement groups that include the cell U17 or the rp_group example3::block2:

```
prompt> get_rp_groups\
-of_objects {U17 example3::block2}
{example3::gp_2_in_example3 example3::top_group}
```

SEE ALSO

```
add_to_rp_group(2)
all_rp_groups(2)
create_rp_group(2)
remove_from_rp_group(2)
set_rp_group_options(2)
write_rp_groups(2)
```

get_scan_cells_of_chain

Returns a collection containing all scan cells of the specified scan chain.

SYNTAX

```
collection get_scan_cells_of_chain  
-chain chain_name
```

Data Types

chain_name string

ARGUMENTS

```
-chain chain_name  
      Specifies the name of the scan chain.
```

DESCRIPTION

This command returns a Tcl collection containing all scan cells for the specified scan chain.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

This example gets all scan cells of the *chain1* scan chain.

```
prompt> get_scan_cells_of_chain -chain "chain1"
```

SEE ALSO

`get_scan_chains(2)`

get_scan_chains

Returns the number of scan chains in the current design.

SYNTAX

status **get_scan_chains**

ARGUMENTS

The **get_scan_chains** command has no arguments.

DESCRIPTION

This command returns the number of scan chains in the current design.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

This example shows how to get the number of scan chains in the design.

```
prompt> get_scan_chains
```

SEE ALSO

`get_scan_cells_of_chain(2)`

get_selection

Returns a collection containing the current selection in the GUI.

SYNTAX

```
collection get_selection
[-slct_targets target_selection_bus
[-slct_targets_operation operation]]
-create_slct_buses
[-name selection_bus]
[-type object_type]
[-design design]
[-more_than more]
[-fewer_than fewer]
[-count]
[-num num]
[-type_list]
```

Data Types

<i>target_selection_bus</i>	string
<i>operation</i>	string
<i>selection_bus</i>	string
<i>object_type</i>	string
<i>design</i>	string
<i>more</i>	integer
<i>fewer</i>	integer
<i>num</i>	integer

ARGUMENTS

-slct_targets *target_selection_bus*
Specifies the name of a selection bus in which to store the result. When you use this option, the *target_selection_bus* value is also returned as result of the command. By default, the result is returned in a collection.

-slct_targets_operation *operation*
Specifies which operation should be used when storing the result in the selection bus specified by the *target_selection_bus* argument. The valid values for the *operation* argument are **clear**, **add**, and **remove**. You can use the **-slct_targets_operation** option only if you also use the **-slct_targets** option.

-create_slct_buses
Specifies to create a new selection bus in which to store the result. Using this option is equivalent to using the **create_selection_bus** command to create a selection bus and then providing the created selection bus to the **-slct_targets** option.

-name *selection_bus*
Specifies the selection bus from which the selection is taken.

-type *object_type*
 Specifies the type of selected object with which to filter the selection. The valid values for *object_type* and their descriptions are as follows:

- design** -design object
- port** -port object
- net** -net object
- cell** -cell object
- pin** -pin object
- path** -timing path object

-design *design*
 Returns only objects in the filter specified by the value of *design*.

-more_than *more*
 Returns true if the selection bus contains more than the number of objects specified by *more*.

-fewer_than *fewer*
 Returns true if the selection bus contains more than the number of objects specified by *fewer*.

-count
 Returns the number of elements contained in the selection bus.

-num *num*
 Returns only the number of objects specified by *num* or fewer.

DESCRIPTION

This command returns the current selection in the tool as a collection. It returns a collection handle (identifier) if any objects are selected. If no objects are selected, the command returns an empty string. If you do not use the **-type** option, the command returns a heterogeneous collection of all objects currently selected. If you use **-type**, the collection is filtered to a homogeneous one containing only objects of the type you specify.

You can use the **get_selection** command at the command prompt, or you can nest it as an argument to another command (for example, **query_objects**), or assign the **get_selection** result to a variable.

When issued from the command prompt, **get_selection** behaves as if you had called **query_objects** to display the objects in the collection. By default, the command displays a maximum of 100 objects. You can change this maximum by using the **collection_result_display_limit** variable.

The "implicit query" property of **get_selection** provides a fast, simple way to display cells in a collection. However, if you want the flexibility provided by the options of the **query_objects** command (for example, if you want to display the object class), use **get_selection** as an argument to **query_objects**.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example returns the collection of the selected cell objects.

```
prompt> get_selection -type cell  
{ "I_PRGRM_CNT_TOP", "I_DATA_PATH", "I_REG_FILE", "I_STACK_TOP" }
```

The following example assigns the collection to a variable named **collect_a** that is passed to the **query_objects** command.

```
prompt> set collect_a [get_selection -type cell]  
{ "I_PRGRM_CNT_TOP", "I_DATA_PATH", "I_REG_FILE", "I_STACK_TOP" }  
prompt> query_objects $collect_a  
{ "I_PRGRM_CNT_TOP", "I_DATA_PATH", "I_REG_FILE", "I_STACK_TOP" }
```

SEE ALSO

```
change_selection(2)  
collections(2)  
filter_collection(2)  
query_objects(2)
```

get_si_xtalk_bumps

Reports individual aggressor bumps for a given net.

SYNTAX

```
status get_si_xtalk_bumps  
net
```

ARGUMENTS

```
net  
    Reports the xtalk bumps for the given net.
```

DESCRIPTION

This command reports xtalk bumps (4 values) for each individual aggressor. The information is given in the following format, which is shown on separate lines for display purposes only. When you use it, you can use 1 continuous line.

```
{aggr1 max_rise_bump max_fall_bump min_rise_bump min_fall_bump}  
{aggr2 max_rise_bump max_fall_bump min_rise_bump min_fall_bump} ...
```

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example shows detailed aggressor bumps for a net that starts with DSPCORE31/exetop0/e_dptop0/abs_sftw_0_.

```
prompt> get_si_xtalk_bumps \  
DSPCORE31/exetop0/e_dptop0/abs_sftw_0_  
{DSPCORE31/exetop0/e_dptop0/abs_sftw_4_ 0.0683 0.0521 0.0705 0.0672}  
{DSPCORE31/exetop0/e_dptop0/e_lu0/N1338 0.0438 0.0300 0.0448 0.0438}
```

SEE ALSO

```
report_noise(2)
```

get_site_rows

Returns a collection of site rows from the current design.

SYNTAX

```
collection get_site_rows
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-filter expression]
[-within region
  | -intersect region
  | -touching region]
patterns
  | -of_objects objects
  | -object_id id
```

Data Types

<i>expression</i>	string
<i>region</i>	list of points
<i>patterns</i>	list
<i>objects</i>	list
<i>id</i>	int

ARGUMENTS

-quiet
Suppresses warning and error messages, if no objects match. Does not suppress syntax error messages.

-regexp
Views the *patterns* argument as a real regular expression rather than a simple wildcard pattern. This option also modifies the behavior of the =~ and !~ filter operators to behave as regular expressions rather than simple wildcard patterns.

-nocase
Makes matches case-insensitive.

-exact
Disables simple pattern matching. For use when searching for objects that contain the * wildcard character.

-filter *expression*
Filters the collection with *expression*. For any site row that match the *patterns* option, the expression is evaluated based on the site row's attributes. If the expression evaluates to *true*, the site row is included in the result.

-within *region*

Creates a collection containing all site rows within the specified region. The region boundary can be rectangle or polygon. The valid format of a rectangle is `{{llx lly} {urx ury}}` or `{llx lly urx ury}`, which specifies the lower-left and upper-right corners of the rectangle. The valid format of a polygon is: `{{x1 y1} {x2 y2} ... {xN yN} {x1 y1}}` and each `{x y}` pair specifies one point of the input polygon. Note that a valid polygon is rectilinear, so the adjacent points have one identical coordinate. The coordinate unit is specified in the technology file (usually it is microns). The **-within**, **-intersect** and **-touching** options are mutually exclusive; you can specify only one.

-intersect *region*

Creates a collection containing all site rows that intersect the specified region. The region boundary can be rectangle or polygon. The valid format of a rectangle is `{{llx lly} {urx ury}}` or `{llx lly urx ury}`, which specifies the lower-left and upper-right corners of the rectangle. The valid format of a polygon is: `{{x1 y1} {x2 y2} ... {xN yN} {x1 y1}}` and each `{x y}` pair specifies one point of the input polygon. Note that a valid polygon is rectilinear, so the adjacent points have one identical coordinate. The coordinate unit is specified in the technology file (usually it is microns). The **-within**, **-intersect** and **-touching** options are mutually exclusive; you can specify only one.

-touching *region*

Creates a collection containing all site rows that touch the specified region. The region boundary can be rectangle or polygon. The valid format of a rectangle is `{{llx lly} {urx ury}}` or `{llx lly urx ury}`, which specifies the lower-left and upper-right corners of the rectangle. The valid format of a polygon is: `{{x1 y1} {x2 y2} ... {xN yN} {x1 y1}}` and each `{x y}` pair specifies one point of the input polygon. Note that a valid polygon is rectilinear, so the adjacent points have one identical coordinate. The coordinate unit is specified in the technology file (usually it is microns). The **-within**, **-intersect** and **-touching** options are mutually exclusive; you can specify only one.

patterns

Matches site row names against the *patterns* argument. The *patterns* argument can include the wildcard character `"*"`. The **-of_objects**, *patterns*, and **-object_id** arguments are mutually exclusive; you must specify only one.

-of_objects *objects*

Creates a collection of site rows connected to the specified objects. In this case, each object must be a core area collection. The **-of_objects**, *patterns*, and **-object_id** arguments are mutually exclusive; you must specify only one.

-object_id *id*

Creates a collection of a site row with the specified Milkyway object ID. The **-of_objects**, *patterns*, and **-object_id** arguments are mutually exclusive; you must specify only one.

DESCRIPTION

The **get_site_rows** command creates a collection of site rows from the current design that match certain criteria. The command returns a collection of site rows if any site row matches the criteria. If no objects match the criteria, the empty string is returned.

You can use the **get_site_rows** command at the command prompt, or you can nest it as an argument to another command, such as **report_attributes**. In addition, you can assign the **get_site_rows** result to a variable.

When issued from the command prompt, the **get_site_rows** command behaves as though the **query_objects** command has been called to report the objects in the collection. By default, a maximum of 100 objects are displayed. You can change this maximum using the **collection_result_display_limit** variable.

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following example returns a collection that contains all the site rows in the current design.

```
prompt> get_site_rows
{SITE_ROW#12345 SITE_ROW#12346 SITE_ROW#12347 my_row1 my_row2}
```

The following example returns a collection that contains the site rows whose name is prefixed with my_row.

```
prompt> get_site_rows my_row*
{my_row1 my_row2}
```

The following example returns a collection that contains all the site rows in the specified core area.

```
prompt> get_site_rows -of_objects [get_core_area]
{SITE_ROW#12345 SITE_ROW#12346 SITE_ROW#12347 my_row1 my_row2}
```

The following example returns a collection that contains the site rows within the specified rectangle.

```
prompt> get_site_rows -within {{2 2} {250 150}}
{test_site_row1}
```

The following example returns a collection that contains the site rows that intersect the specified polygon.

```
prompt> get_site_rows \
-intersect {{5 5} {200 5} {200 100} {5 100} {5 5}}
{test_site_row1 test_site_row2}
```

SEE ALSO

`collections(2)`
`query_objects(2)`
`collection_result_display_limit(3)`
`create_site_row(2)`
`remove_site_row(2)`
`ignore_site_row(2)`
`set_row_type(2)`
`remove_row_type(2)`

get_supply_nets

Creates a collection of supply nets that meet the specified criteria. This command is supported only in UPF mode.

SYNTAX

```
collection get_supply_nets
[-filter expression]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[[patterns] [-hierarchical]
 | -of_objects objects]]
```

Data Types

<i>expression</i>	string
<i>patterns</i>	list
<i>objects</i>	list

ARGUMENTS

-filter *expression*
Filters the collection with *expression*. For any supply nets that match *patterns*, the expression is evaluated based on the supply net's attributes. If the expression evaluates to true, the supply net is included in the result.

-quiet
Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp
Uses the *patterns* argument as real regular expressions rather than simple wildcard patterns.

-nocase
Makes matches case-insensitive.

-exact
Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. This option is available only in UPF mode.

patterns
Matches supply net names against patterns. Patterns can include the wildcard characters * (asterisk) and ? (question mark). For more details about using and escaping wildcards, refer to the **wildcards** man page. The *patterns* argument and **-of_objects** option are mutually exclusive. If you do not specify either, the tool uses "".

-hierarchical

Searches for supply nets in the current scope and all of its descendant scopes. If you specify this option, you can specify only simple names in the *patterns* argument; the name cannot contain hierarchy delimiter characters. If you do not specify this option, the tool searches for supply nets only in the current scope and you can use hierarchy delimiter characters in the *patterns* argument to specify hierarchical supply nets relative to the current scope.

You cannot specify this option when you use the **-of_objects** option. This option is available in UPF mode only.

-of_objects *objects*

Returns a collection of supply nets associated with the specified objects. The objects can be power domains, supply ports, or power switches. The *patterns* argument and **-of_objects** option are mutually exclusive.

DESCRIPTION

The **get_supply_nets** command creates a collection of supply nets in the current design, that match certain criteria. If no supply nets match the criteria and the design is not linked, the tool automatically links the design. If there are still no supply nets that match the criteria, an empty string is returned.

Regular expression matching is the same as in the Tcl **regexp** command. When using **-regexp**, take care in the way you quote the patterns and filter expression. Use rigid quoting with curly braces around regular expressions. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding *".*"* to the beginning or end of the expressions as needed.

You can use the **get_supply_nets** command at the command prompt, or you can nest it as an argument to another command, such as **report_supply_net**. In addition, you can assign the **get_supply_nets** result to a variable.

When issued from the command prompt, **get_supply_nets** behaves as though **query_objects** has been called to display the objects in the collection. By default, a maximum of 100 objects is displayed. You can change this maximum by using the **collection_result_display_limit** variable.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example gets all supply nets under the current scope.

```
prompt> get_supply_nets -hierarchical
```



```
{VDD SUB_INST/VDD}
```

SEE ALSO

```
create_supply_net(2)  
remove_supply_net(2)  
report_supply_net(2)
```

get_supply_ports

Creates a collection of supply ports that meet the specified criteria. This command is supported only in UPF mode.

SYNTAX

```
collection get_supply_ports
[-filter expression]
[-quiet]
[-regexp [-nocase]]
[-exact]
[[patterns]
[-hierarchical] | -of_objects objects]
```

Data Types

<i>expression</i>	string
<i>patterns</i>	list
<i>objects</i>	list

ARGUMENTS

- filter *expression***
Filters the collection with the value of the *expression* argument. For supply ports that match the specified patterns (or objects), the expression is evaluated based on the attributes of the supply port. If the expression evaluates to true, the supply port is included in the result. This option works the same way as the **filter_collection** command works. The **-regexp** and **-nocase** options are applied in the same way as in the **filter_collection** command. See the **filter_collection** man page for more information on how to use a **-filter** option in general.
- quiet**
Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.
- regexp**
Uses the value of the *patterns* argument as a real regular expression rather than as a simple wildcard pattern.
- nocase**
Makes matches case-insensitive. You can use the **-nocase** option only when you also use the **-regexp** option.
- exact**
Disables simple pattern matching. Use this option when searching for objects that contain the "*" (asterisk) wildcard as an actual character.
- patterns**
Matches supply port names against patterns. Patterns can include the wildcard characters * (asterisk) and ? (question mark). For more details about using and escaping wildcards, refer to the **wildcards** man page.

The *patterns* argument and **-of_objects** option are mutually exclusive. If you do not specify either, the tool uses **"*"**.

-hierarchical

Searches for supply ports in the current scope and all of its descendant scopes. When you use the **-hierarchical** option, you can specify only simple names in the *patterns* option; the name cannot contain hierarchy delimiter characters. By default, the tool searches for supply ports only in the current scope, and you can use hierarchy delimiter characters in the *patterns* option to specify hierarchical supply ports relative to the current scope. You cannot specify the **-hierarchical** option when you use the **-of_objects** option.

-of_objects *objects*

Returns a collection of supply ports associated with the specified objects. The objects can be power domains or supply nets. The *patterns* argument and **-of_objects** option are mutually exclusive.

DESCRIPTION

This command creates a collection of supply ports in the current design that match certain criteria. If no supply ports match the criteria and the design is not linked, the tool automatically links the design. If there are still no supply ports that match the criteria, the command returns an empty string.

Regular expression matching in the **get_physical_lib_cells** command is the same as in the Tcl **regexp** command. When using **-regexp**, take be careful how you quote the *patterns* and filter expression. It is recommended that you use rigid quotation marks with curly braces around regular expressions.

Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search simply by adding **.*** to the beginning or end of the expressions, as needed.

You can use the **get_supply_ports** command at the command prompt, nest it as an argument to another command (for example, **report_supply_net**), or assign the **get_supply_ports** result to a variable.

When issued from the command prompt, **get_supply_ports** behaves as if you had called **query_objects** to display the objects in the collection. By default, the command displays a maximum of 100 objects. You can change this maximum by using the **collection_result_display_limit** variable.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example gets all supply ports under the current scope.

```
prompt> get_supply_ports -hierarchical  
{VN SUB_INST/VN12}
```

SEE ALSO

```
create_supply_port(2)  
remove_supply_port(2)  
report_supply_port(2)
```

get_terminals

Creates a collection by selecting terminals from the current design.

SYNTAX

```
collection get_terminals
[patterns
 | -of_objects port_list
 | -object_id object_id]
[-within region
 | -intersect region
 | -touching region
 | -at point]
[-filter expression]
[-regexp [-nocase] | -exact]
[-quiet]
```

Data Types

<i>patterns</i>	list
<i>port_list</i>	collection
<i>object_id</i>	integer
<i>region</i>	list of points
<i>point</i>	point
<i>expression</i>	string

ARGUMENTS

patterns

Matches the terminal names against the specified patterns. Patterns can include the wildcard characters * (asterisk) and ? (question mark) or regular expressions, depending on the use of the **-regexp** option. For more details about using and escaping wildcards, see the **wildcards** man page. Patterns can also include collections of terminals.

The *patterns*, **-of_objects**, and **-object_id** arguments are mutually exclusive; you can use only one. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

-of_objects *port_list*

Creates a collection of terminals that belong to the specified ports. Each element of the *port_list* can be a port name, port name pattern, or a port collection.

The *patterns*, **-of_objects**, and **-object_id** arguments are mutually exclusive; you can use only one. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

-object_id *object_id*

Specifies the object ID of one terminal.

The *patterns*, **-of_objects**, and **-object_id** arguments are mutually exclusive; you can use only one. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

-within *region*

Creates a collection containing all terminals within the specified region. The region boundary can be rectangle or polygon. The valid format of a rectangle is `{{llx lly} {urx ury}}` or `{llx lly urx ury}`, which specifies the lower-left and upper-right corners of the rectangle. The valid format of a polygon is: `{{x1 y1} {x2 y2} ... {xN yN} {x1 y1}}`, and each `{x y}` pair specifies one point of the input polygon. Besides, a list of one polygon is also supported as input for this option, with the format: `{{{x1 y1} {x2 y2} ... {xN yN} {x1 y1}}}`. Pay attention that the valid polygon is rectilinear, so the adjacent points have one same coordinate. The **-within**, **-intersect**, **-touching**, and **-at** options are mutually exclusive; you can specify only one.

-intersect *region*

Creates a collection containing all terminals that intersect the specified region. The region boundary can be rectangle or polygon. The valid format of a rectangle is `{{llx lly} {urx ury}}` or `{llx lly urx ury}`, which specifies the lower-left and upper-right corners of the rectangle. The valid format of a polygon is: `{{x1 y1} {x2 y2} ... {xN yN} {x1 y1}}`, and each `{x y}` pair specifies one point of the input polygon. Besides, a list of one polygon is also supported as input for this option, with the format: `{{{x1 y1} {x2 y2} ... {xN yN} {x1 y1}}}`. Pay attention that the valid polygon is rectilinear, so the adjacent points have one same coordinate. The **-within**, **-intersect**, **-touching**, and **-at** options are mutually exclusive; you can specify only one.

-touching *region*

Creates a collection containing all terminals that touch the specified region. The region boundary can be rectangle or polygon. The valid format of a rectangle is `{{llx lly} {urx ury}}` or `{llx lly urx ury}`, which specifies the lower-left and upper-right corners of the rectangle. The valid format of a polygon is: `{{x1 y1} {x2 y2} ... {xN yN} {x1 y1}}`, and each `{x y}` pair specifies one point of the input polygon. Besides, a list of one polygon is also supported as input for this option, with the format: `{{{x1 y1} {x2 y2} ... {xN yN} {x1 y1}}}`. Pay attention that the valid polygon is rectilinear, so the adjacent points have one same coordinate. The **-within**, **-intersect**, **-touching**, and **-at** options are mutually exclusive; you can specify only one.

-at *point*

Creates a collection containing all terminals that overlap the point specified by the coordinates `{x y}`. The **-within**, **-intersect**, **-touching**, and **-at** options are mutually exclusive; you can specify only one.

-filter *expression*

Filters the collection with the value specified by *expression*. For any terminals that match the *patterns* argument, the expression is evaluated based on the terminal's attributes. If the expression evaluates to true, the terminal is included in the result.

-regexp

Uses the value of the *patterns* argument as a real regular expression rather than as a simple wildcard pattern. This option also modifies the behavior of the `~=` and `!~` filter operators to compare with real regular expressions,

rather than with simple wildcard patterns.
The **-regex** and **-exact** options are mutually exclusive; you can use only one.

-nocase

Makes matches case-insensitive when you use it with the **-regex** option. You can use the **-nocase** option only when you also use the **-regex** option.

-exact

Disables simple pattern matching. You can use this option when searching for objects that contain the * (asterisk) and ? (question mark) wildcard characters.

The **-regex** and **-exact** options are mutually exclusive; you can use only one.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

DESCRIPTION

This command creates a collection of terminals by selecting terminals from the current design that meet the selection criteria.

It returns a collection if one or more terminals meet the selection criteria. If no terminals match the selection criteria, it returns an empty string.

When using **-within**, **-touching**, or **-intersect**, note that the behavior will be different for the different format of the specified region. A terminal object has its boundary and bbox, sometimes they are different. When the specified region is in rectangle format, we will search objects by their bbox; when the specified region is in polygon format, we will search objects by their boundary. And searching by bbox is faster than searching by boundary.

You can use the **get_terminals** command at the command prompt or nest it as an argument to another command (such as **query_objects**). You can also assign the **get_terminals** result to a variable.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates a terminal collection.

```
prompt> get_terminals [list reset clock*]  
{reset clock}
```

```
prompt> get_terminals -object_id 4104
```

```
{reset}
```

The following example returns the terminals within the rectangle with the lower-left corner at {2 2} and the upper-right corner at {25 25}.

```
prompt> get_terminals -within {{2 2} {25 25}}  
{reset}
```

The following example gets terminals that intersect the specified polygon.

```
prompt> get_terminals \  
-intersect {{30 20} {50 20} {50 30} {40 30} {40 40} {30 40} {30 20}}  
{CCEN}
```

SEE ALSO

```
create_terminal(2)  
get_attribute(2)  
query_objects(2)  
remove_terminal(2)
```


get_text

Creates a collection of text objects from the current design.

SYNTAX

```
collection get_text
[-within region
 | -intersect region
 | -touching region
 | -at point]
[-filter expression]
[-quiet]
[patterns]
```

Data Types

<i>region</i>	list
<i>point</i>	string
<i>expression</i>	string
<i>patterns</i>	list

ARGUMENTS

-within *region*

Creates a collection containing all text objects within the specified region. The region boundary can be rectangle or polygon. The valid format of a rectangle is `{{llx lly} {urx ury}}` or `{llx lly urx ury}`, which specifies the lower-left and upper-right corners of the rectangle. The valid format of a polygon is: `{{x1 y1} {x2 y2} ... {xN yN} {x1 y1}}`, and each `{x y}` pair specifies one point of the input polygon. Note that the valid polygon is rectilinear, so the adjacent points have one same coordinate. The coordinate unit is specified in technology file (usually it is microns). The **-within**, **-intersect**, **-touching**, and **-at** options are mutually exclusive; you can specify only one.

-intersect *region*

Creates a collection containing all text objects that intersect the specified region. The region boundary can be rectangle or polygon. The valid format of a rectangle is `{{llx lly} {urx ury}}` or `{llx lly urx ury}`, which specifies the lower-left and upper-right corners of the rectangle. The valid format of a polygon is: `{{x1 y1} {x2 y2} ... {xN yN} {x1 y1}}`, and each `{x y}` pair specifies one point of the input polygon. Note that the valid polygon is rectilinear, so the adjacent points have one same coordinate. The coordinate unit is specified in technology file (usually it is microns). The **-within**, **-intersect**, **-touching**, and **-at** options are mutually exclusive; you can specify only one.

-touching *region*

Creates a collection containing all text objects that touch the specified region. The region boundary can be rectangle or polygon. The valid format of a rectangle is `{{llx lly} {urx ury}}` or `{llx lly urx ury}`, which specifies the lower-left and upper-right corners of the rectangle.

The valid format of a polygon is: `{{x1 y1} {x2 y2} ... {xN yN} {x1 y1}}`, and each `{x y}` pair specifies one point of the input polygon. Note that the valid polygon is rectilinear, so the adjacent points have one same coordinate. The coordinate unit is specified in technology file (usually it is microns). The **-within**, **-intersect**, **-touching**, and **-at** options are mutually exclusive; you can specify only one.

-at *point*

Creates a collection containing all text objects that overlap the point specified by the `{x y}` coordinates. The coordinate unit is specified in technology file (usually it is microns). The **-within**, **-intersect**, **-touching**, and **-at** options are mutually exclusive; you can specify only one.

-filter *expression*

Filters the collection with *expression*. For any text objects that match the *patterns* argument, the expression is evaluated based on the attributes of the text objects. If the expression evaluates to *true*, the text object is included in the result.

-quiet

Suppresses warning and error messages.

patterns

Specifies the patterns of the text objects. The *patterns* argument can include collections of text objects and patterns with the following formats:

- `*` or `TEXT#*` removes all text

- `TEXT#6400` removes one text whose `object_id` is 6400.

If you do not specify this argument, the command uses asterisk (`*`) as the pattern.

DESCRIPTION

This command creates a collection of text objects that meet the selection criteria. It returns a collection of text objects if one or more text objects meet the selection criteria. If no text object matches the selection criteria, it returns an empty string. You can use the **get_text** command as an argument to another command or assign its result to a variable. Refer to the example below for more information.

See the **collections** man page for information about working with collections.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following examples show some uses of this command to create text collections.

The following example returns the text objects found within the specified rectangle:

```
prompt> get_text * -within {{10 9} {45 15}}  
{"TEXT#6401"}
```

The following example returns the text objects that intersect the specified polygon:

```
prompt> get_text -intersect {{50 10} {80 10} {80 20} {50 20} {50 10}}  
{"TEXT#6415"}
```

The following example returns all the text objects in the current design that contain the text "helloworld":

```
prompt> get_text * -filter "text == helloworld"  
{"TEXT#6400" "TEXT#6401"}
```

The following example returns the text objects on the m1 layer:

```
prompt> get_text * -filter "layer == m1"  
{"TEXT#6400" "TEXT#6402"}
```

The following example issues a warning message because there are no text objects that matches the specified pattern:

```
prompt> get_text TEXT#1234  
Warning: Nothing implicitly matched 'TEXT#1234' (SEL-003)
```

SEE ALSO

```
collections(2)  
create_text(2)  
get_attribute(2)  
query_objects(2)  
remove_text(2)
```

get_timing_paths

Creates a collection of timing paths for custom reporting and other processing.

SYNTAX

```
collection get_timing_paths
[-to to_list
  | -rise_to rise_to_list
  | -fall_to fall_to_list]
[-from from_list
  | -rise_from rise_from_list
  | -fall_from fall_from_list]
[-through through_list]
[-rise_through rise_through_list]
[-fall_through fall_through_list]
[-delay_type delay_type]
[-nworst paths_per_endpoint]
[-max_paths max_path_count]
[-enable_preset_clear_arcs]
[-group group_name]
[-true [-true_threshold path_delay]]
[-slack_greater_than greater_slack_limit]
[-slack_lesser_than lesser_slack_limit]
[-greater greater_limit]
[-lesser lesser_limit]
[-include_hierarchical_pins]
[-path_type full_clock_expanded | full]
```

Data Types

<i>to_list</i>	list
<i>rise_to_list</i>	list
<i>fall_to_list</i>	list
<i>from_list</i>	list
<i>rise_from_list</i>	list
<i>fall_from_list</i>	list
<i>through_list</i>	list
<i>rise_through_list</i>	list
<i>fall_through_list</i>	list
<i>delay_type</i>	string
<i>paths_per_endpoint</i>	integer
<i>max_path_count</i>	integer
<i>group_name</i>	list
<i>path_delay</i>	float
<i>greater_slack_limit</i>	float
<i>lesser_slack_limit</i>	float
<i>greater_limit</i>	float
<i>lesser_limit</i>	float

ARGUMENTS

`-to to_list`

Specifies the to pins, ports, nets, or clocks. Path endpoints are the output ports or data pins of registers. If you specify a clock, the command considers all endpoints that are constrained by the clock.

`-rise_to rise_to_list`

This option is similar to the **-to** option, but applies only to paths rising at the endpoint. If you specify a clock, this option selects the paths that are captured by the rising edge of the clock at the clock source, taking into account any logical inversions along the clock path.

`-fall_to fall_to_list`

This option is similar to the **-to** option, but applies only to paths falling at the endpoint. If you specify a clock, this option selects the paths that are captured by the falling edge of the clock at the clock source, taking into account any logical inversions along the clock path.

`-from from_list`

Specifies the from pins, ports, nets, or clocks. Path startpoints are the input ports or clock pins of registers. If you specify a clock, the command considers all startpoints clocked by the clock.

`-rise_from rise_from_list`

This option is similar to the **-from** option, but applies only to paths rising from the specified objects. If you specify a clock, this option selects the startpoints whose paths are launched by the rising edge of the clock at the clock source, taking into account any logical inversions along the clock path.

`-fall_from fall_from_list`

This option is similar to the **-from** option, but applies only to paths falling from the specified objects. If you specify a clock, this option selects the startpoints whose paths are launched by the falling edge of the clock at the clock source, taking into account any logical inversions along the clock path.

`-through through_list`

Reports only paths that pass through the named pins, ports, or clocks. If you do not use the **-through** option, the command defaults to reporting the longest path to an output port if the design has no timing constraints. If it has timing constraints, the default is to report the path with the worst slack within each path group if you do not use the **-group** option. If you use **-group**, the default is to report the path with the worst slack within the group specified by the *group_name* value.

If you specify the **-through** option only once, the tool reports only the paths that travel through one or more of the objects in the list. You can use **-through** more than once in one command invocation. For a discussion of the use of multiple **-through** options, see the *DESCRIPTION* section.

`-rise_through rise_through_list`

This option is similar to the **-through** option, but applies only to paths with a rising transition at the specified objects. You can specify **-rise_through** more than once in a single command invocation. For a discussion of multiple

-through, **-rise_through**, and **-fall_through** options, see the DESCRIPTION section.

-fall_through *fall_through_list*

This option is similar to the **-through** option, but applies only to paths with a falling transition at the specified objects. You can specify **-fall_through** more than once in a single command invocation. For a discussion of multiple **-through**, **-rise_through**, and **-fall_through** options, see the DESCRIPTION section.

-delay_type *delay_type*

Specifies the type of path delay. The valid values are **max**, **min**, **min_max**, **max_rise**, **max_fall**, **min_rise**, and **min_fall**. The rise or fall in the *delay_type* refers to a rising or falling transition at the path endpoint.

-nworst *paths_per_endpoint*

Gets *n* worst paths to the endpoint, where *paths_per_endpoint* is greater than or equal to 1. The default is 1, which indicates that only the worst path to an endpoint is considered. Specifying larger a larger value than 1 increases the runtime.

-max_paths *max_path_count*

Specifies the maximum number of paths to get per path group, where *max_path_count* is greater than or equal to 1. The default is 1.

-enable_preset_clear_arcs

Enables asynchronous preset and clear arcs for the timing path collection. By default, asynchronous timing arcs are disabled during timing verification.

-group *group_name*

Restricts the collection to paths in this *group_name*. Use the **group_path** or **create_clock** command to group paths.

-true

Reports the longest (least-slack) true paths in the design. This option can require a long runtime for a design that has many false paths. When you use the **-true_threshold** option, then the **-true** option returns the first path it finds greater than or equal to *path_delay*, rather than continuing to search for a longer path.

You can use the **true_delay_prove_true_backtrack_limit** and **true_delay_prove_false_backtrack_limit** variables to limit the amount of backtracking during the operation of **get_timing_paths -true**. You can use the **set_case_analysis** command to specify a partial input vector to be considered for the **-true** analysis.

The **-true** option cannot be combined with the **-max_paths(>1)**, **-nworst(>1)**, **-lesser**, **-greater**, **-slack_lesser_than**, **-slack_greater_than**, or **-delay_type** (path type other than *max*) options.

-true_threshold *path_delay*

Specifies a threshold path delay value, in library time units, used by the **-true** option to speed up searching. When you use the **-true_threshold** option, then the **-true** option returns the first path it finds greater than or equal to *path_delay*, rather than continuing to search for a longer path.

`-slack_greater_than greater_slack_limit`
 Selects only those paths with a slack greater than *greater_slack_limit*. The **-slack_greater_than** option can be combined with the **-slack_lesser_than** option to select only those paths inside or outside of a given slack range.

`-slack_lesser_than lesser_slack_limit`
 Selects only those paths with a slack less than *lesser_slack_limit*. The **-slack_greater_than** option can be combined with the **-slack_lesser_than** option to select only those paths inside or outside of a given slack range.

`-greater greater_limit`
 Selects only those paths that have a delay greater than the *greater_limit* value. The **-greater** and **-lesser** options can be combined to select only those paths inside or outside of a given delay range.

`-lesser lesser_limit`
 Selects only those paths that have a delay less than the *lesser_limit* value. The **-greater** and **-lesser** options can be combined to select only those paths inside or outside of a given delay range.

`-include_hierarchical_pins`
 Specifies for the returned timing paths to contain points for each hierarchical pin crossed, as well as for all leaf pins in the path. It also sets to **true** their **hierarchical** attributes. Note that clock paths are always hierarchical.

`-path_type full_clock_expanded | full`
 Specifies whether or not to calculate launch and capture clock paths (if they exist) for the selected paths. The valid values are **full_clock_expanded** and **full**. When you specify *full_clock_expanded*, the tool calculates clock paths. When you specify **full**, it does not calculate them. You can access the clock path data of a path (if calculated) via its **launch_clock_paths** and **capture_clock_paths** attributes.

DESCRIPTION

This command creates a collection of paths for custom reporting or other operations. You can use the **foreach_in_collection** command to iterate among the paths in the collection. You can use the **get_attribute** and **collection** commands to obtain information about the paths. The following attributes are supported on timing paths:

```
capture_clock_paths
clock_uncertainty
clock_path
crpr_common_point
crpr_value
endpoint
endpoint_clock
endpoint_clock_close_edge_type
endpoint_clock_close_edge_value
endpoint_clock_is_inverted
endpoint_clock_is_propagated
endpoint_clock_latency
```

```

endpoint_clock_open_edge_type
endpoint_clock_open_edge_value
endpoint_clock_pin
endpoint_hold_time_value
endpoint_is_level_sensitive
endpoint_output_delay_value
endpoint_recovery_time_value
endpoint_removal_time_value
endpoint_setup_time_value
hierarchical
launch_clock_paths
object_class
path_group
path_type
points
slack
startpoint
startpoint_clock
startpoint_clock_is_inverted
startpoint_clock_is_propagated
startpoint_clock_latency
startpoint_clock_open_edge_type
startpoint_clock_open_edge_value
startpoint_input_delay_value
startpoint_is_level_sensitive
time_borrowed_from_endpoint
time_lent_to_startpoint

```

One attribute of a timing path is the **points** collection. A point corresponds to a pin or port along the path. You can iterate through these points by using the **foreach_in_collection** command and get their attributes by using the **get_attribute** command. The following attributes are available for points of a timing path:

```

arrival
object
object_class
rise_fall
slack

```

See the **collections** and **foreach_in_collection** man pages for detailed information.

By default, the **get_timing_paths** command uses a reporting engine that can run extremely fast on large designs, especially for larger values of the **nworst** and **max_paths** arguments. For designs that have multiple timing paths with identical slack, this engine might report different paths (with the same slack value) than a previous engine that was once in use.

You can use multiple iterations of the **-through**, **-rise_through**, and **-fall_through** options in a single command to specify paths that traverse multiple points in the design. The following example specifies paths beginning at A1, passing through B1, then through C1, and ending at D1.

```
prompt> get_timing_paths -from A1\
```


-through B1 -through C1 -to D1

If more than one object is specified by one **-through** option, the path can pass through any of the objects. The following example specifies paths beginning at A1, passing through either B1 or B2, then passing through either C1 or C2, and ending at D1.

```
prompt> get_timing_paths -from A1 -through {B1 B2} -through {C1 C2} -to D1
```

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example prints out the startpoint name, endpoint name, and slack of the worst path in each path group.

```
proc custom_report_worst_path_per_group {} {      echo [format "%-20s %-
20s %7s" "From" "To" "Slack"]
    echo "-----"
    foreach_in_collection path [get_timing_paths] {      set slack [get_attribute
$path slack]
        set startpoint [get_attribute $path startpoint]
        set endpoint [get_attribute $path endpoint]
        echo [format "%-20s %-20s %s" [get_attribute $startpoint full_name] \
            [get_attribute $endpoint full_name] $slack]
    }
}
```

```
prompt> custom_report_worst_path_per_group
```

From	To	Slack

ffa/CP	QA	0.1977
ffb/CP	ffd/D	3.8834

The following example shows total negative slack, total positive slack, and worst negative slack for the current design.

```
proc report_design_slack_information {} {  set design_tns 0
    set design_wns 100000
    set design_tps 0
    foreach_in_collection group [get_path_groups *] {      set group_tns 0
        set group_wns 100000
        set group_tps 0
        foreach_in_collection path [get_timing_paths -nworst 10000 -
group $group] {      set slack [get_attribute $path slack]
            if {$slack < $group_wns} {      set group_wns $slack
                if {$slack < $design_wns} {      set design_wns $slack
```

```

}
}
    if {$slack < 0.0} {        set group_tns [expr $group_tns + $slack]
} else {        set group_tps [expr $group_tps + $slack]
}
}

set design_tns [expr $design_tns + $group_tns]
set design_tps [expr $design_tps + $group_tps]
set group_name [get_attribute $group full_name]
echo [format "Group '%s' Worst Negative Slack : %g" $group_name $group_wns]
echo [format "Group '%s' Total Negative Slack : %g" $group_name $group_tns]
echo [format "Group '%s' Total Positive Slack : %g" $group_name $group_tps]
echo ""
}
echo "-----"
echo [format "Design Worst Negative Slack : %g" $design_wns]
echo [format "Design Total Negative Slack : %g" $design_tns]
echo [format "Design Total Positive Slack : %g" $design_tps]
}

```

prompt> **report_design_slack_information**

```

Group 'CLK' Worst Negative Slack : -3.1166
Group 'CLK' Total Negative Slack : -232.986
Group 'CLK' Total Positive Slack : 4.5656

Group 'vclk' Worst Negative Slack : -4.0213
Group 'vclk' Total Negative Slack : -46.1982
Group 'vclk' Total Positive Slack : 0

```

```

-----
Design Worst Negative Slack : -4.0213
Design Total Negative Slack : -279.184
Design Total Positive Slack : 4.5656

```

SEE ALSO

```

collections(2)
create_clock(2)
foreach_in_collection(2)
get_attribute(2)
group_path(2)
report_timing(2)

```

get_tracks

Creates a collection of tracks from the current design.

SYNTAX

```
collection get_tracks
[-within rectangle | -intersect rectangle
 | -at at_point | -touching rectangle]
[-filter expression]
[patterns | -of_objects layers]
[-quiet]
```

Data Types

<i>rectangle</i>	list of points
<i>at_point</i>	point
<i>expression</i>	string
<i>patterns</i>	string
<i>layers</i>	list

ARGUMENTS

-within *rectangle*

Creates a collection containing all tracks within the specified rectangle. The format of a rectangle specification is `{{llx lly} {urx ury}}`, which specifies the lower-left and upper-right corners of the rectangle. The coordinate unit is specified in the technology file; typically it is microns.

The **-within**, **-intersect**, **-at**, and **-touching** options are mutually exclusive; you can specify only one of these options. If you do not specify any of these options, the command returns all tracks in the current design that meet the selection criteria.

-intersect *rectangle*

Creates a collection containing all tracks that intersect the specified rectangle. A track intersects a rectangle if it is neither completely outside the rectangle nor completely within it. The format of a rectangle specification is `{{llx lly} {urx ury}}`, which specifies the lower-left and upper-right corners of the rectangle.

The coordinate unit is specified in the technology file; typically it is microns.

The **-within**, **-intersect**, **-at**, and **-touching** options are mutually exclusive; you can specify only one of these options. If you do not specify any of these options, the command returns all tracks in the current design that meet the selection criteria.

-at *at_point*

Creates a collection containing all tracks that overlap the specified point. The format of a point specification is `{x y}`.

The coordinate unit is specified in the technology file; typically it is microns.

The **-within**, **-intersect**, **-at**, and **-touching** options are mutually exclusive;

you can specify only one of these options. If you do not specify any of these options, the command returns all tracks in the current design that meet the selection criteria.

-touching *rectangle*

Creates a collection containing all tracks that touch the specified rectangle. A track touches a rectangle if it is either completely or partially within the specified rectangle. The format of a rectangle specification is `{{llx lly} {urx ury}}`, which specifies the lower-left and upper-right corners of the rectangle.

The coordinate unit is specified in the technology file; typically it is microns.

The **-within**, **-intersect**, **-at**, and **-touching** options are mutually exclusive; you can specify only one of these options. If you do not specify any of these options, the command returns all tracks in the current design that meet the selection criteria.

-filter *expression*

Filters the collection with *expression*, which must be a Boolean expression based on the following track attributes:

Attribute names	Description
bbox	The bounding box of the track
name	Name of the track
layer	The name of the layer that the track is on
direction	The wire direction of the object
count	The number of grid in a track object
space	The track step of the objects
start	The start position of the object
stop	The stop position of the object

The tool evaluates the expression for any tracks that match the specified criteria. If the expression evaluates to true, the track is included in the result.

patterns

Matches the track names in the current design against the specified patterns. You can specify the patterns by using the following formats:

- A collection of tracks
- An asterisk (*), which indicates all tracks

- Track names, which are in the format `TRACK_object_id`

The *patterns* argument and **-of_objects** option are mutually exclusive. You can specify only one of these options. If you do not specify either option, the command uses * (asterisk) as the pattern.

-of_objects *layers*

Creates a collection of tracks that are located on the specified layers. Each element of the layer list can be a layer name, layer name pattern, or a layer collection.

The *patterns* argument and **-of_objects** option are mutually exclusive. You can specify only one of these options.

-quiet

Suppresses warning and error messages if no objects match. Syntax error

messages are not suppressed.

DESCRIPTION

This command returns a collection of tracks of the current design that meet the selection criteria. You can use this command to query the tracks inside the Milkyway database by layer, by point, or by region. You can also use this command to query tracks by filtering their attributes.

EXAMPLES

The following example returns the tracks within the rectangle with the lower-left corner at {2 2} and the upper-right corner at {25 25}.

```
prompt> get_tracks * -within {{2 2} {25 25}}
{"TRACK_5389"}
```

The following example returns the tracks that are located on the METAL2 layer.

```
prompt> get_tracks * -filter "layer~=METAL2"
{"TRACK_4683"}
```

The following example returns all tracks.

```
prompt> get_tracks "TRACK_*"
{"TRACK_4683 TRACK_5389"}
```

The following example shows how to use the **get_attribute** command to query the value of a track attribute.

```
prompt> get_attribute [get_tracks TRACK_4683] layer
METAL2
prompt> get_attribute [get_tracks TRACK_4683] bbox
{{0.000 0.000} {100.000 100.000}}
prompt> get_attribute [get_tracks TRACK_4683] start
0.000 0.000
```

To get a complete report of the attribute values of all tracks, use the **report_attribute** command, as shown in the following example.

```
prompt> report_attribute -application [get_tracks TRACK_4683]
Design  Object      Type   Attribute Name  Value
-----
CORE    TRACK_4683    string  bbox            {0.000 0.000} {100.000 100.000}
CORE    TRACK_4683    int      cell_id         3
CORE    TRACK_4683    string   layer           METAL2
CORE    TRACK_4683    string   name            TRACK_4683
CORE    TRACK_4683    string   object_class    track
CORE    TRACK_4683    int      object_id       4683
CORE    TRACK_4683    string   direction       horizontal
CORE    TRACK_4683    string   start           0.000 0.0000
CORE    TRACK_4683    string   stop            100.000 100.000
CORE    TRACK_4683    int      count           51
```

CORE	TRACK_4683	int	space	2.000

SEE ALSO

`create_track(2)`
`get_attribute(2)`
`remove_track(2)`
`report_attribute(2)`

get_user_grid

Gets the user grid for this session.

SYNTAX

```
status get_user_grid  
[design]
```

Data Types

design string

ARGUMENTS

design

Specifies the design in which to get the user grid information. By default, the tool gets global user grid information, which is used for the default value of each successive run of this command.

DESCRIPTION

This command gets the user grid for this session of the tool. The return value is a list in the following format.

```
{{x_offset y_offset} {x_step y_step}}
```

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example gets the global user grid for this session of the tool.

```
prompt> get_user_grid  
{{0 0} {0.01 0.01}}  
1
```

SEE ALSO

set_user_grid(2)

get_user_shapes

Creates a collection of one or more user shapes.

SYNTAX

```
collection get_user_shapes
[-filter expression]
[-quiet]
[-regex [-nocase] | -exact]
[-within region
  | -intersect region
  | -touching region]
[patterns]
```

Data Types

<i>expression</i>	string
<i>region</i>	list of points
<i>patterns</i>	list

ARGUMENTS

-filter *expression*

Filters the collection with the value specified by *expression*. For any user shapes that match the *patterns* option, the expression is evaluated based on the shape's attributes. If the expression evaluates to true, the user shape is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regex

Uses the value of the *patterns* argument as a real regular expression rather than as a simple wildcard pattern. This option also modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions, rather than with simple wildcard patterns. The **-regex** and **-exact** options are mutually exclusive; you can use only one.

-nocase

Makes matches case-insensitive when you use it with the **-regex** option. You can use the **-nocase** option only when you also use the **-regex** option.

-exact

Disables simple pattern matching. You can use this option when searching for objects that contain the * (asterisk) and ? (question mark) wildcard characters. The **-regex** and **-exact** options are mutually exclusive; you can use only one.

-within *region*

Creates a collection containing only user shapes within the specified region. The region boundary can be rectangle or polygon. The valid format of a

rectangle is `{{llx lly} {urx ury}}` or `{llx lly urx ury}`, which specifies the lower-left and upper-right corners of the rectangle. The valid format of a polygon is: `{{x1 y1} {x2 y2} ... {xN yN} {x1 y1}}`, and each `{x y}` pair specifies one point of the input polygon. Besides, a list of one polygon is also supported as input for this option, with the format: `{{{x1 y1} {x2 y2} ... {xN yN} {x1 y1}}}`. Pay attention that the valid polygon is rectilinear, so the adjacent points have one same coordinate.

The **-within**, **-intersect**, and **-touching** options are mutually exclusive; you can specify only one.

-intersect region

Creates a collection containing only user shapes intersect the specified region. The region boundary can be rectangle or polygon. The valid format of a rectangle is `{{llx lly} {urx ury}}` or `{llx lly urx ury}`, which specifies the lower-left and upper-right corners of the rectangle. The valid format of a polygon is: `{{x1 y1} {x2 y2} ... {xN yN} {x1 y1}}`, and each `{x y}` pair specifies one point of the input polygon. Besides, a list of one polygon is also supported as input for this option, with the format: `{{{x1 y1} {x2 y2} ... {xN yN} {x1 y1}}}`. Pay attention that the valid polygon is rectilinear, so the adjacent points have one same coordinate.

The **-within**, **-intersect**, and **-touching** options are mutually exclusive; you can specify only one.

-touching region

Creates a collection containing only user shapes touching the specified region. The region boundary can be rectangle or polygon. The valid format of a rectangle is `{{llx lly} {urx ury}}` or `{llx lly urx ury}`, which specifies the lower-left and upper-right corners of the rectangle. The valid format of a polygon is: `{{x1 y1} {x2 y2} ... {xN yN} {x1 y1}}`, and each `{x y}` pair specifies one point of the input polygon. Besides, a list of one polygon is also supported as input for this option, with the format: `{{{x1 y1} {x2 y2} ... {xN yN} {x1 y1}}}`. Pay attention that the valid polygon is rectilinear, so the adjacent points have one same coordinate.

The **-within**, **-intersect**, and **-touching** options are mutually exclusive; you can specify only one.

patterns

Matches the user shape names against patterns in the current design. You can specify the patterns by using the following formats:

- ***** (asterisk) - allows all user shapes.
- **RECTANGLE#*** - allows rectangle user shapes.
- **RECTANGLE#** *object_id* - allows all rectangle user shapes that have a specified object ID.
- **TRAPEZOID#*** - allows all trapezoid user shapes.
- **TRAPEZOID#** *object_id* - allows all trapezoid user shapes that have a specified object ID.

Patterns can include the wildcard characters the ***** (asterisk) and **?** (question mark) wildcard characters or regular expressions, depending on the use of the **-regexp** option. For more details about using and escaping wildcards, see the **wildcards** man page.

If you do not specify either the *patterns* and **-of_objects** options, the command uses *** as the pattern by default.
The *patterns* and **-of_objects** options are mutually exclusive; you can use only one.

DESCRIPTION

This command creates a collection of user shapes.

When using **-within**, or **-intersect**, note that the behavior will be different for the different format of the specified region. A shape object has its boundary and bbox, sometimes they are different. When the specified region is in rectangle format, we will search objects by their bbox; when the specified region is in polygon format, we will search objects by their boundary. And searching by bbox is faster than searching by boundary.

You can use the **get_user_shapes** command at the command prompt or nest it as an argument to another command, such as **query_objects**. You can also assign the **get_user_shapes** result to a variable.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example queries the shapes in a specified area. Although the output looks like a list, it is only a display.

```
prompt> get_user_shapes\  
-within {{10.5 22.3} {32.0 40}}  
{RECTANGLE#100 "TRAPEZOID#120 POLYGON#145}
```

The following example gets user shapes that intersect the specified polygon:

```
prompt> get_user_shapes \  
-intersect {{30 20} {50 20} {50 30} {40 30} {40 40} {30 40} {30 20}}  
{POLYGON#9936}
```

The following example queries shapes that have the specified route type. Although the output looks like a list, it is only a display.

```
prompt> get_user_shapes\  
-filter {route_type=="P/G Std. Cell Pin Conn"}  
{PATH#1134336 PATH#1134337 PATH#1134338 PATH#1134339 PATH#1134340}
```

SEE ALSO

`collections(2)`
`create_user_shape(2)`
`filter_collection(2)`
`get_net_shapes(2)`
`query_objects(2)`
`remove_user_shape(2)`

get_via_masters

Returns a name list of contact codes (via masters) defined in the current library.

SYNTAX

```
status get_via_masters
[-of_objects via_list]
-cut_layer cut_layer
-up_layer upper_layer
-low_layer lower_layer
patterns
```

Data Types

<i>via_list</i>	list
<i>cut_layer</i>	list
<i>upper_layer</i>	list
<i>lower_layer</i>	list
<i>patterns</i>	list

ARGUMENTS

-of_objects *via_list*
Gets the via master of the specified vias.

-cut_layer *cut_layer*
Creates a collection containing the via masters with the specified cut layer.

-up_layer *upper_layer*
Creates a collection containing the via masters with the specified upper layer.

-low_layer *lower_layer*
Creates a collection containing the via masters with the specified lower layer.

patterns
Matches via master names against patterns. Patterns can include the wildcard character * (asterisk) and the ? (question mark).

DESCRIPTION

This command returns a name list of via master names that are defined in the technology file of the current library. You can use the name of a contact code in the **create_via** command to specify the master of a via.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example returns the via masters whose lower layer is METAL1.

```
prompt> get_via_masters -low_layer METAL1  
VIA12 VIA12FAT  
1
```

The following example returns one via master that can be used by other commands.

```
prompt> create_via -at {356 278} -no_net\  
-master [get_via_masters VIA12  
{"VIA#9051"}]
```

SEE ALSO

`create_via(2)`

get_vias

Creates a collection by selecting vias from the current design.

SYNTAX

```
collection get_vias
[-within region
 | -intersect region
 | -touching region
 | -at point]
[-filter expression]
[-of_objects net_list]
[-quiet]
[patterns]
```

Data Types

<i>region</i>	list
<i>point</i>	string
<i>expression</i>	string
<i>net_list</i>	list
<i>patterns</i>	list

ARGUMENTS

-within *region*

Creates a collection containing all vias within the specified region. The region boundary can be rectangle or polygon. The valid format of a rectangle is `{{llx lly} {urx ury}}` or `{llx lly urx ury}`, which specifies the lower-left and upper-right corners of the rectangle. The valid format of a polygon is: `{{x1 y1} {x2 y2} ... {xN yN} {x1 y1}}`, and each `{x y}` pair specifies one point of the input polygon. Besides, a list of one polygon is also supported as input for this option, with the format: `{{{x1 y1} {x2 y2} ... {xN yN} {x1 y1}}}`. Pay attention that the valid polygon is rectilinear, so the adjacent points have one same coordinate.

The **-within**, **-intersect**, **-touching**, and **-at** options are mutually exclusive; you can specify only one.

-intersect *region*

Creates a collection containing all vias intersect the specified region. The region boundary can be rectangle or polygon. The valid format of a rectangle is `{{llx lly} {urx ury}}` or `{llx lly urx ury}`, which specifies the lower-left and upper-right corners of the rectangle. The valid format of a polygon is: `{{x1 y1} {x2 y2} ... {xN yN} {x1 y1}}`, and each `{x y}` pair specifies one point of the input polygon. Besides, a list of one polygon is also supported as input for this option, with the format: `{{{x1 y1} {x2 y2} ... {xN yN} {x1 y1}}}`. Pay attention that the valid polygon is rectilinear, so the adjacent points have one same coordinate.

The **-within**, **-intersect**, **-touching**, and **-at** options are mutually exclusive; you can specify only one.

-touching *region*

Creates a collection containing all vias touching the specified region. The region boundary can be rectangle or polygon. The valid format of a rectangle is `{{llx lly} {urx ury}}` or `{llx lly urx ury}`, which specifies the lower-left and upper-right corners of the rectangle. The valid format of a polygon is: `{{x1 y1} {x2 y2} ... {xN yN} {x1 y1}}`, and each `{x y}` pair specifies one point of the input polygon. Besides, a list of one polygon is also supported as input for this option, with the format: `{{{x1 y1} {x2 y2} ... {xN yN} {x1 y1}}}`. Pay attention that the valid polygon is rectilinear, so the adjacent points have one same coordinate.

The **-within**, **-intersect**, **-touching**, and **-at** options are mutually exclusive; you can specify only one.

-at *point*

Creates a collection containing all vias that overlap the point specified by the coordinates `{x y}`. The **-within**, **-intersect**, **-touching**, and **-at** options are mutually exclusive; you can specify only one.

-filter *expression*

Filters the collection with *expression*. For any vias that match the patterns option, the expression is evaluated based on the via's attributes. If the expression evaluates to *true*, the via is included in the result.

Use the **list_attributes** command to determine the via attributes.

-of_objects *net_list*

Creates a collection containing the vias connected to the specified nets.

-quiet

Suppresses warning and error messages.

patterns

Matches the via names against the patterns in the current design. You can specify the patterns with the following formats:

- `*` (asterisk) indicates all vias
- `via#*` indicates all vias
- `via#7689` indicates one route shape whose `object_id` is 7689.

If both this argument and **-of_objects** are omitted, the command uses `*` (asterisk) as the *pattern*.

DESCRIPTION

This command creates a collection of vias by selecting vias from the current design that meet the selection criteria. It returns a collection handle if one or more vias meet the selection criteria. If no vias match the selection criteria, it returns an empty string.

Use the **get_vias** command as an argument to another command or assign its result to a

variable. Refer to the example below for more information.

See the collection command man page for information about working with collections.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following examples create via collections:

```
prompt> get_vias -of_objects n300
{"VIA6766"}
```

The following examples get all vias within specified rectangle:

```
prompt> get_vias -within {{100 100} {105 105}}
{"VIA#12345"}
```

The following example gets vias intersect specified polygon:

```
prompt>get_vias -intersect \
{{30 20} {50 20} {50 30} {40 30} {40 40} {30 40} {30 20}}
{VIA#9936 VIA#1234}
```

SEE ALSO

```
create_net_shape(2)
create_via(2)
get_attribute(2)
query_objects(2)
remove_net_shape(2)
remove_via(2)
set_via_array_size(2)
```


get_voltage_areas

Creates a collection of voltage areas from the current design.

SYNTAX

```
collection get_voltage_areas
[-quiet]
[-regex [-nocase] | -exact]]
[-filter expression]
patterns | -of_objects cell_list
```

Data Types

<i>expression</i>	string
<i>patterns</i>	list
<i>cell_list</i>	list

ARGUMENTS

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regex

Uses the value of the *patterns* option as a real regular expression rather than as a simple wildcard pattern. This option also modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions, rather than with simple wildcard patterns. The **-regex** and **-exact** options are mutually exclusive; you can use only one.

-nocase

You can use the **-nocase** option only when you also use the **-regex** option.

-exact

Disables simple pattern matching. You can use this option when searching for objects that contain the * (asterisk) and ? (question mark) wildcard characters. The **-regex** and **-exact** options are mutually exclusive; you can use only one.

-filter *expression*

Filters the collection with the value specified by *expression*. For any voltage areas that match the *patterns* argument, the expression is evaluated based on the voltage area's attributes. If the expression evaluates to true, the voltage area is included in the result.

patterns

Matches voltage area names against patterns. Patterns can include the wildcard characters "*". If this option is not specified, this command uses ,Ãú*,Ãù as the default pattern.

Matches voltage area names against patterns. Patterns can include the wildcard characters the * (asterisk) and ? (question mark) wildcard characters or regular expressions, depending on the use of the **-regex**

option. For more details about using and escaping wildcards, see the **wildcards** man page. The **-of_objects** and *patterns* options are mutually exclusive; you can use only one. By default, the command uses * (asterisk) as the *pattern*.

-of_objects *cell_list*

Creates a collection of voltage areas that contains the specified objects. The *cell_list* value should be a list of hierarchical cells. The **-of_objects** and *patterns* options are mutually exclusive; you can use only one.

DESCRIPTION

This command creates a collection of voltage areas from the current design that match certain criteria. The command returns a collection handle (identifier) if any voltage areas match the value of the *pattern* option and pass the filter (if specified). If no objects match the criteria, the command returns an empty string.

You can use the **get_voltage_areas** command at the command prompt or nest it as an argument to another command (such as **report_voltage_area**). You can also assign the **get_voltage_areas** result to a variable.

When issued from the command prompt, **get_voltage_areas** behaves as if you had called **report_voltage_area** to report the objects in the collection. By default, a maximum of 100 objects is displayed. You can change the maximum by using the **collection_result_display_limit** variable.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

SEE ALSO

`create_voltage_area(2)`
`remove_voltage_area(2)`
`report_voltage_area(2)`
`update_voltage_area(2)`

get_zero_interconnect_delay_mode

Reports whether or not the timer is currently using zero interconnect delay mode.

SYNTAX

status **get_zero_interconnect_delay_mode**

ARGUMENTS

None.

DESCRIPTION

This command checks the timer to determine if it is currently using zero interconnect delay mode. The command returns 1 if the timer is using zero interconnect delay mode; otherwise it returns 0.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

SEE ALSO

set_zero_interconnect_delay_mode(2)

get_zrt_net_properties

Gets the value of the specified Zroute property for the specified net.

SYNTAX

```
status get_zrt_net_properties
-net net_name
-property property_name
```

Data Types

<i>net_name</i>	collection of one item
<i>property_name</i>	string

ARGUMENTS

```
-net net_name
    Specifies the net for which to get a Zroute property value. You can specify
    only a single net.
    This is a required option.

-property property_name
    Specifies the Zroute property whose value is to be returned. You can specify
    the following value for this option: ignore_voltage_areas.
    This is a required option.
```

DESCRIPTION

The **get_zrt_net_properties** command gets the value of the specified Zroute property for the specified net.

EXAMPLES

The following example gets the value of the **ignore_voltage_areas** property on the n1 net:

```
prompt> get_zrt_net_properties -net "n1" -property ignore_voltage_areas
```

SEE ALSO

```
set_zrt_net_properties(2)
report_zrt_net_properties(2)
```

getenv

Returns the value of a system environment variable.

SYNTAX

```
string getenv  
variable_name
```

Data Types

```
variable_name      string
```

ARGUMENTS

```
variable_name  
    Indicates the name of the environment variable to be retrieved.
```

DESCRIPTION

The **getenv** command searches the system environment for the specified *variable_name* and sets the result of the command to the value of the environment variable. If the variable is not defined in the environment, the command returns a Tcl error. The command is catchable.

Environment variables are stored in the Tcl array variable *env*. The environment commands **getenv**, **setenv**, and **printenv** are convenience functions to interact with this array.

The application you are running inherited the initial values for environment variables from its parent process (that is, the shell from which you invoked the application). If you set the variable to a new value using **setenv**, you see the new value within the application and within any new child processes you initiate from the application using the **exec** command. However, these new values are not exported to the parent process. Further, if you set an environment variable using the appropriate system command in a shell you invoke using the **exec** command, that value is not reflected in the current application.

See the **setB**, **unset**, and **printvar** commands for information about dealing with non-environment variables.

EXAMPLES

In the following example, **getenv** is used to return you to your home directory.

```
prompt> set home [getenv "HOME"]  
/users/disk1/bill  
prompt> cd $home  
prompt> pwd  
/users/disk1/bill
```

In the following example, **setenv** is used to change the value of an environment variable.

```
prompt> getenv PRINTER
laser1
prompt> setenv PRINTER "laser3"
laser3
prompt> getenv PRINTER
laser3
```

In the following example, the environment variable requested is not defined. Note that the error message shows that the Tcl variable *env* was indexed with the value **UNDEFINED**, which resulted in an error. In the second command, **catch** was used to suppress the message.

```
prompt> getenv "UNDEFINED"
Error: can't read "env(UNDEFINED)": no such element in array
      Use error_info for more info. (CMD-013)
prompt> if {[catch {getenv "UNDEFINED"} msg]} {
      setenv UNDEFINED 1
}
```

SEE ALSO

```
printenv(2)
printvar(2)
setenv(2)
```

group

Creates a new level of hierarchy.

SYNTAX

```
status group
[cell_list | -logic | -pla | -fsm]
[-soft
  | -hdl_block block_name
  | -hdl_all_blocks
  | -hdl_bussed]
[-design_name design_name]
[-cell_name cell_name]
[-except exclude_list]
```

Data Types

<i>cell_list</i>	list
<i>block_name</i>	string
<i>design_name</i>	string
<i>cell_name</i>	string
<i>exclude_list</i>	list

ARGUMENTS

cell_list

Groups a list of cells in the current design into a new level of hierarchy. If more than one cell is specified, they must be enclosed in {} (braces). Cell names cannot be specified with the following options:

- logic
- pla
- fsm
- hdl_block
- hdl_all_blocks
- hdl_bussed

Note: Control logic cells (those designated as type "c" in the report displayed by using the **report_cell** command) must be grouped with the cells they are controlling.

-logic

Groups combinational cells into a new level of hierarchy. A cell is combinational if it is a leaf or library cell and its output pins have Boolean functions specified. The **-logic** option cannot be used with the *cell_list*, **-pla**, or **-fsm** option.

-pla

Groups programmable logic array (PLA) specifications in the design into a level of hierarchy. The **-pla** option cannot be used with the **-logic**, **-fsm**, or *cell_list* option. The **-pla** option is used only when the design contains blocks read as a PLA. To group combinational gates in the PLA output, use the **group** command with the **-logic -design_name** *design_name* options. After running the

compile command, the design can be written out as a PLA.

-fsm

Groups cells that are part of a finite-state machine design into a new level of hierarchy suitable for finite-state machine extraction. The sequential elements used to store the state of the finite-state machine are specified before using the **group** command by using the **set_fsm_state_vector** command. Cells in the transitive fanin or fanout of these state vector cells are grouped into the new hierarchy level. Noncombinational cells in the transitive fanin or fanout of the state vector cells are not included in the new hierarchy level. Upon completion of the group, the new design can be extracted into a state table format. This option cannot be used with the *cell_list*, **-logic** or **-pla** option.

-soft

Specifies that cells will be part of the same placement group. When you use this option, the **group** command sets the **group_name** attribute only to the selected cells, and the design hierarchy is not modified. Use the **-soft** option with the **-logic**, **-pla**, and **-fsm** options. You cannot use the **-soft** option with the **-hdl_block**, **-hdl_all_blocks**, or **-hdl_bussed** option. You can set the value of the **group_name** attribute by using the **-cell_name** option.

-hdl_block *block_name*

Groups all cells in a block created in the HDL file. The **-hdl_block** option requires that you specify the block in the HDL by using block labels. Block labels are separated by a / (slash), (for example, *my_process/my_function*).

-hdl_all_blocks

Groups all HDL blocks into a level of hierarchy. If you use the **-hdl_all_blocks** option alone, it recursively groups the HDL blocks in the current design. If you use it with the **-hdl_block** option, the blocks specified by the *block_name* argument are recursively grouped.

-hdl_bussed

Places each group of bused gates created from the HDL into a level of hierarchy. If you use it with the **-hdl_block** option, the blocks specified by the *block_name* argument are recursively grouped.

-design_name *design_name*

Names the design containing the new level of hierarchy. This name cannot already exist in the current design. Do not use the **-design_name** option, when you use the **-hdl_all_blocks** or **-hdl_bussed** option.

-cell_name *cell_name*

Names the instance of the new design. By default, the command generates a unique cell name. Do not use the **-cell_name** option, when you use the **-hdl_all_blocks** or **-hdl_bussed** option. Unique cell names are generated based on the **unique_cell_prefix** attribute in the current design. You can use the **get_attribute** command to obtain the value of the **unique_cell_prefix** attribute. You can set this value by using the **set_attribute** command. Note that there is no variable controlling the creation of unique cell names.

-except *exclude_list*

Specifies a list of cells in the current design to exclude from grouping. If more than one cell is specified, they must be enclosed in {} (braces). Do not

use the **-except** option with the **-hdl_block**, **-hdl_all_blocks**, or **-hdl_bussed** option.

DESCRIPTION

This command groups any number of cells or instances in the current design into a new design, creating a new level of hierarchy. The new design name must be defined by using the **-design_name** option (unless you are using the **-hdl_all_blocks** or **-hdl_bussed** option). The new design is copied into the current design.

To specify the cells to group into a new design, use one of the following options:

```
cell_list
-logic
-pla
-fsm
-hdl_block
-hdl_all_blocks
-hdl_bussed
```

Specify only one of these options (except for **-hdl_all_blocks** and **-hdl_bussed**, which can be used together or with **-hdl_block**). The **cell_list** can contain cells from anywhere in the hierarchy, but the parent of the cells in the list should be the same.

The instance name of the new design is set by using the **-cell_name** option. Otherwise, the following naming format is used:

```
cell{integer}
```

where *integer* is an integer value that ensures a unique name.

The hierarchy below a grouped block can also be grouped by issuing the **group** command specifying the new cell names.

Names for each level of hierarchy are automatically generated from the label of the grouped block. The names have the following format:

```
label_digits
```

where the *label* portion represents the label assigned to the block, and *digits* portion represents an integer that makes the name unique. **HDL_BLOCK** is the default label.

Ports in the new design are named the same as the nets to which they are connected in the current design. The direction of each port in the new design is determined for pins on the net according to the following table:

Inside pins	Outside pins	Resulting port direction
-----	-----	-----

IN	OUT	IN
OUT	IN	OUT
IN & OUT	OUT	IN
IN & OUT	IN	OUT
IN	IN & OUT	IN
OUT	IN & OUT	OUT
IN & OUT	IN & OUT	INOUT (bidir)

You can use the **-soft** option to define cell grouping constraints, which can be used by the layout placement tools and can be written to a file by using the **write_constraints** command. The **group -soft** command specifies that cells will be part of a same placement group. With this option, the **group** command sets only the **group_name** attribute for the cells in *cell_list* and the design hierarchy is not modified. You can set the value of the **group_name** attribute by using the **-cell_name** option. The **write_constraints** command reports cells with the **group_name** attribute as grouping constraints for placement tools. To remove the **group_name** attribute, use the **ungroup -soft** or **remove_attribute** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example groups two cells into a new design.

```
prompt> group {u1 u2} -design_name NEW\
-cell_name U
```

The following example groups cells with names beginning with ANALOG into a design.

```
prompt> group\
-design_name ANALOG_CELLS [get_cells ANALOG*]
```

The following example groups all cells in the HDL function named bar and process named proc into a design.

```
prompt> group -hdl_block proc/bar\
-design_name my_hdl_block
```

The following example sets the current design to the new design, and then uses the **group** command to group a nested function named my_other_function in the new design.

```
prompt> current_design my_hdl_block

prompt> group -hdl_block my_other_function
```

The following example groups all bused gates under the process named proc into separate levels of hierarchy (or designs).

```
prompt> group -hdl_block proc -hdl_bussed
```

The following example sets the `group_name` attribute to the name `proc` for all combinational cells. The design hierarchy is not modified.

```
prompt> group -logic -soft -cell_name proc
```

The following example creates a new hierarchy under a design named `BOT` with cells named `A`, `B`, and `C`. `PROC/U1` in an instance of `BOT`.

```
prompt> group\  
-design_name new_design {PROC/U1/A PROC/U1/B PROC/U1/C}
```

SEE ALSO

```
change_names(2)  
link(2)  
ungroup(2)  
current_design(3)
```

group_path

Groups a set of paths for cost function calculations.

SYNTAX

```
int group_path
[-weight weight_value]
[-critical_range range_value]
-default | -name group_name
[-from from_list
  | -rise_from rise_from_list
  | -fall_from fall_from_list]
[-through through_list]
[-rise_through rise_through_list]
[-fall_through fall_through_list]
[-to to_list
  | -rise_to rise_to_list
  | -fall_to fall_to_list]
```

Data Types

<i>weight_value</i>	float
<i>range_value</i>	float
<i>group_name</i>	string
<i>from_list</i>	list
<i>rise_from_list</i>	list
<i>fall_from_list</i>	list
<i>through_list</i>	list
<i>rise_through_list</i>	list
<i>fall_through_list</i>	list
<i>to_list</i>	list
<i>rise_to_list</i>	list
<i>fall_to_list</i>	list

ARGUMENTS

-weight *weight_value*

Specifies a cost function weight for this group. The *weight_value* must be a number between 0.0 and 100.0. The default is 1.0. A weight of 0.0 eliminates the paths in this group from cost function calculations. Do not use very small values (for example, 0.0001); smaller values can prevent **compile** from implementing small improvements to the design. If you specify **-weight** when you add members to an existing group, the new weight for the group is used.

-critical_range *range_value*

Specifies a margin of delay for *group_name* during optimization. *range_value* must be positive or 0.0. If you don't specify a *range_value* for a group, the default is the value set with the **set_critical_range** command (the default setting is 0.0). A *range_value* of 0.0 means that only the most critical paths (the ones with the worst violation) are optimized. If you specify a nonzero *range_value*, other near-critical violating paths (one per endpoint) within that amount of the worst path are also optimized if possible. If more than

one critical path to an endpoint must be optimized, use a separate **group_path** command for each distinct critical path to that endpoint. To force **compile** to optimize all violating paths, use a very large *range_value* (larger than any expected path violations).

-default

Specifies that endpoints or paths be moved to the default group and removed from the current group. You must specify **-default** unless you use **-name**; **-name** and **-default** are mutually exclusive.

-name *group_name*

Specifies a name for the group. If a group with this name already exists, the paths or endpoints are added to that group. Otherwise, a new group is created and named *group_name*. You must specify **-name** unless you use **-default**; **-name** and **-default** are mutually exclusive.

-from *from_list*

Specifies a list of names of clocks, ports, or pins that specify path startpoints. If a clock is given, all path startpoints related to that clock are implicitly included. This includes flip-flops in the transitive fanout of the clocks source pin or port, and ports that have input delay related to the clock.

-rise_from *rise_from_list*

Same as the **-from** option, except that the path must rise from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by rising edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of **-from**, **-rise_from**, and **-fall_from**.

-fall_from *fall_from_list*

Same as the **-from** option, except that the path must fall from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of **-from**, **-rise_from**, and **-fall_from**.

-through *through_list*

A list of path throughpoints (port, pin, or leaf cell names) of the current design. The path grouping applies only to paths that pass through one of the points in the *through_list*. If more than one object is included, the objects must be enclosed either in quotes or in '{}' braces. If the **-through** option is specified multiple times, the group path setting applies to paths that pass through a member of each *through_list* in the order the lists were given. In other words, the path must first pass through a member of the first *through_list*, then through a member of the second list, and so on for every through list specified. If the **-through** option is used in combination with the **-from** or **-to** options, the group path applies only if the **-from** or **-to** conditions are satisfied and the **-through** conditions are satisfied.

-rise_through *rise_through_list*

Same as the **-through** option, but applies only to paths with a rising transition at the specified objects. You can specify **-rise_through** more than once in a single command invocation as with the **-through** option.

`-fall_through fall_through_list`

Same as the **-through** option, but applies only to paths with a falling transition at the specified objects. You can specify **-fall_through** more than once in a single command invocation as with the **-through** option.

`-to to_list`

Specifies a list of names of clocks, ports, or pins that specify path endpoints. If a clock is given, all path endpoints related to that clock are implicitly included. This includes flip-flops in the transitive fanout of the clocks source pin or port, and ports that have output delay related to the clock.

`-rise_to rise_to_list`

Same as the **-to** option, but applies only to paths rising at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths captured by rising edge of the clock at clock source, taking into account any logical inversions along the clock path. You can use only one of **-to**, **-rise_to**, and **-fall_to**.

`-fall_to fall_to_list`

Same as the **-to** option, but applies only to paths falling at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of **-to**, **-rise_to**, and **-fall_to**.

DESCRIPTION

Groups a set of paths or endpoints for cost function calculations. The delay cost function is the sum of all groups (weight * violation), where violation is the amount for which setup was violated for all paths within the group. If there is no violation within a group, its cost is zero. Groups enable you to specify a set of paths to optimize even though there might be larger violations in another group. When endpoints are specified, all paths leading to those endpoints are grouped.

If a clock is specified in *from_list* or *to_list*, all endpoints related to that clock are included in the group. **create_clock** automatically creates a group for a new clock with a weight of 1.0 and named the same as the clock name.

The weight of the default group is 1.0. If *weight_value* is not specified for a new group, the default is 1.0.

To undo **group_path**, use **reset_design** or **group_path -default**.

To list the path groups that are defined, use **report_path_group**. To show the maximum delay cost for each path group, use **report_constraint**.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example groups all endpoints clocked by CLK1A or CLK1B into a new group "group1" with weight = 2.0.

```
prompt> group_path -name "group1" -weight 2.0 -to {CLK1A CLK1B}
```

The following example adds OUT1 and ff34/D to the existing path group named "ADDR".

```
prompt> group_path -name "ADDR" -to {OUT1 ff34/D}
```

The following example removes OUT1 and CLK2 from existing groups and places them in the default group.

```
prompt> group_path -default -to {OUT1 CLK2}
```

This command adds all paths that first pass through either u1/Z or u2/Z then pass through u5/Z or u6/Z into the group named "blue".

```
prompt> group_path -name blue -through {u1/Z u2/Z} -through {u5/Z u6/Z}
```

The following example groups all paths from inputs I1 and I2 to outputs O5 and O7 into a group named "serious" with weight 10.0.

```
prompt> group_path -name serious -weight 10.0 -from {I1 I2} -to {O5 O7}
```

The following example groups all endpoints clocked by CLK into a group with critical range of 0.5.

```
prompt> group_path -name CLK -critical_range 0.5 -to CLK
```

The following example uses a large *range_value* to optimize all paths. For example, assume that the worst violation is expected to be 100 units. Setting a *range_value* larger than the worst violation causes **compile** to speed up all paths, as long as the transformations do not increase the worst violation (that is, make the worse path slower) for that group. In this example, the *range_value* is set higher than the worst expected violation to ensure that all violating paths are considered during optimization.

```
prompt> group_path -name CLK -critical_range 1000.0 -to CLK
```

SEE ALSO

```
create_clock(2)
current_design(2)
report_constraint(2)
report_path_group(2)
reset_design(2)
set_input_delay(2)
set_output_delay(2)
set_critical_range(2)
```

gui_clear_filter_errors

Remove all filtering from the error list.

SYNTAX

string **gui_clear_filter_errors**

DESCRIPTION

Remove all filtering and restore the error set to one set by the last **gui_set_current_errors**.

EXAMPLES

The following example removes all filters:

```
gui_clear_filter_errors
```

SEE ALSO

```
gui_error_browser(2)  
gui_set_current_errors(2)  
gui_filter_errors(2)
```


gui_clear_selected_errors

Clears selection in the Error Browser.

SYNTAX

string **gui_clear_selected_errors**

ARGUMENTS

DESCRIPTION

Clears selected errors in the Error Browser.

EXAMPLES

The following example clears selected errors:

```
gui_clear_selected_errors
```

SEE ALSO

```
gui_error_browser(2)  
gui_load_error_cel(2)  
gui_set_current_errors(2)  
gui_set_selected_errors(2)
```

gui_create_tk_palette_type

Create a user configurable main window palette type.

SYNTAX

string **gui_create_tk_palette_type**

```
{-type type_name}  
{-title palette_title}  
{-window_types window_type_list}  
{-create_command tcl_command}  
[-dock_edge left/right/top/bottom]
```

```
string      type_name  
string      palette_title  
string      window_type_list  
string      tcl_command
```

ARGUMENTS

-type *type_name*

Defines a name for the new palette type. If the specified name is not unique, an error occurs.

-title *palette_title*

Defines a title string to be used to describe palettes of this type. For example, context menus which controll palette visibility will use the title.

-window_types *window_types*

Specifies the types of main windows that will contain the new palette type. After a palette type is defined, any new main window of the types specified are created with a palette.

-create_command *tcl_command*

Specifies a command to be called each time a main window of a type specified with *window_types* is created. The command is responsible for populating the palette with Tk control widgets. The command is called with two arguments: the name of the Tk widget to be populated and the name of the new main window.

-dock_edge *left/right/top/bottom*

Specifies the default dock edge to use when palettes of this type are created. If palettes are repositioned using the user interface, the new positions will override the default dock position the next time the application is run.

DESCRIPTION

This command defines a type of main window palette which contains a Tk widget. An instance of each defined palette type will be automatically created each time a main window of any of the specified types is instantiated. During the palette's initialization, the user's tcl command will be called to provide an opportunity for defining Tk widgets within the new palette. Creating a palette type does not cause a

palette to be created. It only specifies the parameters to be used to construct palettes of that type when main windows are subsequently created.

SEE ALSO

`gui_hide_palette(2)`, `gui_show_palette(2)`.

gui_error_browser

Show or hide the Error Browser Dialog

SYNTAX

```
string gui_error_browser [-hide | -show | -toggle]
```

ARGUMENTS

-hide

Hides the Error Browser Dialog.

-show

Shows the Error Browser Dialog. If no option flag is provided, this is the default.

-toggle

Show the Error Browser Dialog if it is currently hidden, else hide it.

DESCRIPTION

The **gui_error_browser** command controls the visibility of the Error Browser Dialog. The Error Browser Dialog displays violations recorded in an error cel by an DRC engine. This command is valid only when there is a design cel open. If the Error Browser Dialog is shown and it was not previously shown, it will automatically load error cels of default names for the currently open design cel. If the Error Browser Dialog is shown and it was previously shown, it will show the error cels last loaded.

EXAMPLES

The following example creates and shows the Error Browser Dialog:

```
gui_error_browser -show
```

SEE ALSO

gui_filter_errors

Filter the current error set with the given filter.

SYNTAX

```
string gui_filter_errors -show | -hide -selected | group GroupList  
GroupList          String List
```

ARGUMENTS

-show | -hide

Mutually exclusive required option to specify whether the filtering should hide the records selected by the filter criteria or prune to the records selected by the filter criteria.

-selected | -group *GroupList*

Mutually exclusive required argument to specify whether filtering should be applied based on the selection state of error records or on the group string of the error record. **GroupList** specifies the group name list, to match against the error record group if group string match option is used.

DESCRIPTION

Applies the given filter to the current errors set by the last **gui_set_current_errors** call. You can either filter out errors matching the filter specification (use **-hide**) or prune the current errors to errors matching the filter specification (use **-show**). Filter is applied once at the time of command execution and does not set a filter mode. For example, if you hide selected errors, errors that are selected after the command executes will not become hidden.

A new filter operates on the currently visible errors in the error list and always removes errors from the list, never adds. In order to show errors that have been hidden by previous filters, first clear the filters with **gui_clear_filter_errors**, then apply new filters.

The group labels used to filter is the grouping that is not set as the current grouping. For example, if you have selected to group errors by types by setting **gui_set_error_browser_option -grouping type**, then the grouping you specify here will be interpreted as layer strings.

EXAMPLES

The following example removes selected errors from display:

```
gui_filter_errors -hide -selected
```

The following example removes unselected errors from display:

```
gui_filter_errors -show -selected
```

The following example removes errors with layer string "Met1" when errors are

grouped by error type:

```
gui_filter_errors -hide -group "Met1"
```

The following example only shows errors of type "Short" when errors are grouped by layer:

```
gui_filter_errors -show -group "Open"
```

SEE ALSO

```
gui_error_browser(2)  
gui_set_current_errors(2)  
gui_set_selected_errors(2)  
gui_clear_filter_errors(2)  
gui_set_error_browser_option(2)
```

gui_get_error_browser_option

Get Error Browser Dialog option values

SYNTAX

```
string gui_get_error_browser_option -grouping | -show_mode | -view_mode | -zoom_factor | -hide_fixed | -dim | -show_open_locator_nodes | -list_limit
```

ARGUMENTS

```
-grouping | -show_mode | -view_mode | -zoom_factor | -hide_fixed | -dim | -show_open_locator_nodes | -list_limit
```

Mutually exclusive required option to specify the Error Browser option value to query.

DESCRIPTION

This command gets the value of the requested option setting for the Error Browser. One option value can be queried at a time.

The **-grouping** option returns *type* or *layer* which indicates whether errors are grouped by types or layers.

The **-show_mode** option returns *all*, *selected*, or *none* which indicates whether all errors, selected errors, are no errors are shown in layout views.

The **-view_mode** option returns *zoom*, *pan*, or *off* which indicates whether the layout view zooms to, pans to, or does not change to the currently selected errors in the Error Browser.

The **-zoom_factor** option returns a float value between *0.1* and *10.0* and indicates the zoom factor used when the *view_mode* is *zoom*.

The **-hide_fixed** option returns *true* to mean "hidden" or *false* to mean "not hidden" and indicates whether fixed errors are hidden from display in the Error Browser and layout views.

The **-dim** option returns *true* to mean "dimmed" or *false* to mean "not dimmed" and indicates whether layout views are dimmed when errors are displayed.

The **-show_open_locator_nodes** returns *true* to mean "highlighted" or *false* to mean "not highlighted" and indicates whether net nodes are highlighted along with the open locator flylines for selected open locator errors.

The **-list_limit** returns the number of errors that are displayed at one time in the error list.

EXAMPLES

The following example gets the grouping option

```
gui_get_error_browser_option -grouping
```

The following example gets whether layout views are dimmed when errors are displayed

```
gui_get_error_browser_option -dim
```

SEE ALSO

```
gui_error_browser(2)  
gui_set_error_browser_option(2)  
gui_page_errors(2)
```


gui_get_errors

Return errors in a collection.

SYNTAX

```
string gui_get_errors [-current | -visible | -selected] [-error_cel ErrorCel | -  
error_cel_type CellType] [-id ErrObjIdList] [-fixed FixedState]  
ErrorCel           String  
CellType           String  
ErrObjIdList       Integer List
```

ARGUMENTS

`[-current | -visible | -selected]`

These mutually exclusive flags specify a creation of collection from errors currently displayed in the Error Browser. These options are mutually exclusive with the `-error_cel_type` plus `-id` option that allows creation of collections from error object IDs.

`-current` specifies that all errors in the current error set set by the last call to `gui_set_current_errors` (or the last click in the Error Browser tree view) are returned in the collection.

`-visible` specifies that errors currently visible in the Error Browser error list are returned. Any errors that are currently hidden due to filtering or the current fixed error visibility settings are omitted.

`-selected` specifies that selected errors be returned

`-error_cel_type CellType`

CellType specifies the error cel of errors to select: one of "Detail DRC", "DRC", "LVS", "Signoff DRC", "Unknown". The cel type is ignored if `-current`, `-visible` or `-selected` is given.

`-error_cel ErrorCel | -error_cel_type CellType`

These mutually exclusive options identifies an error cel. The error cel, along with the list of error IDs given with the `-id` option will uniquely identify error objects to return. The **-error_cel** option specifies with a collection of error cels and The **-error_cel_type** option specifies with the error cel type: One of "Detail DRC", "DRC", "LVS", "Signoff DRC", "Unknown". The cel specification is ignored if `-current`, `-visible` or `-selected` is given.

`-id ErrObjIdList`

ErrObjIdList is a Tcl list of error object IDs of the errors to return. The IDs along with an error cel specification will uniquely identify error objects to return. The id list is ignored if an error cel is not specified.

`-fixed FixedState`

The **-fixed** argument specifies the states, fixed or not fixed, to match. If given, only errors matching the given fixed state will be returned.

DESCRIPTION

Return a collection of error objects.

EXAMPLES

The following example sets the current errors to "Open Locator" type errors from the loaded error cel written by verify_lvs, then creates a collection of errors of these errors, then sets their "fixed" state to true:

```
gui_set_current_errors -error_cel_type LVS -group "Open Locator" set errs  
[gui_get_errors -current] gui_set_error_fixed -errors $errs -fixed true
```

The following example gets the selected errors and outputs a textual report on the errors to stdout:

```
set errs [gui_get_errors -selected] gui_report_errors -errors $errs
```

The following example gets the error with ID 209 from the loaded error cel written by Hercules, then queries the bounding boxes of these errors:

```
gui_get_error_attribute -bbox -errors [gui_get_errors -error_cel_type DRC -id 209]
```

SEE ALSO

```
gui_error_browser(2)  
gui_load_error_cel(2)  
gui_set_current_errors(2)  
gui_set_error_fixed(2)  
gui_filter_errors(2)  
gui_get_attribute(2)  
gui_set_error_browser_option(2)
```

gui_get_routes_between_objects

Return routes that connect two objects on the same net.

SYNTAX

```
string gui_get_routes_between_objects  
[-slct_targets slct_bus]  
[-slct_targets_operation operation]  
[-create_slct_buses]  
objects
```

Data Types

<i>slct_bus</i>	string
<i>operation</i>	string
<i>objects</i>	collection

ARGUMENTS

-slct_targets *slct_bus*
Specifies the name of a selection bus to store the result in. By default the result of this command is returned in a collection. If this option is specified selection bus *slct_bus* is used instead. *slct_bus* is also returned as result of the command.

-slct_targets_operation *operation*
Specifies which operation should be used when storing the result in selection bus *slct_bus*. This is only allowed if you specify the **-slct_targets** option. Legal operations are clear, add and remove.

-create_slct_buses
Specifies that a new selection bus is created and used to store the result in. Using this option is equivalent to first creating a selection bus using the **create_selection_bus** command and then providing the created selection bus to option **-slct_targets**.

objects
A collection of two objects. The objects can be either ports or leaf pins.

DESCRIPTION

If *objects* contains two objects of port or leaf pin on the same net and the two objects are physically connected by routes of the net, this command returns a collection of routes that connect these two objects. Otherwiase the command returns an empty collection.

EXAMPLES

The following example selects routes that connect two leaf pin *A0_12__reg/Q* *RES_I109/ADD_6/A1*.

```
prompt> change_selection [gui_get_routes_between_objects [get_flat_pins {A0_12__reg/  
Q RES_I109/ADD_6/A1}]]
```

SEE ALSO

gui_inspect_violations

Brings up specified DFT DRC violations in a new Violation Inspector window. This command is only available in XG Mode.

SYNTAX for XG Mode

```
gui_inspect_violations  
    [-type AType] AList
```

Data Types

<i>AType</i>	String
<i>AList</i>	List

ARGUMENTS

-type

This argument is used to specify the type of violation (e.g. D1) when inspecting multiple violations of same type. This is optional argument but which affects how AList is interpreted.

AList

When specified with *ntype*, this can be a list of one or more violation ids to inspect. Violation ids is the violation number shown in the first column of violation Browser. When *ntype* argument is not used, this specifies a single violation instance to be inspected in violation inspector. Violation instances are named in the format (similar to Tetramax) ViolationType-ViolationId e.g. D1-1.

DESCRIPTION

This command brings up specified DFT DRC violations in a new Violation Inspector window unless a Violation Inspector window has been marked for reuse. If no violation inspector window exists, a new violation inspector window is created a new top-level window. Subsequent windows are created in the active top-level window. New violation inspector window created is not marked reusable. This command is only available in XG Mode.

EXAMPLES

To inspect multiple violations (5, 9 & 13) of type D1, use:

```
gui_inspect_violations ntype D1 {5 9 13}
```

To inspect a single violation 4 type D2, use:

```
gui_inspect_violations ntype D2 4  
or  
gui_inspect_violations D2-4
```

SEE ALSO

`gui_wave_add_signal(2)`
`gui_violation_schematic_add_objects(2)`

gui_load_area_net_connection_vm

Loads the data for visual mode of net connection in an area.

SYNTAX

```
status  gui_load_area_net_connection_vm

        -area area

        [-type list_of_types]
```

Data Types

<i>area</i>	list
<i>list_of_types</i>	list

ARGUMENTS

-area area

Analyzes area for net connectivity.

-type list_of_types

Types of nets to show.
Possible types are power, ground, clock, signal, tie_high, and tie_low.
This option takes types as a list of strings.

DESCRIPTION

The command generates visual mode of net connections in a given area.
Net connection will be categorized to:

- local connection: nets have all their pins within the given area.

- global connection: nets have at least one pin outside of the given area, and one pin inside of given area.
- path-through connection: nets have all their pins outside of given area.

Nets in the visual mode are displayed based on layout view setting of net.

EXAMPLES

The following example will show clock and signal types of net connections in the specified area.

```
prompt> gui_load_area_net_connection_vm -area \  
{550.710 1840.430 1247.100 2150.640} -type {clock signal}
```

SEE ALSO

None.

gui_load_cell_density_mm

Loads the data for cell density map mode.

SYNTAX

```
status gui_load_cell_density_mm  
[-area area]
```

Data Types

area list

ARGUMENTS

```
-area area  
    Analyzes area for cell density. If area is not given, the default area is  
    whole design area.
```

DESCRIPTION

The command generates the map mode view of cell density in a given area. Each grid is colored based on the percentage of occupied area of cells in the grid.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example will show cell density in the specified area.

```
prompt> gui_load_cell_density_mm -area \  
{284.820 390.670 1202.790 954.120}
```

SEE ALSO

gui_load_cell_slack_vm

Loads the data for cell slack visual mode.

SYNTAX

```
status gui_load_cell_slack_vm

    [-num_bins count]

    [-bin_range range]

    [-lower_bound value]

    [-lower_bound_strict]

    [-upper_bound value]

    [-upper_bound_strict]

    [-cells cell_list]
```

Data Types

<i>count</i>	integer
<i>range</i>	float
<i>value</i>	float
<i>cell_list</i>	list

ARGUMENTS

-num_bins *count*

Number of bins to be created.

-bin_range *range*

Range of each bin that is to be created
when number of bins is not specified.

`-lower_bound value`

Lower bound value for the bins.

Values below this bound are placed in a separate bin.

`-lower_bound_strict`

Uses lower bound as an exact minimum limit on the bound.

`-upper_bound value`

Upper bound value for the bins.

Values above this bound are placed in a separate bin.

`-upper_bound_strict`

Uses upper bound as an exact maximum limit on the bound.

`-cells cell_list`

Cells to be used in the visual mode.

If this option is not specified, all the cells in the design are used by default.

DESCRIPTION

The command generates the visual mode view of cell slack.

Each cell is colored based on the color of the bin it falls into.

EXAMPLES

The following example will place all the cells in the design into 8 bins.

```
prompt> gui_load_cell_slack_vm -num_bins 8 \
```

-lower_bound 0

The following example will place all the currently selected cells into 8 bins.

```
prompt> gui_load_cell_slack_vm -num_bins 8 \  
-lower_bound 0 -cells [get_selection]
```

SEE ALSO

None.

gui_load_clock_delay_vm

Loads the data for clock latency/transition visual mode.

SYNTAX

```
status  gui_load_clock_delay_vm

        [-num_bins count]

        [-bin_range range]

        [-lower_bound value]

        [-lower_bound_strict]

        [-upper_bound value]

        [-upper_bound_strict]

        [-delay latency | transition]

        [-arrival max_early_rise | max_early_fall | max_late_rise |
max_late_fall | min_early_rise | min_early_fall |
min_late_rise | min_late_fall]

        -clock clock_name
```

Data Types

<i>count</i>	integer
<i>range</i>	float
<i>value</i>	float
<i>clock_name</i>	string

ARGUMENTS

`-num_bins` *count*

Number of bins to be created.

`-bin_range range`

Range of each bin that is to be created
when number of bins is not specified.

`-lower_bound value`

Lower bound value for the bins.
Values below this bound are placed in a separate bin.

`-lower_bound_strict`

Uses lower bound as an exact minimum limit on the bound.

`-upper_bound value`

Upper bound value for the bins.
Values above this bound are placed in a separate bin.

`-upper_bound_strict`

Uses upper bound as an exact maximum limit on the bound.

`-delay latency | transition`

Specifies whether to use latency or transition in creating
the visual mode.

`-arrival max_early_rise | max_early_fall | max_late_rise |`

`max_late_fall | min_early_rise | min_early_fall | min_late_rise`
`| min_late_fall"`
Specifies type of arrival.

`-clock clock_name`

Clock for which the visual mode is to be created.

DESCRIPTION

The command generates the visual mode view for a clock based on clock net delay type.

Each clock sink pin is colored based on the color of the bin it falls into.

EXAMPLES

The following example will place all the clock sink pins for the given clock into 8 bins based on clock latency.

```
prompt> gui_load_clock_delay_vm -num_bins 8 \  
-lower_bound 0 -delay latency -clock pclk \  
-arrival max_early_rise
```

SEE ALSO

None.

gui_load_clock_tree_vm

Loads the data for clock tree visual mode.

SYNTAX

```
status  gui_load_clock_tree_vm

        [-exclude_cells]

        [-exclude_nets]

        -clock_trees level_type_list
```

Data Types

level_type_list list

ARGUMENTS

-exclude_cells

Exclude the cells.

-exclude_nets

Excludes the nets.

-clock_trees *level_type_list*

List of levels and types to be colored for each clock tree. It is a list containing alternate strings of clock trees and a list that specifies the levels and types for each clock tree. The list that specifies the levels and types has two items. The first item is a list of levels. Second item is a list of types. Valid types are sink, preexisting, inverter and buffer.

Ex: {sys_clk { {0 1} {sink buffer} } ...)

DESCRIPTION

Loads the data for clock tree visual mode which colors clock trees based on the levels and types (sink, preexisting, inverter, buffer).

EXAMPLES

The following example will color levels 0 and 1, sink and preexisting gates for clock tree "sys_clk" excluding the cells.

```
prompt> gui_load_clock_tree_vm -exclude_cells \  
  
-clock_trees { sys_clk { { 0 1 } { sink preexisting } } }
```

SEE ALSO

None.

gui_load_delta_delay_vm

Loads the data for delta delay visual mode.

SYNTAX

```
status gui_load_delta_delay_vm

    [-nets count]

    [-delay_type max | min]

    [-num_bins count]

    [-slack value]

    [-delay value]
```

Data Types

<i>count</i>	integer
<i>value</i>	float

ARGUMENTS

-nets *count*

Number of worst nets to use.

-delay_type max | min

Specifies the type of delay to be used.

-num_bins *count*

Number of bins to be created.

`-slack value`

Maximum slack to be used.

`-delay value`

Minimum delta delay to be used.

DESCRIPTION

Loads the data for delta delay visual mode.

Delta delay visual mode colors switching (setup or hold time) noise violations on victim nets, where crosstalk-induced delay is a function of switching noise height and the transition time.

EXAMPLES

The following example will place all the nets with maximum slack 0.00 and minimum delta delay of 0.00 into 4 bins.

```
prompt> gui_load_delta_delay_vm -nets 1000 \  
  
-delaytype max -num_bins 4 -slack 0.00 -delay 0.00
```

SEE ALSO

None.

gui_load_hierarchy_vm

Loads the data for hierarchy visual mode.

SYNTAX

status **gui_load_hierarchy_vm**

-clear

-level ncount

-cells cell_list

Data Types

level integer

cell_list list

ARGUMENTS

-clear

Removes the coloring from all hierarchy levels.

-level *level*

Colors all the cells at a particular hierarchy level.

-cells *cell_list*

Colors specific hierarchy cells.

If cells not provided, the default is cells in the entire design.

DESCRIPTION

Loads the data for hierarchy visual mode.

The hierarchy visual mode provides a high-level view of the placement quality of logic blocks and hierarchical cells in your physical design.

This visual mode can be used to color all the cells on a particular hierarchy level or just the specified hierarchical cells.

EXAMPLES

The following example clears the hierarchy visual mode.

```
prompt> gui_load_hierarchy_vm -clear
```

The following example colors all the cells at hierarchical level 2.

```
prompt> gui_load_hierarchy_vm -level 2
```

The following example colors all the cells in the selection.

```
prompt> gui_load_hierarchy_vm -cells [get_selection]
```

SEE ALSO

None.

gui_load_illegal_cell_placement_vm

Loads the data for illegal cell placement visual mode.

SYNTAX

status **gui_load_illegal_cell_placement_vm**

Data Types

ARGUMENTS

DESCRIPTION

The command generates the visual mode view of illegal cell placements.
Each illegal cell is colored based on the type of violation.

EXAMPLES

The following example will place all the llegal cells in the design
into different bins based on the type of placement violation.

```
prompt> gui_load_illegal_cell_placement_vm
```

SEE ALSO

None.

gui_load_imported_path_pins_vm

Loads the data for imported path pins visual mode.

SYNTAX

```
status  gui_load_imported_path_pins_vm

        [-timing_report file_name]

        [-timing_text report_text]

        [-clear]
```

Data Types

<i>file_name</i>	string
<i>report_text</i>	string

ARGUMENTS

`-timing_report file_name`

Generates the imported path pins visual mode form the timing report file.

`-timing_report report_text`

Generates imported path pins visual mode from the timing report text.

`-clear`

Removes coloring from all pins and flylines.

DESCRIPTION

Loads the data for imported path pins visual mode.

After importing timing path data from a timing report, the imported path pins visual mode can be used to examine the pin placement and connectivity in one or more timing paths.

This visual mode colors the pins and flylines with a different color for each imported timing path.

EXAMPLES

The following example will load the timing report file `"/remote/misc16/timing_report"` to generate the imported path pins visual mode.

```
prompt> gui_load_imported_path_pins_vm \  
  
-timing_report "/remote/misc16/timing_report"
```

SEE ALSO

None.

gui_load_net_capacitance_vm

Loads the data for net capacitance visual mode.

SYNTAX

status **gui_load_net_capacitance_vm**

[-num_bins count]

[-bin_range range]

[-lower_bound value]

[-lower_bound_strict]

[-upper_bound value]

[-upper_bound_strict]

[-nets net_list]

Data Types

count integer

range float

value float

net_list list

ARGUMENTS

-num_bins *count*

Number of bins to be created.

-bin_range *range*

Range of each bin that is to be created
when number of bins is not specified.

`-lower_bound value`

Lower bound value for the bins.

Values below this bound are placed in a separate bin.

`-lower_bound_strict`

Uses lower bound as an exact minimum limit on the bound.

`-upper_bound value`

Upper bound value for the bins.

Values above this bound are placed in a separate bin.

`-upper_bound_strict`

Uses upper bound as an exact maximum limit on the bound.

`-nets net_list`

Nets to be used in the visual mode.

If the nets are not provided, the default is all the nets in the design.

DESCRIPTION

The command generates the visual mode view of net capacitance.

Each net is colored based on the color of the bin it falls into and is drawn as a flyline.

EXAMPLES

The following example will place all the nets in the design into 8 bins.

```
prompt> gui_load_net_capacitance_vm -num_bins 8 \
```

-lower_bound 0

The following example will place all the nets returned by `get_nets` into 8 bins.

```
prompt> gui_load_net_capacitance_vm -num_bins 8 \  
-lower_bound 0 -nets [get_nets]
```

SEE ALSO

None.

gui_load_path_slack_vm

Loads the data for path slack visual mode.

SYNTAX

```
status  gui_load_path_slack_vm

        [-num_bins count]

        [-bin_range range]

        [-lower_bound value]

        [-lower_bound_strict]

        [-upper_bound value]

        [-upper_bound_strict]

        [-paths path_list]
```

Data Types

<i>count</i>	integer
<i>range</i>	float
<i>value</i>	float
<i>path_list</i>	list

ARGUMENTS

-num_bins *count*

Number of bins to be created.

-bin_range *range*

Range of each bin that is to be created
when number of bins is not specified.

`-lower_bound value`

Lower bound value for the bins.
Values below this bound are placed in a separate bin.

`-lower_bound_strict`

Uses lower bound as an exact minimum limit on the bound.

`-upper_bound value`

Upper bound value for the bins.
Values above this bound are placed in a separate bin.

`-upper_bound_strict`

Uses upper bound as an exact maximum limit on the bound.

`-paths path_list`

Paths to be used in the visual mode.
If paths are not provided, the default is all the paths in the design.

DESCRIPTION

The command generates the visual mode view of path slack.
Each path is colored based on the color of the bin it falls into.

EXAMPLES

The following example will place all the paths in the design
into 10 bins.

```
prompt> gui_load_path_slack_vm -num_bins 10 \
```

-lower_bound 0

The following example will place all the paths returned by `get_timing_paths` into 10 bins.

```
prompt> gui_load_path_slack_vm -num_bins 10 \  
-lower_bound 0 -paths [get_timing_paths]
```

SEE ALSO

None.

gui_load_pin_density_mm

Loads the data for pin density map mode.

SYNTAX

```
status gui_load_pin_density_mm  
[-area area]
```

Data Types

area list

ARGUMENTS

```
-area area  
        Areato be analyzed for pin density. If area option is not given, the default  
        area is whole design area.
```

DESCRIPTION

The command generates map mode view of pin density in a given area. The grids are colored based on the number of pins within the grid.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example will show pin density in the specified area.

```
prompt> gui_load_pin_density_mm -area \  
{284.820 390.670 1202.790 954.120}
```

SEE ALSO

gui_load_power_density_mm

Loads the data for power density map mode.

SYNTAX

```
status  gui_load_power_density_mm

        -mode leakage_power | internal_power | switching_power

        | dynamic_power | total_power
```

Data Types

ARGUMENTS

```
-mode leakage_power | internal_power | switching_power

      | dynamic_power | total_power"
      Specifies which power density map to be loaded.
```

DESCRIPTION

Loads the data for power density map mode.

Power density maps can be used to examine power density levels in the design.

This map divides the core area into a grid and colors each box to represent the power density level on a vertical plane and a horizontal plane. Each map color represents a range of power density values called a bin.

EXAMPLES

The following example will load leakage power density map.

```
prompt> gui_load_power_density_mm -mode
```

```
gui_load_power_density_mm
1042
```


SEE ALSO

None.

gui_load_relative_placement_vm

Loads the data for relative placement visual mode.

SYNTAX

status **gui_load_relative_placement_vm**

[-color group | hierarchy]

[-num_colors count]

[-cell_only]

Data Types

count integer

ARGUMENTS

-color group | hierarchy

Specifies whether to color objects by relative placement group or relative placement hierarchy level.

-num_colors *count*

Number of colors to use in the visual mode.

-cell_only

Colors only cells.

DESCRIPTION

Loads the data for relative placement visual mode.

gui_load_relative_placement_vm
1044

Relative placement visual mode helps to examine the placement of the instances (cells and other relative placement groups) in each group. This visual mode displays a colored map of relative placement groups with a different color either for each group (by default) or for each relative placement hierarchy level.

EXAMPLES

The following example will use 11 colors to color the cells based on relative placement groups.

```
prompt> gui_load_relative_placement_vm -color group \  
  
-num_colors 11 -cell_only
```

SEE ALSO

None.

gui_load_scan_chain_vm

Loads the data for scan chain visual mode.

SYNTAX

```
status  gui_load_scan_chain_vm  
        -chains scan_chain_names_list
```

Data Types

```
scan_chain_names_list    list
```

ARGUMENTS

```
-chains scan_chain_names_list
```

Loads the list of scan chains into the visual mode.

DESCRIPTION

Loads the data for scan chain visual mode.

This visual mode can be used to examine the placement of scan cells in the design once a scan is inserted. This visual mode colors each scan chain with a different color.

EXAMPLES

The following example will display the scan chains 1, 2, and 3.

```
prompt> gui_load_scan_chain_vm -chains {1 2 3}
```

SEE ALSO

None.

gui_load_static_noise_vm

Loads the data for static noise visual mode.

SYNTAX

```
status gui_load_static_noise_vm

    [-nets count]

    [-noise_type below_high | above_low | any]

    [-num_bins count]

    [-slack value]

    [-noise value]
```

Data Types

<i>count</i>	integer
<i>value</i>	float

ARGUMENTS

-nets *count*

Number of worst nets to use.

-noise_type *below_high | above_low | any*

Specifies the type of noise to be used.

-num_bins *count*

Number of bins to be created.

`-slack value`

Maximum slack to be used.

`-noise value`

Minimum noise to be used.

DESCRIPTION

Loads the data for static noise visual mode.

This visual mode can be used to examine a high-level view of the static noise levels for routed nets in the physical design. This visual mode displays a colored map of the accumulated noise levels for routed nets in the layout view. Various colors indicate different noise value ranges.

EXAMPLES

The following example will place all the nets with maximum slack 0.00 and minimum noise of 0.00 into 4 bins.

```
prompt> gui_load_static_noise_vm -nets 1000 \  
  
-noisetype below_high -num_bins 4 -slack 0 -noise 0
```

SEE ALSO

None.

gui_load_voltage_area_vm

Loads the data for voltage area visual mode.

SYNTAX

status **gui_load_voltage_area_vm**

Data Types

ARGUMENTS

DESCRIPTION

The command generates the visual mode view of voltage areas.
Each voltage area is colored based on the its type.

EXAMPLES

The following example will place all the voltage areas in the design into different bins based on their type.

```
prompt> gui_load_voltage_area_vm
```

SEE ALSO

None.

gui_page_errors

Display a new page of errors in the Error Browser

SYNTAX

```
string gui_page_errors [-page PageNumber | -next | -prev]  
PageNumber      Integer
```

ARGUMENTS

-page *PageNumber*

Loads the given page of errors in the Error Browser error list if it is available. The command is a no-op if the page is not available. Page numbering begins at 1.

-next

Loads the next page of errors in the Error Browser error list if it is available. The command is a no-op if there is no next page.

-prev

Loads the next page of errors in the Error Browser error list if it is available. The command is a no-op if there is no next page.

DESCRIPTION

The **gui_page_errors** command controls the current set of errors displayed in the Error Browser error list. The number of errors shown in the error list at a time is set as an Error Browser option with the **gui_set_error_browser_option** command. The set of errors shown from the current error group is controlled using paging.

EXAMPLES

The following example sets the number of errors shown at a time to 2000, then shows the second 2000 errors from the current error group.

```
gui_set_error_browser_option -list_limit 2000 gui_page_errors -page 2
```

SEE ALSO

```
gui_error_browser(2)  
gui_set_current_errors(2)  
gui_set_error_browser_option(2)
```

gui_report_errors

Output a textual report of errors.

SYNTAX

```
string gui_report_errors [-errors Errors] [-file FileName] [-include_hidden]  
Errors           String  
FileName         String
```

ARGUMENTS

-errors *Errors*

The **-errors** specifies the handle for a collection of errors to report on. This argument is optional. If omitted, a report is output for the errors currently loaded in the Error Browser error list.

[-file *FileName*]

FileName specifies the name of the file to write the report to. This argument is optional. If omitted, the report is output to stdout.

[-include_hidden]

-include_hidden specifies that errors that are currently hidden from the errors list due to filters or fixed errors being hidden should be included in the report. This option is ignored if errors are given with the **-errors** option.

DESCRIPTION

Output a tabular textual representation of errors. If the **-errors** argument is omitted and an output file is specified, performs the same operation as the Error Browser option menu command to save to file.

EXAMPLES

The following example saves a textual report of current errors to a file named myErrors.txt:

```
gui_report_errors -file "myErrors.txt"
```

The following example outputs a textual report of the fixed errors in the current error set to stdout: set errs [gui_get_errors -fixed true] gui_report_errors -errors \$errs

SEE ALSO

```
gui_error_browser(2)  
gui_load_error_cel(2)  
gui_get_errors(2)  
gui_set_current_errors(2)
```

gui_set_current_errors

Sets the current error cel type and the current error group in the Error Browser.

SYNTAX

```
string gui_set_current_errors [-error_cel_type CellType] [-group Group]  
CellType      String  
Group         String
```

ARGUMENTS

-error_cel_type *CellType*

CellType specifies the error cel to make current, one of "Detail DRC", "DRC", "LVS", "Signoff DRC", "Unknown". The cel type may be omitted.

-group *Group*

Group specifies the group name, either a layer name or error type name, to make current. The group name may be omitted.

DESCRIPTION

Sets the current error cel type and the current error group in the Error Browser. This is analogous to selecting error cel type and group in the Error Browser tree view. By default, there are no current errors. The current errors determine the errors that are displayed in the error list and in the layout view by default.

If the error cel type option is omitted, selects the design cell, the parent of all loaded error cels. The group option is valid only when there is a single error cel specified, and raises an error if there is no error cel specified. The group string is interpreted as error type if error grouping has been set to **type**, else the group string is interpreted as layer. If the group option is omitted, all groups in the specified error cel are made current.

EXAMPLES

The following example sets error grouping by error type, loads an error cel written by verify_lvs and sets the "Open Locator" type errors as the current errors:

```
gui_set_error_browser_option -grouping type gui_load_error_cel -error_cel_type LVS  
gui_set_current_errors -error_cel_type LVS -group "Open Locator"
```

The following example loads all error cels associated with the current design cell and sets the design cell as current, displaying all errors in all error cells in the error list and layout view.

```
gui_load_error_cel gui_set_current_errors
```

SEE ALSO

gui_error_browser(2)

```
gui_load_error_cel(2)  
gui_set_selected_errors(2)
```

gui_set_error_browser_option

Set Error Browser Dialog options

SYNTAX

```
string gui_set_error_browser_option [-grouping Group] [-show_mode ShowMode] [-  
view_mode ViewMode] [-zoom_factor ZoomFactor] [-hide_fixed DoHide] [-dim DoDim] [-  
show_open_locator_nodes DoShow] [-list_limit Limit]
```

Group	<i>String</i>
ShowMode	<i>String</i>
ViewMode	<i>String</i>
ZoomFactor	<i>float</i>
DoHide	<i>Boolean</i>
DoDim	<i>Boolean</i>
DoShow	<i>Boolean</i>
Limit	<i>Integer</i>

ARGUMENTS

-grouping *Group*

The **-grouping** option accepts argument values *type* or *layer* and sets whether errors are grouped by types or layers. This is analogous to selecting or deselecting *Show by Layer* option in the Error Browser *Options* menu. By default, errors are grouped by type by default.

-show_mode *ShowMode*

The **-show_mode** option accepts argument values *all*, *selected*, or *none* and sets whether all errors, selected errors, or no errors are shown in layout views. This is analogous to making a selection from the *Show* field in the Error Browser. The default is that all errors are shown.

-view_mode *ViewMode*

The **-view_mode** option accepts argument values *zoom*, *pan*, or *off* and sets the layout view to zoom to, pan to, or not change to the currently selected errors in the Error Browser. This is analogous to making a selection from the *Follow* field in the Error Browser. The default is to zoom to the selected error.

-zoom_factor *ZoomFactor*

The **-zoom_factor** option accepts float values between *0.1* and *10.0* and sets the zoom factor used when the *view_mode* is *zoom*. The zoom factor rounds to 1/10 precision. The default value is *1.0*.

-hide_fixed *DoHide*

The **-hide_fixed** option accepts Boolean values *1* or *true* to mean "hide" and *0* or *false* to mean "do not hide". The option governs whether fixed errors are hidden or shown in the Error Browser and layout views. This is analogous to selecting or deselecting *Hide Fixed Errors* option in the Error Browser *Options* menu. The default is that fixed errors are shown.

-dim *DoDim*

The **-dim** option accepts Boolean values *1* or *true* to mean "dim" and *0* or *false* to mean "do not dim". The option governs whether layout views are dimmed when

errors are displayed. This is analogous to checking of unchecking the *Dim* option in the Error Browser. The default is that layout views are not dimmed.

`-show_open_locator_nodes DoShow`

The **-show_open_locator_nodes** option accepts Boolean values *1* or *true* to mean "show" and *0* or *false* to mean "do not show" and governs whether net nodes are highlighted along with the open locator flylines for selected open locator errors. This is analogous to selecting or deselecting *Display Net Nodes for Selected Open Locators* option in the Error Browser *Options* menu. The default is to not show the net node highlights.

`-list_limit Limit`

The **-list_limit** option accepts an integer value between *500* and *10000*. It sets the number of errors that are displayed at a time in the Error Browser error list. The given value is rounded to the nearest allowable value. The default is *1000*.

DESCRIPTION

This command sets option setting for the Error Browser. Multiple option values can be set at a time.

EXAMPLES

The following example sets the grouping of errors by layers, to show all errors, and to pan to selected errors.

```
gui_set_error_browser_option -grouping layer -show_mode all -view_mode pan
```

The following example sets the option to dim layout views when errors are displayed and to hide fixed errors:

```
gui_set_error_browser_option -dim true -hide_fixed true
```

SEE ALSO

```
gui_error_browser(2)  
gui_get_error_browser_option(2)  
gui_page_errors(2)
```

gui_set_error_fixed

Set fixed state of one or more error records.

SYNTAX

```
string gui_set_error_fixed -errors Errors -fixed FixedState  
Errors                String
```

ARGUMENTS

-errors *Errors*

Errors specifies the handle for a collection of errors on which to set the fixed state.

-fixed *FixedState*

The **-fixed** argument specifies the states, fixed or not fixed, to set the state to. If 0 or false, the error state is set to "not fixed". If 1 or true, the error state is set to fixed.

DESCRIPTION

This command sets the fixed state of the given error objects to the given fixed state.

The fixed state of an error displayed in the Error Browser and an error can be hidden based on its fixed state. The state is an in-memory one with a lifetime of the error cel being loaded in the Error Browser. It is not written to the error cel. The state is a part of the textual report written to a file by gui_report_errors.

EXAMPLES

The following example sets errors with IDs 1108, 1304, and 1305 in the currently loaded error cel written by verify_lvs to "fixed" state:

```
gui_set_error_fixed -errors [gui_get_errors -error_cel_type LVS -id {1108 1304  
1305}] -fixed true
```

The following example creates a collection of errors then sets their fixed state to "not fixed":

```
set errs [gui_get_errors -error_cel_type DRC -id {58 59}] gui_set_error_fixed -  
errors $errs -fixed false
```

SEE ALSO

```
gui_error_browser(2)  
gui_load_error_cel(2)  
gui_get_errors(2)  
gui_set_error_browser_option(2)  
gui_report_errors(2)
```

```
gui_toggle_fixed_selected_errors(2)
```


gui_set_selected_errors

Change selection in the Error Browser to add or replace errors.

SYNTAX

```
string gui_set_selected_errors -add | -replace Errors  
Errors String
```

ARGUMENTS

```
-add | -replace Errors  
    Mutually exclusive required option to specify errors to select and to specify  
    whether the given errors are added to the current selection or replaces the  
    current selection.
```

DESCRIPTION

Set new selected errors or add new selected errors in the Error Browser. Errors are identified with a collection of error objects. Use `gui_get_errors` command to create the collection. If **-add** is given, then the command adds the given errors to currently selected errors. If **-replace** is given, then the command replaces current selection with given errors. The command fails if the given errors are not in the current errors set by `gui_set_current_errors` (or with a click in the Error Browser tree view). Returns 1 if successful, 0 otherwise.

EXAMPLES

The following example loads all error cels associated with the current design, sets the current errors to all error types in the current loaded error cel written by `verify_lvs`, then selects error records with IDs 1108, 1304, and 1305 in the error cel:

```
gui_load_error_cel gui_set_current_errors -error_cel_type LVS  
gui_set_selected_errors -replace [gui_get_errors -error_cel_type LVS -id {1108 1304  
1305}]
```

The following example sets the current errors to all errors in all loaded error cels, then creates a collection of error objects with IDs 58 and 59 in the current loaded Hercules DRC error cel, then adds the errors in the collection to the current selection.

```
gui_set_current_errors set errs [gui_get_errors -error_cel_type DRC -id {58 59}]  
gui_set_selected_errors -add $errs
```

SEE ALSO

```
gui_error_browser(2)  
gui_load_error_cel(2)  
gui_get_errors(2)  
gui_set_current_errors(2)
```

```
gui_clear_selected_errors(2)
```

gui_show_form

Displays gui support information for one or more commands.

SYNTAX

```
status gui_show_form pattern
```

Data Types

pattern string

ARGUMENTS

pattern
Specifies the command matching pattern

DESCRIPTION

The **gui_show_form** command is used to display GUI support information for one or more commands. When the *pattern* identifies a uniquely supported command, the corresponding command dialog box will be shown. The console will also display the possible menu context (menu item, window and task) that can be used to display the command dialog box. If the *pattern* identifies multiple commands, all matching information will be presented in the manpage viewer, allowing the user to review the man page or bring up the dialog box by clicking on the appropriate area. The *pattern* can also be one of the group names returned by `help -groups`, in which case all commands in the given group will be considered as the match.

If the uniquely identified context can not be shown, an appropriate error message will be displayed. For example, the `route_area` command dialog box can not be displayed, unless a design is open.

A GUI session should be active for this command to work.

The command returns successes if one or more commands are matched.

EXAMPLES

The following example shows the dialog box for the command `set_route_options`.

```
prompt> gui_show_form set_route_options
```

The following example shows information about all commands in command group `routing` in the manpage viewer.

```
prompt> gui_show_form routing
```

SEE ALSO

`help(2)`
`gui_show_man_page(2)`
`gui_start(2)`

gui_toggle_fixed_selected_errors

Toggle the fixed state of selected errors.

SYNTAX

```
string gui_toggle_fixed_selected_errors
```

DESCRIPTION

This command toggles the fixed state of the selected error records. If any of the selected errors are not fixed, then all selected errors will be set fixed. If all of the selected errors are fixed, then all selected errors will be set not fixed.

The fixed state of an error is displayed in the Error Browser and an error record can be hidden based on its fixed state. The state is an in-memory one with a lifetime of the error cel being loaded in the Error Browser. It is not written to the error cel. The state is a part of the textual report output by `gui_report_errors`.

EXAMPLES

The following example toggles the fixed state of the currently selected errors:

```
gui_toggle_fixed_selected_errors
```

SEE ALSO

```
gui_error_browser(2)  
gui_load_error_cel(2)  
gui_unload_error_cel(2)  
gui_set_error_browser_option(2)  
gui_report_errors(2)  
gui_set_error_fixed(2)
```

gui_unload_error_cel

Unload an error cel from the Error Browser.

SYNTAX

```
string gui_unload_error_cel [-error_cel ErrorCel | -error_cel_type CelType]  
ErrorCel      String  
CelType       String
```

ARGUMENTS

-error_cel *ErrorCel* | **-error_cel_type** *CelType*

These mutually exclusive options specify the error cel to unload. **-error_cel** option specifies with a collection of error cels and **-error_cel_type** option specifies the type of error cel to unload: One of "Detail DRC", "DRC", "LVS", "Signoff DRC", "Unknown". May be omitted.

DESCRIPTION

Unloads the given error cel from the Error Browser Dialog. If cel specification is omitted, unloads all error cels from the Error Browser.

EXAMPLES

The following example unloads the error cel generated by verify_lvs.

```
gui_load_error_cel -error_cel_type LVS
```

The following examples unloads all error cels.

```
gui_load_error_cel
```

SEE ALSO

```
gui_error_browser(2)  
gui_load_error_cel(2)  
gui_get_loaded_error_cels(2)
```

gui_violation_schematic_add_objects

Adds the specified signals to the waveform view of a specified violation inspector.

SYNTAX

```
gui_violation_schematic_add_objects  
    [-window AWnd]  
    [-clct]  
    AList
```

Data Types

<i>AWnd</i>	string
<i>AList</i>	list

ARGUMENTS

-window *AWnd*
Adds the *AList* objects to the Schematic window if *AWnd* is valid. If there is no such violation inspector window, the command exits with an error message. If no window is specified, the objects are added to the current active window.

-clct
Implies that the command is to treat *AList* as a collection of object handles. If not specified, *AList* is treated as a list of object names.

AList
Specifies a list of object names if no **-clct** option is specified. Otherwise, the list specifies a collection of object handles. In both cases, the list can consist of heterogeneous names.

DESCRIPTION

This command adds the specified objects to the schematic view of a specified violation inspector window and selects them.

EXAMPLES

The following example adds a cell object to the active ViolationInspector Schematic window:

```
prompt> gui_violation_schematic_add_objects gate_clock
```

The following example adds a cell named *gate_clk* and a port named *i_rd* to a specified ViolationInspector.2 window:

```
prompt> gui_violation_schematic_add_objects -window violationInspector.2 \  
      {gate_clk i_rd}
```

The following example adds selected objects to a specified ViolationInspector.3 window:

```
prompt> gui_violation_schematic_add_objects -window ViolationInspector.3 \  
        -clct [get_selection]
```

SEE ALSO

`gui_inspect_violations(2)`

gui_wave_add_signal

Adds the specified signals to a waveform view of a specified violation inspector.

SYNTAX

```
gui_wave_add_signal
    [-window AWnd]
    [-clct]
    AList
```

Data Types

<i>AWnd</i>	string
<i>AList</i>	list

ARGUMENTS

-window *AWnd*
Adds the signal to the WaveForm Viewer of the specified violation inspector window, if *AWnd* is valid. If there is no such violation inspector window, the command exits with an error message. If no window is specified, the signal is added to the waveform view of first ViolationInspector window created.

-clct
Implies that the command is to treat *AList* as a collection of object handles. If not, *AList* is treated as a list of object names.

AList
Specifies a list of object names if no **-clct** option is specified. Otherwise, the list specifies a collection of object handles. In both cases, the list can consist of heterogeneous names.

DESCRIPTION

This command adds specified objects to the waveform view of a specified violation inspector window. If you specify a cell, a group is created in the waveform view and all pins of the cell are added to this group as the list of signals. For a bus, all nets are added. Added objects are selected.

EXAMPLES

The following example adds a port object named *i_rd* to the first created ViolationInspector window with a waveform view:

```
prompt> gui_wave_add_signal i_rd
```

The following example adds selected objects to the given window ViolationInspector.3:

```
prompt> gui_wave_add_signal -window ViolationInspector.3 -clct [get_selection]
```

SEE ALSO

`gui_inspect_violations(2)`

gui_write_layout_image

Save a layout image in a file

SYNTAX

```
status gui_write_layout_image  
-output file_name  
[-window window_title]  
[-view_only]
```

Data Types

file_name string
window_title string

ARGUMENTS

-output *file_name*
Specifies the output file name, including a suffix (.jpg, .png, .xpm, or .bmp) that identifies format.

-window *window_title*
Identifies the layout window. The default is the most recently used Layout window.

-view_only
Writes only the layout view contents. The default is to write all of the layout window contents.

DESCRIPTION

The command `gui_write_layout_image` command writes the layout image, as seen on the screen, into a file. The image content can be the complete layout window (including layout window title, menus, menu bars and other visible toolbars docked in the layout window) or simply that of the view. The output file name must include one of supported formats as file suffix.

The supported formats are JPEG, PNG, XPM, and BMP.

This command works only in a GUI session.

EXAMPLES

The following example writes the latest layout image, excluding the layout window title, menus, and toolbars, in JPEG format to a file named `risc_core.jpg` that resides in the current directory.

```
prompt> gui_write_layout_image -view_only risc_core.jpg
```

SEE ALSO

`gui_start(2)`
`gui_stop(2)`

help

Displays quick help for one or more commands.

SYNTAX

```
string help  
[-verbose] [-groups] [pattern]
```

Data Types

pattern string

ARGUMENTS

-verbose
Displays options, for example "command -help".

-groups
Displays a list of command groups only.

pattern
Displays commands matching pattern.

DESCRIPTION

The **help** command is used to get quick help for one or more commands or procedures. This is not the same as **man** which displays reference manual pages for a command. There are many levels of help.

By typing the **help** command, commands are listed in paragraph format by command group. There are dedicated groups for Builtin commands and Procedures. Other groups are defined by the application.

List all of the commands in a group by typing the group name as the argument to **help**. Each command is followed by a one-line description of the command.

You can get a one-line description help for a single command by typing **help** followed by the command name. You can specify a wildcard pattern for the name. For example, all commands containing the string "alias". Finally, get syntax help for one or more commands by adding the **-verbose** option.

List only the command groups in the application by passing only the **-groups** option.

EXAMPLES

```
prompt> help
```

Procedures:

ls, sh,

Builtins:

alias, append, array, break, catch, cd, close, concat, continue
...

prompt> **help Procedures**

ls # List files
sh # Execute a shell command

prompt> **help a***

alias # Create a command which expands to words.
append # Builtin
array # Builtin

prompt> **help -verbose source**

source # Read a file and execute it as a script
 [-echo] (Echo all commands)
 [-verbose] (Display intermediate results)
 file_name (Script file to read)

SEE ALSO

man(2)

history

Displays or modifies the commands recorded in the history list.

SYNTAX

```
string history  
[-h] [-r] [args...]
```

ARGUMENTS

-h

Displays the history list without the leading numbers. You can use this for creating scripts from existing history. You can then source the script with the **source** command. You only can combine this option with a single numeric argument. Note that this option is a non-standard extension to Tcl.

-r

Reverses the order of output so that most recent history entries display first rather than the oldest entries first. You only can combine this option with a single numeric argument. Note that this option is a non-standard extension to Tcl.

DESCRIPTION

The **history** command performs one of several operations related to recently-executed commands recorded in a history list. Each of these recorded commands is referred to as an ``event''. The most commonly used forms of the command follow. You can combine each with either the **-h** or **-r** option, but not both.

- With no arguments, the **history** command returns a formatted string (intended for you to read) giving the event number and contents for each of the events in the history list.

- If a single, integer argument *count* is specified, only the most recent *count* events are returned. Note that this option is a non-standard extension to Tcl.

- Initially, 20 events are retained in the history list. You can change the length of the history list using the following:

history keep *count*

Tcl supports many additional forms of the **history** command. See the following ADVANCED TCL HISTORY section.

EXAMPLES

The following examples show the basic forms of the **history** command. First, you can limit the number of events shown using a single numeric argument.

```
prompt> history 3
      7  set base_name "my_file"
      8  set fname [format "%s.db" $base_name]
      9  history 3
```

Using the **-r** option creates the history listing in reverse order.

```
prompt> history -r 3
      9  history -r 3
      8  set fname [format "%s.db" $base_name]
      7  set base_name "my_file"
```

Using the **-h** option removes the leading numbers from each history line.

```
prompt> history -h 3
set base_name "my_file"
set fname [format "%s.db" $base_name]
history -h 3
```

ADVANCED TCL HISTORY

The **history** command performs one of several operations related to recently-executed commands recorded in a history list. Each of these recorded commands is referred to as an **event**. When specifying an event to the **history** command, the following forms may be used:

- A number: if positive, it refers to the event with that number (all events are numbered starting at 1). If the number is negative, it selects an event relative to the current event (**-1** refers to the previous event, **-2** to the one before that, and so on). Event **0** refers to the current event.

- A string: selects the most recent event that matches the string. An event is considered to match the string either if the string is the same as the first characters of the event, or if the string matches the event in the sense of the **string match** command.

The **history** command can take any of the following forms:

history Same as **history info**, described below.

history addcommand?exec? Adds the *command* argument to the history list as a new event. If **exec** is specified (or abbreviated) then the command is also executed and

its result is returned. If **exec** isn't specified then an empty string is returned as result.

history changenewValue *?event?* Replaces the value recorded for an event with *newValue*. *Event* specifies the event to replace, and defaults to the *current* event (not event **-1**). This command is intended for use in commands that implement new forms of history substitution and wish to replace the current event (which invokes the substitution) with the command created through substitution. The return value is an empty string.

history clear Erase the history list. The current keep limit is retained. The history event numbers are reset.

history event *?event?* Returns the value of the event given by *event*. *Event* defaults to **-1**.

history info*?count?* Returns a formatted string (intended for humans to read) giving the event number and contents for each of the events in the history list except the current event. If *count* is specified then only the most recent *count* events are returned.

history keep*?count?* This command may be used to change the size of the history list to *count* events. Initially, 20 events are retained in the history list. If *count* is not specified, the current keep limit is returned.

history nextid Returns the number of the next event to be recorded in the history list. It is useful for things like printing the event number in command-line prompts.

history redo*?event?* Re-executes the command indicated by *event* and return its result. *Event* defaults to **-1**. This command results in history revision: see below for details.

HISTORY REVISION

Pre-8.0 Tcl had a complex history revision mechanism. The current mechanism is more limited, and the old history operations **substitute** and **words** have been removed. (As a consolation, the **clear** operation was added.)

The history option **redo** results in much simpler ``history revision''. When this option is invoked then the most recent event is modified to eliminate the history command and replace it with the result of the history command. If you want to redo an event without modifying history, then use the **event** operation to retrieve some event, and the **add** operation to add it to history and execute it.

SEE ALSO

hookup_power_gating_ports

Connects the control pins of retention registers through the design hierarchy based on the specification. This command is supported only in non-UPF mode.

SYNTAX

```
status hookup_power_gating_ports  
[-type style]  
[-default_port_naming_style naming_style]  
[-port_naming_styles list_of_port_naming_styles]
```

Data Types

<i>style</i>	string
<i>naming_style</i>	string
<i>list_of_port_naming_styles</i>	list

ARGUMENTS

- type *style***
Specifies the type of retention registers to be hooked up.
- default_port_naming_style *naming_style***
Specifies a single string to use as the naming style. You can embed the value of the **power_gating_style** attribute into the name with the %s reserved string, and the value of the **power_gating_class** attribute into the name through the %d reserved string. You can use the **-default_port_naming_style** option as an alternative to the **-port_naming_styles** option.
- port_naming_styles *list_of_port_naming_styles***
Specifies the names to use for ports and pins created for the design hierarchy. The argument is a list and the position in the list corresponds to the value of the power pin class, which is the first value of the **power_gating_pin** attribute. For example, to specify the ports used for stitching pins with the **power_pin_class** attribute value 1 as *SLEEP* and with value 2 as *WAKE*, the list should be specified as {*SLEEP WAKE*}.

DESCRIPTION

Power Compiler hooks up the control pins of retention registers through the design hierarchy either to primary input ports of the design, or to output pins of the driving cells in the design. There are 2 steps in the stitching process: First, you specify the ports or pins on the design hierarchy to connect, along with the kind of control pins, and the style of retention registers. Second, stitch the control pins across the design hierarchy.

Use the **hookup_power_gating_ports** command for the second step. The command stitches the control pins of retention registers starting from the bottom and moving up. The stitching goes through hierarchical pins with compatible attributes until it reaches a port of the current design or a driver pin of a cell. If no ports or pins of compatible attributes were set using the **set_power_gating_signal** command for a

particular design hierarchy, the tool creates a new pin.

By default, the tool stitches the control pins of all retention registers in the design. On the hierarchical boundary, only the pins without the **power_gating_style** attribute are used to connect to the control pins. If the **-type** option is used, only the pins of retention registers with the specified type are stitched. The hierarchical pin must also have the same attribute to connect to those control pins.

If a port or pin with compatible attributes is not found, the tool creates a new port. The **-port_naming_styles** option specifies the names to use for a port created for a design hierarchy. The value is a list of up to 5 names, and the position in the list corresponds to the value of the power pin class. The first entry in the list is the value of the **power_gating_style** attribute.

If there is a conflict of names for the designs on the hierarchy boundary, a similar name is created. You can specify where to place the additional character in the name. For example, specifying **{REST%s SAVE%s}** creates **RESTa** and **SAVEa**, if there is a naming conflict. The **-default_port_naming_style** option is an alternative to **-port_naming_styles**. The **-default_port_naming_style** option can only accept one string. You can embed the value of **power_gating_style** into the name with the **%s** reserved string, and the value of **power_gating_class** into the name through the **%d** reserved string.

EXAMPLES

The following example hooks up the retention registers of the *clock_high* type. If there is no port or pin with the matching attribute on any design hierarchy, a new pin named **REST** is created for the control pins whose **power_pin_class** is *1*, and another new pin named **SAVE** is created for the control pins whose **power_pin_class** is *2*.

```
prompt>hookup_power_gating_ports -type clock_high \  
      -port_naming_styles {REST SAVE}
```

The following example creates the pin or port named **SAVE1** for the control pins of retention registers with a **power_pin_class** of *1*, and a pin or port named **SAVE2** for the control pins of retention registers with a **power_pin_class** of *2*, and so on.

```
prompt>hookup_power_gating_ports -type clock_low \  
      -default_port_naming_style SAVE%d
```

SEE ALSO

```
report_power_gating(2)  
set_power_gating_signal(2)  
set_power_gating_style(2)  
power_enable_power_gating(3)  
target_library(3)
```

hookup_retention_register

Hooks up the save and restore pins of the retention registers to the save signal and the restore signal. This command is supported only in UPF mode.

SYNTAX

```
status hookup_retention_register
```

ARUGMENTS

The **hookup_retention_register** command has no arguments.

DESCRIPTION

This command hooks up the retention register's save pin to the save signal that you specify using the the **set_retention_control** UPF command if the save pin is not hooked up. The command hooks up the retention register's restore pin to the restore signal that is you specify with the **set_retention_control** UPF command if the restore pin is not hooked up.

This command works on the entire design. It finds all of the UPF retention registers and hooks them up to the control signals.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following is an example of how to use the **hookup_retention_register** command:

```
prompt> hookup_retention_register
```

SEE ALSO

```
map_retention_cell(2)  
set_retention(2)  
set_retention_control(2)
```

identify_clock_gating

Identifies Power Compiler inserted clock-gating circuitry in a structural netlist.

SYNTAX

```
integer identify_clock_gating  
[-reset]  
[-reset_only cell_or_pin_list]  
[-gating_element gating_cell]  
[-gated_element gated_cell_or_pin_list]  
[-ungated_element ungated_cell_list]
```

Data Types

<i>cell_or_pin_list</i>	list
<i>gating_cell</i>	cell
<i>gated_cell_or_pin_list</i>	list
<i>ungated_cell_list</i>	list

ARGUMENTS

-reset
Resets all clock-gating attributes. By default, this option is off.

-reset_only *cell_or_pin_list*
Resets clock-gating attributes for the specified list of cells or pins. By default, this option is off.

-gating_element *gating_cell*
Marks the specified cell as a gating element. By default, this option is off.

-gated_element *gated_cell_or_pin_list*
Marks the specified cells or pins as gated elements. By default, this option is off.

-ungated_element *ungated_cell_list*
Marks the specified cells as cells that do not have clock gating. By default, this option is off.

DESCRIPTION

The **identify_clock_gating** command will be obsolete in a future release. Set the **power_cg_auto_identify** variable to true to activate automatic identification of clock-gating circuitry.

This command identifies the Power Compiler inserted clock-gating circuitry in the structural netlist. Identification refers to the process of detecting clock gate and the corresponding gated element association and setting different attributes on these objects. These attributes enable the later steps in the tool to work efficiently.

This command is used when the design has been written out, post-processed, and read back into the shell in the form of a structural netlist. The command is only run after technology mapping. There is no need to invoke this command if the netlist was in DDC or MilkyWay format and the clock-gating structure was not changed outside the tool.

It is considered best practice to set the correct clock-gating style by using the **set_clock_gating_style** command before invoking the **identify_clock_gating** command.

The most common usage is to invoke the command without any options. The **identify_clock_gating** command searches all of the clocks in the design for clock gates and gated cells. The tool then marks the clock-gating properties using attributes that are used by subsequent clock gating commands. Since only clocks are traversed in this case, it is important to run the **create_clock** command before running **identify_clock_gating** in this mode.

If test control pins are present on the clock gates, **identify_clock_gating** tries to reinstate the necessary attributes for the **insert_dft** command to work. The control signal is assumed to be of *scan_enable* type, unless otherwise marked as *test_mode*, by using the **set_clock_gating_style** command with the **-control_signal test_mode** options, prior to invoking **identify_clock_gating**. These attributes are applied to all the clock gates of the design that have been identified.

Use the **-gating_element** and **-gated_element** options together to mark the specified *gating_cell* as the corresponding clock gate of the *gated_elements*. If you specify only the **-gating_element** option, the command automatically detects the gated elements through netlist traversal.

Use the **report_clock_gating** command to verify the accuracy of the identification step. If there are any problems, rerun the **identify_clock_gating** command with different options to make repairs. This command does not support identification of user-inserted clock gates.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows the typical flow for using the **identify_clock_gating** command to identify the clock-gating circuitry:

```
prompt> read -f verilog mapped_design.v
prompt> current_design top
prompt> link
prompt> create_clock clk
prompt> identify_clock_gating
prompt> report_clock_gating
```

The following example shows how to apply clock gating attributes to the user-inserted *user_cg1* clock gate as the corresponding clock gate of the *reg1_reg* register bank:

```

prompt> read -f verilog mapped_design.v
prompt> current_design top
prompt> link
prompt> identify_clock_gating #identify tool inserted clock gates
prompt> identify_clock_gating -gated [get_cells reg1_reg*] \
    -gating user_cg1
prompt> report_clock_gating

```

The following example shows how to apply clock gating attributes to the user-inserted *user_cg1* clock gate and allow the tool identify the corresponding registers:

```

prompt> read -f verilog mapped_design.v
prompt> current_design top
prompt> identify_clock_gating #identify tool inserted clock gates
prompt> identify_clock_gating -gating user_cg1
prompt> report_clock_gating

```

The following example shows how to use the **identify_clock_gating** command with the **report_clock_gating** command to fix problems:

```

prompt> read -f verilog mapped_design.v
prompt> current_design top
prompt> link
prompt> create_clock -per 3 clk
prompt> identify_clock_gating
prompt> report_clock_gating -gated
    # Inspect if all clock gates are identified.
    # Assume cg1 is not identified as a clock gate.
prompt> identify_clock_gating -gating cg1
prompt> report_clock_gating -gated

```

SEE ALSO

```

create_clock(2)
report_clock_gating(2)
placer_gated_register_area_multiplier(3)

```

ignore_site_row

Indicates to ignore specified site rows or all site rows defined in the current design.

SYNTAX

```
string ignore_site_row  
[-row row_name]  
[-site site_name]
```

Data Types

```
row_name      string  
site_name     string
```

ARGUMENTS

```
-row row_name  
    Specifies one or more rows to be ignored.  
  
-site site_name  
    Specifies one or more sites to be ignored.
```

DESCRIPTION

The **ignore_site_row** command ignores site rows defined in the current design. If no row name or site name is specified, all site rows in current design will be ignored. If *site_name* is specified, all rows with the specified site name will be ignored, even if the row name is not specified.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command specifies to ignore all site rows in current design.

```
prompt> ignore_site_row
```

Each of the following commands specify to ignore all site rows in site CORE1.

```
prompt> ignore_site_row -site CORE1  
prompt> ignore_site_row -row row1 -site CORE1
```

The following command specifies to ignore only row1.

```
prompt> ignore_site_row -row row1
```


SEE ALSO

`create_site_row(2)`
`report_design(2)`

import_designs

Imports one or more design files in Verilog or .ddc format.

SYNTAX

```
status import_designs  
-format verilog | ddc  
[-top design_name]  
[-cell cell_filename]  
[-rp_constraint tcl_script_file]  
file_list
```

Data Types

<i>design_name</i>	string
<i>cell_filename</i>	string
<i>tcl_script_file</i>	string
<i>file_list</i>	list

ARGUMENTS

-format verilog | ddc
Specifies the format of a design being loaded into memory. This is a required argument.
The **verilog** format is an IEEE Standard Verilog format.
The **ddc** format is a Synopsys internal database format.

-top *design_name*
Specifies the top level module name of the design. The top level module is the module that is not instantiated by any other modules. In general, there is only one top module in a design and the **import_designs** command identifies the top module automatically. If the design has multiple top level modules, you must use this option to specify one.
When importing a .ddc design file, this option specifies the current design name. By default, the current design is automatically set to the first design that is read.

-cell *cell_filename*
Specifies the cell name. The command saves the design in the database using the specified cell name. If you do not specify this option when importing design files in verilog format, the tool uses the name of the top module as the cell name. If you do not specify this option when importing a design file in .ddc format, the tool uses the name of the current design as the cell name.

-rp_constraint *tcl_script_file*
Specifies a Tcl script file to load the relative placement constraints when considering the requirements of the physical datapath flow.

file_list
Specifies a list of files to be imported. When specifying more than one file, enclose them in braces {}.

DESCRIPTION

This command imports the designs specified in the **file_list** argument. It is a wrapper of a numbers commands to perform the loading of design files, saving the design in the Milkyway databae and also uniquify the design. The embedded commands in this wrapper is slightly different for importing files in verilog format than importing files in .ddc format, as shown below.

The **import_designs** command executes the following when importing design files in verilog format.

```
read_verilog                /* design is saved at this step */
uniquify_fp_mw_cel
```

The **import_designs** command executes the following when importing design files in .ddc format.

```
read_file -format ddc
current_design
link
uniquify
save_mw_cel                /* design is saved at this step */
```

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example imports three Verilog design files, design1.v, design2.v, and design3.v. The top module name of the design is called design_top. The design will be saved with the cell name called design_top.

```
prompt> import_designs -format verilog -top design_top -cel design_top \
        {design1.v design2.v design3.v}
prompt> save_mw_cel
```

SEE ALSO

```
current_design(2)
link(2)
read_file(2)
read_verilog(2)
save_mw_cel(2)
uniquify(2)
uniquify_fp_mw_cel(2)
```

import_fp_black_boxes

Imports one or more black boxes from the hierarchy preservation information (logical view) into the physical cell.

SYNTAX

```
status import_fp_black_boxes
[-max_count num_black_box]
[-side_length length
 | -gate_count gate_count | [-utilization utilization]]
black_boxes
```

Data Types

<i>num_black_box</i>	int
<i>length</i>	float
<i>gate_count</i>	int
<i>utilization</i>	float
<i>black_boxes</i>	collection

ARGUMENTS

-max_count *num_black_box*
Specifies the maximum number of black boxes to be imported. The default value of **-max_count** is 100.

-side_length *length*
Specifies the dimension of each side of the black box in unit of microns. This is a size-related option. You cannot specify this option in conjunction with the **-gate_count**, **-utilization** options. The area of the resulting black box is side_length x side_length. The default value of **-side_length** is 300.

-gate_count *gate_count*
Specifies the number of equivalent gates in the black box. This is a size-related option. You may specify this option in conjunction with the **-utilization** option. The value specified must be a positive non-zero number. The default value of **-gate_count** is 0.

-utilization *utilization*
Specifies the utilization value for the black boxes. This is a size-related option. The value specified must be between 0.0 and 1.0 . If you specify the utilization value, you must also specify the **-gate_count** option. The default value of **-utilization** is 0.7.

black_boxes
Specifies a list of black boxes to be imported.

DESCRIPTION

This command imports one or more black boxes from the hierarchy preservation information (logical view) into the physical cell. During the loading of the verilog

design files, black boxes are flattened and needed to be restored to the physical cells to allow prototyping of these modules.

Use the size-related **-side_length**, **-gate_count**, and **-utilization** options to define the size of the black boxes. These black boxes are soft macros. Use of all 3 size-related options simultaneously is prohibited.

When you specify the **-side_length** option, the utilization value is set to 1.0 internally.

When you specify the **-utilization** option, you must also specify a non-zero value for the **-gate_count** option. The default value of the **-gate_count** option is 0.

When You can specify the **-gate_count** option by itself, the default utilization value of 0.7 is used.

If you do not specify any of the size-related options, the default side length of 300 microns is used and the utilization value is set to 1.0 internally. The area of the resulting soft macro is 300x300 um**2. .

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example imports all feedthrough logical black boxes in a design:

```
prompt> set cells [get_cells -filter "is_logical_black_box==true&&\
        black_box_type==Feedthru"]
prompt> import_fp_black_box $cells
```

SEE ALSO

get_cells(2)
estimate_fp_black_boxes(2)

index_collection

Given a collection and an index, if the index is in range, extracts the object at that index and creates a new collection containing only that object. The base collection remains unchanged.

SYNTAX

```
collection index_collection  
collection1  
index
```

Data Types

<i>collection1</i>	collection
<i>index</i>	integer

ARGUMENTS

collection1
Specifies the collection to be searched.

index
Specifies the index in the collection. Allowed values are integers from 0 to **sizeof_collection** - 1 (minus 1).

DESCRIPTION

You can use the **index_collection** command to extract a single object from a collection. The result is a new collection containing that object.

The range of indices is from 0 to **sizeof_collection** - 1 (minus 1). If the specified index is outside that range, an error message is generated.

Commands that create a collection of objects do not impose a specific order on the collection, but they do generate the objects in the same predictable order each time. Applications that support the sorting of collections allow you to impose a specific order on a collection.

You can use the empty string for the *collection* argument. However, by definition, any index into the empty collection is invalid. So using **index_collection** with the empty collection always generates the empty collection as a result and generates an error message.

Note that not all collections can be indexed.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows the first object in a collection being extracted:

```
prompt> set c1 [get_cells {u1 u2}]  
{"u1", "u2"}  
  
prompt> query_objects [index_collection $c1 0]  
{"u1"}
```

SEE ALSO

```
collections(2)  
query_objects(2)  
sizeof_collection(2)
```

infer_power_domains

Infers power domains from the \$power specification.

SYNTAX

```
status infer_power_domains  
[-verbose]
```

ARGUMENTS

-verbose
Specifies to print equivalent **create_power_domain** commands as power domains are inferred.

DESCRIPTION

The **infer_power_domains** command translates \$power constructs specified in the RTL into equivalent power domains.

The \$power construct is used in the RTL to simulate the power down behavior of the circuit. Using \$power, you can specify the power domain name, signal controlling power rails of the domain, power down acknowledgment signal, and blocks in the power domain. A sense information on the power down and power acknowledgment signal can also be specified. Since sense information cannot be modeled during synthesis, the **infer_power_domains** command ignores sense information.

For more information on \$power, see the *HDL Compiler (Presto Verilog) Reference Manual* and the *HDL Compiler (Presto VHDL) Reference Manual*.

The **infer_power_domains** command can also be used to specify exception always-on paths. If a domain with primitive instances is created, and if the domain is always on (that is, if it does not have a power controlling signal associated with it), the **infer_power_domains** command marks input pins of all primitives with always-on attributes. Always-on synthesis ensures that the always-on paths are synthesized with always-on logic.

The **infer_power_domains** command traverses the design hierarchy in a top-down manner. As subdesigns are visited, it searches for \$power directives on each of the subdesigns. It then generates a unique name for the power domain, collects a power down signal, power acknowledge signal, and an instance list to generate power domains.

You can create power domains using the **create_power_domain** command and the **infer_power_domains** command.

EXAMPLES

The following example shows the creation of power domains:

```
prompt> infer_power_domain -verbose
```

```
infer_power_domains  
1090
```



```

#create_power_domain top <HardSpace\
#       -power_down -power_down_ctrl [get_nets Clk]
#create_power_domain A <HardSpace\
#       -power_down -power_down_ctrl [get_nets Clk] <HardSpace\
#       -object_list [get_cells <HardSpace\
#                   I_CONTROL <HardSpace\
#                   I_ALU <HardSpace\
#]
#create_power_domain A_0 <HardSpace\
#       -power_down -power_down_ctrl [get_nets I_PRGRM_CNT_TOP/Clk] <HardSpace\
#       -object_list [get_cells <HardSpace\
#                   I_PRGRM_CNT_TOP/I_PRGRM_FSM <HardSpace\
#]
1

```

The following example creates exception always-on paths in the design:

infer_power_domain -verbose

```

#create_power_domain A <HardSpace\
#       -object_list [get_cells <HardSpace\
#                   A_INST0 <HardSpace\
#                   A_INST1 <HardSpace\
#]
#set_attribute [get_pins AN2_INST0/A] always_on TRUE
#set_attribute [get_pins AN2_INST0/B] always_on TRUE
#create_power_domain A_0 <HardSpace\
#       -power_down -power_down_ctrl [get_nets A_INST0/PD] <HardSpace\
#       -object_list [get_cells <HardSpace\
#                   A_INST0/B_INST <HardSpace\
#]
#create_power_domain A_1 <HardSpace\
#       -power_down -power_down_ctrl [get_nets A_INST1/PD] <HardSpace\
#       -object_list [get_cells <HardSpace\
#                   A_INST1/B_INST <HardSpace\
#]
1

```

SEE ALSO

```

create_power_domain(2)
remove_power_domain(2)
report_power_domain(2)

```

initialize_floorplan

Produces a floorplan with chip boundary, core, rows, and wire tracks.

SYNTAX

```
status initialize_floorplan
[-control_type aspect_ratio | width_and_height | row_number | boundary]
[-core_aspect_ratio float]
[-core_utilization float]
[-row_core_ratio float]
[-core_width float]
[-core_height float]
[-number_rows int]
[-use_vertical_row]
[-no_double_back]
[-start_first_row]
[-flip_first_row]
[-left_io2core float]
[-right_io2core float]
[-bottom_io2core float]
[-top_io2core float]
[-keep_macro_place]
[-keep_std_cell_place]
[-min_pad_height]
[-pad_limit]
[-pin_snap]
[-keep_io_place]
```

ARGUMENTS

-control_type aspect_ratio | width_and_height | row_number | boundary
Specifies a control type for the size of the core area of a floorplan. The default control type is the **aspect_ratio**.
The **aspect_ratio** control type indicates that the core area of the floorplan in the current Milkyway CEL is determined by the ratio of the height divided by the width.
The **width_and_height** control type indicates that the core area is determined by the exact width and height.
The **row_number** control type indicates that the core area is specified in terms of the number of rows.
The **boundary** control type indicates that the core area is specified by the place and route boundary of the current Milkyway CEL.

-core_aspect_ratio float
Specifies a value for the aspect ratio which is the height divided by the width.
For example, an aspect ratio of 1.00 means that the height and the width of the core area are the same and the shape of the core is a square. An aspect ratio of 3.00 means that the height is three times the width, and an aspect ratio of 0.50 means that the width is twice the height.
Note that the aspect ratio you specified is only a target. The actual core area is created as close as possible to the specified aspect ratio.

-core_utilization float

Specifies an utilization number between 0 and 1.0. This number indicates the amount of the core area used for placement. This number is calculated as a ratio of the total cell area to the core area. The cell area includes all standard and macro cells.

For example, a core utilization of 0.8 means that 80 percent of the core area is used for cell placement and 20 percent is available for routing. This option is only valid when the **-control_type row_number | aspect_ratio** option is specified.

-row_core_ratio float

Specifies a number between 0 and 1.0 to indicate the amount of channel area between cell rows, reserved for routing relative to the core area. A smaller row to core ratio means that more space is left for the routing channels. A value of 1.0 means no routing channel space.

Note that this ratio should be equal to or greater than the core utilization value.

-core_width float

Specifies the width of the core area in user units. This option is valid only if you specify the **-control_type width_and_height** option.

-core_height float

Specifies the height of the core area in user units. This option is valid only if you specify the **-control_type width_and_height** option.

-number_rows int

Specifies the number of rows that comprise the core area. This option is valid only if you specify the **-control_type row_number** option.

-use_vertical_row

Indicates that cell rows are placed vertically in the core area. If this option not specified, cell rows are placed horizontally in the core area.

-no_double_back

Indicates that the core area does not contain pairs of cell rows with one row flipped in each pair. By default, cell rows in the core area are placed with pairing or flipping.

-start_first_row

Indicates that the pairing of cell rows starts at the bottom of a core area with horizontally placed cell rows or the left side of a core area with vertically placed cell rows . This option is valid only if you do not specify the **-no_double_back** option.

-flip_first_row

Indicates to flip the first row at the bottom of a core area with horizontally placed cell rows or the left side of a core area with vertically placed cell rows. This option is invalid only when the **-no_double_back** is specified.

-left_ioore float

Specifies the distance between the left side of the core area and the right side of the closest terminal or pad.

`-right_ioore float`
 Specifies the distance between the right side of the core area and the left side of the closest terminal or pad.

`-bottom_ioore float`
 Specifies the distance between the bottom side of the core area and the top side of the closest terminal or pad.

`-top_ioore float`
 Specifies the distance between the top side of the core area and the bottom side of the closest terminal or pad.

`-keep_macro_place`
 Indicates to keep the preplaced macro cells.

`-keep_std_cell_place`
 Indicates to keep the preplaced standard cells.

`-min_pad_height`
 Indicates to set the core area based on the minimum pad height.

`-pad_limit`
 Indicates to place pads without spaces in between.

`-pin_snap`
 Indicates to snap the center of terminals to the center of the closest wire track.
 Note that this option does not snap power terminals, ground terminals, or terminals with IO constraints.

`-keep_io_place`
 Indicates to keep the terminal's or pad's location, even if there are IO constraints specified previously using the **`read_io_constraints`** command.

DESCRIPTION

The `initialize_floorplan` command performs floorplanning on the current Milkyway CEL. Before executing this command, you must open a physical Milkyway CEL by using the **`open_mw_cel`** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example runs the `initialize_floorplan` command with the default settings:

```
prompt> open_mw_cel top -lib design
{"top"}
```

```
prompt> initialize_floorplan  
1
```

SEE ALSO

```
read_io_constraints(2)
```

initialize_rectilinear_block

Creates L-, T-, U-, and cross-shaped floorplans for rectilinear blocks.

SYNTAX

```
status initialize_rectilinear_block
[-shape L | T | U | X]
[-control_type ratio | length]
[-
core_side_dim {side_a_dim side_b_dim side_c_dim side_d_dim [side_e_dim side_f_dim]}}]
[-use_current_boundary]
[-row_core_ratio ratio_type]
[-core_utilization ratio_type]
[-orientation N | W | S | E]
[-use_vertical_row]
[-no_double_back]
[-start_first_row]
[-flip_first_row]
[-left_io2core distance]
[-right_io2core distance]
[-top_io2core distance]
[-bottom_io2core distance]
[-keep_macro_place]
[-keep_std_cell_place]
[-keep_io_place]
```

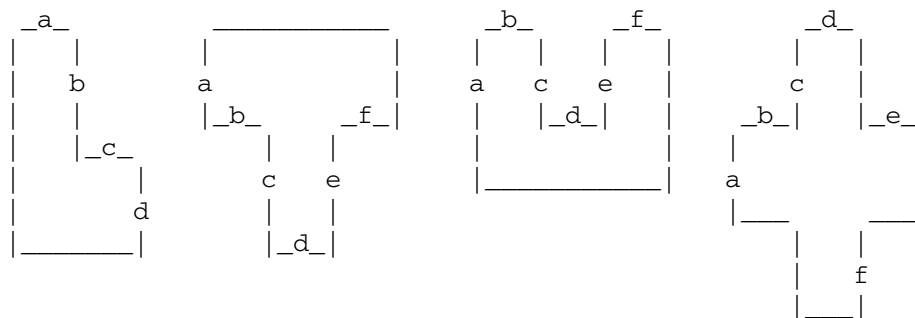
Data Types

```
side_a_dim      float >= 0.0
side_b_dim      float >= 0.0
side_c_dim      float >= 0.0
side_d_dim      float >= 0.0
ratio_type      0.0 <= float <= 1.0
```

ARGUMENTS

-shape L | T | U | X

Specifies a template shape used to determine the cell boundary and core shape of the rectilinear block. The following diagram shows the definition of the edges and the orientation of the L-, T-, U-, and X-shaped rectilinear blocks.



-control_type *ratio* | *length*
 Specifies how to interpret the list of dimensions provided for the **-core_side_dim** option.
 When the control type is *ratio*, each dimension in the list represents the relative proportion of the dimension of the edge to the sum of all the dimensions listed. For example, if the list of dimensions of an L-shaped block is {1 2 1 1}, the tool interprets the dimension of side a, c, or d is 20% of the sum of the dimensions listed, and the dimension of side b is 40% of the summation.
 When the control type is *length*, the dimensions in the list represent the actual physical dimensions for each edge of the polygon.

-core_side_dim {*side_a_dim side_b_dim side_c_dim side_d_dim [side_e_dim side_f_dim]*}

Specifies the dimensions of the edges of the rectilinear block. This list contains a maximum of 6 values, depending on the shape specified by the **-shape** option. The semantics of the values depends on the **-control_type** option specified. If you provide more values than needed to describe the specified shape, the extra values are ignored. If you do not provide all the values needed to describe the specified shape, the tool issues an error.

-use_current_boundary
 Uses the rectilinear boundary that is already defined. You should have defined the boundary using either the **initialize_rectilinear_block -shape** command or the **create_boundary -poly** command. This option is invalid if the existing boundary is rectangular.

-row_core_ratio *ratio_type*
 Specifies a number between 0 and 1.0 to indicate the amount of routing area between cell rows relative to the core area. A smaller row to core ratio means that more space is reserved for the routing channels. A value of 1.0 means no routing channel space.
 Note that this ratio must be equal to or greater than the core utilization value.

-core_utilization *ratio_type*
 Specifies an utilization number between 0 and 1.0. This number indicates the amount of the core area used for cell placement. This number is calculated as a ratio of the total cell area to the core area. The cell area is the total area of all standard and macro cells.
 For example, a core utilization of 0.8 means that 80 percent of the core area is used for cell placement and 20 percent is available for routing.

-orientation N | W | S | E
 Specifies one of the four possible orientations for the specified rectilinear shape. The tool repositions the block to the specified orientation by rotating it in a clockwise direction.

-use_vertical_row
 Indicates to cut standard cell rows vertically in the core area. If you do not specify this option, cell rows in the core area are cut in horizontal fashion.

-no_double_back
 Disables the use of the back-to-back style of row pairs of cell rows. By default, cell rows are cut in a back-to-back pairing format.

-start_first_row
 Indicates that the first row is part of the back-to-back pair. This option is valid only when you do not specify the **-no_double_back** option.

-flip_first_row
 Indicates that the first row is cut in a flipped orientation causing the back-to-back pairs to share power or ground rails in an alternate style. This option is valid only when you do not specify the **-no_double_back** option

-left_ioore distance
 Specifies the shortest distance between the IO pin and the left side of the core boundary.

-right_ioore distance
 Specifies the shortest distance between the IO pin and right side of the core boundary.

-top_ioore distance
 Specifies the shortest distance between the IO pin and the top side of the core boundary.

-bottom_ioore distance
 Specifies the shortest distance between the IO pin and the bottom side of the core boundary.

-keep_macro_place
 Indicates to keep the preplaced macro cells.

-keep_std_cell_place
 Indicates to keep the preplaced standard cells.

-keep_io_place
 Indicates to keep the current placement of the I/O pins. This option is valid only when you specify the **-use_current_boundary** option.

DESCRIPTION

The `initialize_rectilinear_block` command creates L-, T-, U- and cross-shaped floorplans for these rectilinear blocks. This command leaves all I/O pins at the origin. Optionally it can leave previously placed I/O pins in their current location.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates an L-shaped core with a core area computed with a `core_utilization` factor of 0.8. Each core bounding box edge will be at an offset of 10 microns from the core boundary. The core contains cell rows without routing channel space.

```
initialize_rectilinear_block
1098
```



```
prompt> initialize_rectilinear_block -shape L -control_type ratio \  
      -core_side_dim {50 50 50 50} \  
      -core_utilization 0.8 -row_core_ratio 1.0 \  
      -left_io2core 10 \  
      -top_io2core 10 \  
      -right_io2core 10 \  
      -bottom_io2core 10
```

SEE ALSO

```
initialize_floorplan(2)  
create_boundary(2)
```

insert_buffer

Inserts buffer cells on nets connected to specified pins.

SYNTAX

```
collection insert_buffer
[-new_net_names new_net_names]
[-new_cell_names new_cell_names]
[-no_of_cells number]
[-inverter_pair]
[-repeater_distance distance_between [-tolerance length]]
[-divide_load_by denominator [-ignore_pin_cap]]
[-port_half_distance]
[-location coordinate_list]
[-on_route]
[-orientation orientation_list]
object_list
buffer_lib_cell
```

Data Types

<i>new_net_names</i>	list
<i>new_cell_names</i>	list
<i>number</i>	integer
<i>distance_between</i>	float
<i>length</i>	float
<i>denominator</i>	float
<i>coordinate_list</i>	list
<i>orientation_list</i>	list
<i>object_list</i>	list
<i>buffer_lib_cell</i>	collection

ARGUMENTS

-new_net_names *new_net_names*

Specifies the names of the new nets that are inserted. You must specify one net name per buffer when inserting buffers. You must specify two net names per inverter pairs when inserting inverter pairs. If the specified net name is already present, the command adds a suffix of "_%d" or "_%d_%d". Optionally you can specify only the common base name and new net names will be generated by adding unique numeric suffixes to the common base name. The names specified can be any valid net names, but must be the leaf names. They must not be the hierarchical names. The names must not contain embedded hierarchical separators and must be unique in the current context as specified by the current instance. By default, the command uses the base name *eco_net*.

-new_cell_names *new_cell_names*

Specifies the names of the new cells that are inserted. You must specify one cell name per buffer when inserting buffers. You must specify two cell names per inverter pairs when inserting inverter pairs. If the specified cell name is already present, the command adds a suffix of "_%d" or "_%d_%d". Optionally you can specify only the common base name and the new cell names will be

generated by adding unique numeric suffixes to the common base name. These names can be any valid cell names, but must be the leaf names. They must not be the hierarchical names. The new names must not contain embedded hierarchical separators and must be unique in the current context, as specified by the current instance. By default, the command uses the common base name *eco_cell*.

-no_of_cells *number*

Specifies the number of buffer cells or inverter pairs to be introduced. The repeaters introduced will be connected back to back in series. By default, the command introduces single buffer cell or inverter pair.

The **-no_of_cells**, **-divide_load_by**, and **-repeater_distance** options are mutually exclusive.

-inverter_pair

Indicates that a pair of inverting library cells has to be inserted instead of a single non-inverting library cell. If you specify this option, you need to supply a library cell with an inverting output for the **-buffer_lib_cell** option. You can use this option when the library cell (buffer) specified has both inverting and non-inverting outputs.

-repeater_distance *distance_between*

Specifies the distance between each buffer to be introduced along the route. The value specified for *distance_between* should be in units of micron. This option always works on the entire physical route underlying the nets or pins specified. If you specify a net in the middle of the hierarchy, this option will find the driver of the net, possibly in another level of hierarchy, and buffer the entire set of logical nets associated with the physical route. Buffers are not inserted into ILMs.

The **-repeater_distance** option can only be used in conjunction with the **-on_route** option. The **-repeater_distance**, **-no_of_cells**, and **-divide_load_by** options are mutually exclusive.

-tolerance *length*

Specifies an incremental distance between each buffer to be introduced along a route. The value specified for *length* should be in units of micron. You can use this option only if you also use the **-repeater_distance** option. The *length* value is added to the *distance_between* value of the **-repeater_distance** option. This option provides a convenience way to adjust the value of the **-repeater_distance** incrementally per command.

-divide_load_by *denominator*

Enables the buffering of a net such that resulting nets after buffer insertion, have a load no more than the original load divided by the specified *denominator*. The valid values of *denominator* are any integer greater than or equal to **2**. This option always works on the entire physical route underlying the nets specified. If you specify a net in the middle of the hierarchy, this option will find the driver of the net, possibly in another level of hierarchy, and buffer the entire set of logical nets associated with the physical route. Buffers are not inserted into ILMs.

Use the **-divide_load_by** option only in conjunction with the **-on_route** option. The **-divide_load_by**, **-no_of_cells**, and **-repeater_distance** options are mutually exclusive.

-ignore_pin_cap

Considers only the net capacitance when buffering the net. The tool ignores the pin capacitance. You can use this option only if you also use the **-divide_load_by** option. If you do not use the **-ignore_pin_cap** option, the processing of the **-divide_load_by** option considers net capacitance and pin capacitance.

-port_half_distance

Indicates that some external routing is assumed while processing the **-repeater_distance** option. Assuming half of the **repeater_distance** value is the routing distance outside each top-level port, the tool places a buffer half the distance from the port on long nets. From then on, the full **repeater_distance** is used. You can use this option only if you also use the **-repeater_distance** option.

-location_coordinate_list

Specifies the lower left coordinates for the locations of the buffer or inverter pair cells, in the format {x1 y1 x2 y2 ...}. You must specify one location per buffer when inserting buffers. You must specify two locations per inverter pair when inserting inverter pairs. These locations can be any valid x-y coordinate pair inside the placement region of the core area, relative to the chip origin. Specify the coordinates in units of microns. By default, the tool derives locations for the buffers based on the pins specified for the **object_list** option.

-on_route

Places buffer or inverter cells along the route of the net. If the command is introducing multiple cells, then it spreads them along the path of the route that is common to the given **object_list**. This option is effective only on routed and extracted designs.

You can use the **-on_route**, **-repeater_distance**, and **-divide_load_by** options together to use the distance based or the load based buffer insertion along the route. This option also controls the recording of the modified nets for the **split_net** command. See **split_net** man page for more details.

-orientation orientation_list

Specifies the orientation of the buffer cells or inverter pairs. You must specify one orientation per buffer when inserting buffers. You must specify two orientations per inverter pair when inserting inverter pairs. These orientations can be any valid orientation allowed in the library, for example, N, W, S, E, FN, FW, FS, or FE. By default, the command uses the N orientation.

object_list

Specifies a list of nets, pins, or ports that must be buffered. The new buffer cells or inverter pairs are placed close to the specified pins or ports. If you specify a net, the tool connects buffer or inverter pairs such that a new buffer cell is the new load of the original net. If you specify pins, the tool groups all the specified pins based on the nets to which they are connected. When the grouped pins are load pins, the buffers are introduced such that the new buffer cells drive them. When the grouped pins are driver pins, the new buffer cells are connected such that they become the load of the specified driver pin.

`buffer_lib_cell`

Specifies the library cell object to be used as buffer. In this case, the object is either a named library cell or a library cell collection. If the library cell is a buffer cell, the number of instances of buffers introduced is equal to the number specified by the **-no_of_cells** option. If the library cell is an inverter, the number of instances of inverters introduced is twice the number specified by **-no_of_cells**.

If the library cell has both inverting and non-inverting outputs (that is, it can act both as a buffer or an inverter), the **-inverter_pair** option controls which output is used. If the library cell has multiple outputs, the command uses the first non-inverting or inverting output.

DESCRIPTION

This command adds buffers at one or more specified pins or ports. A library cell with a single input and multiple outputs is a buffer, as long as each output has the same or inverted logic function as the input.

Like all other netlist editing commands, for the **insert_buffer** command to succeed, all of its arguments must be valid. If any argument specified is invalid, the netlist remains unchanged. If the command succeeds, the result is a collection of the newly inserted cells. If the command fails, the result is an empty collection or an empty string.

By default, each newly created cell has a name beginning with the `eco_cell` string and ending with a unique numeric suffix. Each newly created net has a name beginning with the `eco_net` string and ending with a unique numeric suffix. To override the automatic name generation by the tool, use the **-new_net_names** and **-new_cell_names** options.

Note that the locations of the new cells specified by the **-location** option or the locations derived by the tool, are not legalized. If you use the **legalize_placement** command after using the **insert_buffer** command, the tool may move these new cells around to find a legal locations.

You can mimic buffer insertion using other commands such as **create_cell**, **create_net**, **disconnect_net**, and **connect_net**, but the **insert_buffer** command provides a more efficient and fail safe way to insert buffers.

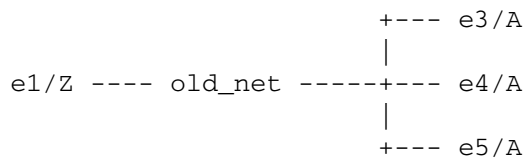
The **insert_buffer** command uses the following basic rules to check its arguments for validity:

- The pin or port must be in scope at or below the current instance. For a description of some special scoping rules, see the Buffering Inside Boundary Pins section.
- The pin or port must be connected to a net.
- Bidirectional pins cannot be buffered.
- The `lib_cell` cannot be sequential.
- The `lib_cell` must be a buffer, as previously defined.

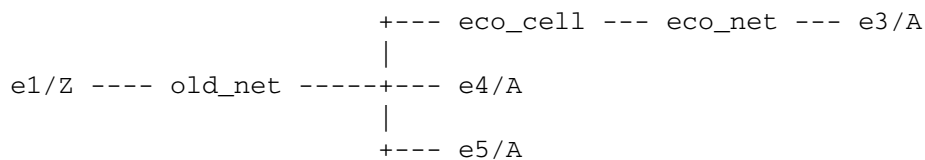
- The library cell must have an inverting output, when you use the **-inverter_pair** option. The command uses the first inverting output, inserting two cells, and preserving the logic of the path.
- The *number* argument of the **-no_of_cells** option must be a positive, non-zero number.

Inserting and Connecting the New Buffer

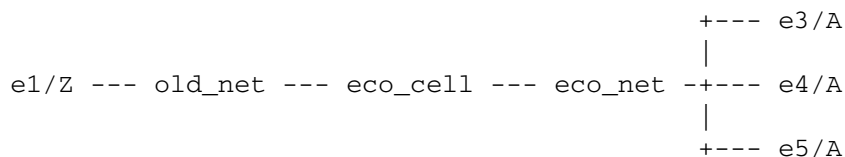
The **insert_buffer** command connects the new buffer or inverter pair according to the rules as explained with the following diagrams.



- If the specified pin is a load, the load is disconnected from its old net, and connected to the new net, as shown below.



- If the specified pin is a driver, all loads on that net are disconnected from the old net and connected to the new net, as shown below.



- If a list of load pins on the same net are specified, they are grouped, and the new net will drive them, as shown below.

```

          +--- e3/A
          |
e1/Z --- old_net +- eco_cell --- eco_net -+--- e4/A
                                   |
                                   +--- e5/A

```

- Buffering cannot be done if the net is multi-driven.

Buffering Inside Boundary Pins

When you specify insertion of a buffer at a pin on the boundary of a hierarchical block, the **insert_buffer** command inserts the buffer either inside or outside the hierarchical block, depending on the current hierarchical scope. To insert the buffer inside it, set the scope to that block by using the **current_instance** command, then execute the **insert_buffer** command. The buffer is inserted within the block to which **current_instance** is set.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example specifies a *lib_cell* that is not a buffer. The command fails and an error message is generated.

```

prompt> insert_buffer e1/Z class/AN2P
Information: Buffering net old_net. (HFS-701)
Error: library cell 'AN2P' is not a buffer or an inverter. (NLE-010)

```

The following example specifies a buffer for the *lib_cell*.

```

prompt> insert_buffer e1/Z class/B1I
Information: Buffering net old_net. (HFS-701)
Creating net 'eco_net' in design 'top'.
Creating cell 'eco_cell' in design 'top'.
Disconnecting net 'old_net' from pin 'e3/A'.
Disconnecting net 'old_net' from pin 'e4/A'.
Disconnecting net 'old_net' from pin 'e5/A'.
Connecting net 'eco_net' to pin 'e3/A'.
Connecting net 'eco_net' to pin 'e4/A'.
Connecting net 'eco_net' to pin 'e5/A'.

```

```

Connecting net 'eco_net' to pin 'eco_cell/Z'.
Connecting net 'old_net' to pin 'eco_cell/I'.
Warning: Setting default orientation 'North' on the object 'eco_cell'. (PSYN-294)
{eco_cell}

```

The following example specifies an inverter from the *lib_cell*. The command succeeds and creates the new cells *eco_cell*, *eco_cell_1*, *eco_cell_2* and the new nets *eco_net*, *eco_net_1*, *eco_net_2*.

```

prompt> insert_buffer e3/A class/IV
Information: Buffering net old_net. (HFS-701)
Creating net 'eco_net' in design 'top'.
Creating net 'eco_net_1' in design 'top'.
Creating cell 'eco_cell' in design 'top'.
Creating cell 'eco_cell_1' in design 'top'.
Disconnecting net 'old_net' from pin 'e3/A'.
Connecting net 'eco_net' to pin 'e3/A'.
Connecting net 'eco_net' to pin 'eco_cell/Z'.
Connecting net 'eco_net_1' to pin 'eco_cell/A'.
Connecting net 'eco_net_1' to pin 'eco_cell_1/Z'.
Connecting net 'old_net' to pin 'eco_cell_1/A'.
Warning: Setting default orientation 'North' on the object 'eco_cell'. (PSYN-294)
Warning: Setting default orientation 'North' on the object 'eco_cell_1'. (PSYN-294)
{eco_cell eco_cell_1}

```

```

prompt> report_cell -connections e3
*****
Report : cell
        -connections
*****

```

```

Connections for cell 'e3':
Reference:      B1I
Library:       class

Input Pins      Net
-----
A               eco_net

Output Pins     Net
-----
Z               out2

```

```

1
prompt> report_net -connections eco_net
*****
Report : net
        -connections
*****

```

```

Connections for net 'eco_net':

Driver Pins      Type
-----

```


eco_cell/Z

Output Pin (IV)

Load Pins

Type

e3/A

Input Pin (B1I)

1

SEE ALSO

current_instance(2)
get_pins(2)
legalize_placement(2)
remove_buffer(2)
report_cell(2)
report_net(2)
size_cell(2)

insert_diode

Inserts diodes into the design to fix antenna violations.

SYNTAX

```
status insert_diode
[-nets collection_of_nets]
[-antenna_check_engine internal | hercules]
[-internal_check_option all | top_layer_only]
[-no_auto_cell_selection]
[-diode_cells collection_of_diode_cells]
[-prefix prefix_name]
[-use_hierarchical_diode_instance_name]
[-same_row]
[[[-dont_freeze_existing_placement]
  | [-routing skip | when_all_violations_are_gone | always]]]
[-
signal_route_options ignore_lower_layers | include_lower_layers | include_all_lower
_layers | advanced]
[-max_ratio max_ratio_number]
```

Data Types

<i>collection_of_nets</i>	collection
<i>collection_of_diode_cells</i>	collection
<i>prefix_name</i>	string
<i>max_ratio_number</i>	integer

ARGUMENTS

-nets *collection_of_nets*
Specifies a collection of nets.

-antenna_check_engine internal | hercules
Specifies either the internal antenna checker and/or Hercules. The default is **internal**.

-internal_check_option all | top_layer_only
Specifies to report either all antenna rule violations or to report only the top-layer antenna violations. The default is **all**.

-no_auto_cell_selection
Specifies not to perform searches of the design library and its reference libraries for all diode cells. If you use this option, you must also use the **-diode_cells** option to specify one or more diode cells. The default is for the tool to search design library and reference libraries for all diode cells.

-diode_cells *collection_of_diode_cells*
Specifies one or more diode cells. If you specify more than one diode port and if more than one diode port can be used to fix the antenna violation, the tool inserts the smallest diode port to fix the problem.

`-prefix prefix_name`

Specifies a common prefix for all of the diode cell instances to insert. Use a unique prefix such as `ant_fix_0428_` so that the diode instances can be easily identified later.

`-use_hierarchical_diode_instance_name`

Specifies whether or not to use the hierarchical diode instance name.

`-same_row`

Specifies whether or not to place diodes in the same row as violation port. The tool will try to place the diode in the same row as the cell where the violation occurs whenever possible. This may not be possible due to existing placed cells. By default, the tool decides best location to place the diode. This option will automatically turn on **`-dont_freeze_existing_placement`** option. The **`-same_row`** **`-routing`** options are mutually exclusive.

`-dont_freeze_existing_placement`

Specifies not to freeze the placement of existing cell instances. The **`-dont_freeze_existing_placement`** **`-routing`** options are mutually exclusive. By default, the tool freezes the placement of all existing cell instances before inserting any diode cell instances.

You can also choose to let the command complete the routing by using the **`-routing`** option. The "complete routing" option is always set to **No** in this case, because it would not benefit you to run the engineering change order (ECO) detail router if the existing placement is already altered. You might need to re-run the global router and track assignment.

`-routing skip | when_all_violations_are_gone | always`

Specifies the extent of the routing process. Using **`skip`** stops routing after placing the diode cell instances. Using **`when_all_violations_are_gone`** invokes the detail router only when diodes are added to all of the reported violations. Using **`always`** invokes the detail router in ECO mode to route diode ports after placing diode cell instances.

If the command failed to add diodes to all of the violations (for example, if no diode is large enough to fix it) and you have invoked the ECO detail router, runtime might now be impacted because the router continues to attempt to fix the remaining antenna violations under tight ECO constraints. You can obtain a better result by using the normal (non-ECO) *search & repair*. Using **`always`** allows you to decide what to do when diodes cannot be added for all violations.

The **`-dont_freeze_existing_placement`** **`-routing`** options are mutually exclusive. The default is **`when_all_violations_are_gone`**.

`-signal_route_options ignore_lower_layers | include_lower_layers | include_all_lower_layers | advanced`

Specifies the HPO Signal route options for checking a charge-collecting antenna. By default, the tool does not check or fix charge-collecting antennas. Using **`ignore_lower_layers`** calculates the maximum ratio for checking antennas using the top mask layer only as the wiring area. If this is specified, the tool calculates the maximum ratio for checking antennas by using the area from the input port to the closest top layer as the wiring area.

Using **`include_all_lower_layers`** calculates the maximum ratio for checking antennas by using the area from the input port to the farthest top layer as the wiring area.

Using **advanced**, causes the router to use the antenna rule defined by the **define_antenna_layer_rule** and **define_antenna_rule** IC Compiler commands, instead of the ratio specified by using the **-max_ratio** option or the ratio set by using the **set_parameter** command.

-max_ratio *max_ratio_number*

Specifies the maximum allowable ratio of wiring area and gate area for checking a charge-collecting antenna.

DESCRIPTION

This command inserts diodes into the design to fix antenna violations. First, the tool runs the internal antenna checker or uses Hercules output to identify antenna violations. Next, a diode cell (or multiple diodes when the antenna ratio requires the protection of more than one diode) is placed for each violation. You can freeze all of the existing cells' placement and routing so the diode cells are placed only in existing empty spaces to minimize the impact on existing placement and routing. You can also let the **insert_diode** command complete the routing of the diode cells when cell placement is already frozen. Whenever possible, the tool ties diode cells to the existing wires without ripping and rerouting the entire net.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following is an example of the **insert_diode** command.

```
prompt> insert_diode -signal_route_options \  
include_lower_layers -max 1000
```

The following is an example of the **insert_diode** command.

```
prompt> insert_diode -antenna_check_engine hercules \  
-signal_route_options include_all_lower_layers -max_ratio 200
```

The following is an example of the **insert_diode** command.

```
prompt> insert_diode -no_auto_cell_selection \  
-signal_route_options ignore_lower_layers -max_ratio 100 \  
-diode_cells pv0a
```

The following is an example of the **insert_diode** command.

```
prompt> insert_diode \  
-signal_route_options include_all_lower_layers \  
-max 150 -routing skip -prefix diode
```

```
insert_diode  
1110
```

SEE ALSO

`insert_metal_filler(2)`
`insert_ng_filler(2)`
`insert_pad_filler(2)`
`insert_stdcell_filler(2)`
`insert_well_filler(2)`

insert_isolation_cell

Inserts isolation cells on the specified nets, pins or ports. Isolation cell is a general term that applies to isolation (ISO) cells and enabled level-shifter (ELS) cells.

SYNTAX

```
status insert_isolation_cell
[-force]
[-verbose]
-enable enable_signal
-object_list objects
-reference lib_cell_name
```

Data Types

objects collection

ARGUMENTS

- force
Forces insertion of isolation cells on nets with the dont_touch or always_on attributes. By default, the tool does not insert isolation cells on such nets.
- verbose
Provides detailed report on the isolation cells inserted in the design.
- enable *enable_signal*
Specifies the signal in the design to be connected to the enable pin of the inserted isolation cells. You can specify this signal as either a net, a pin, or a port. This is a required argument.
- object_list *objects*
Specifies a list of pins, ports, or nets on which to insert the isolation cells. If you specify object names, the tool resolves conflicts in the order of ports, pins and nets. This is a required argument.
- reference *lib_cell_name*
Specifies the name of the library cell of the isolation cell. This can be either an isolation (ISO) cell or an enabled level-shifter (ELS) cell. This is a required argument.

DESCRIPTION

The **insert_isolation_cell** command inserts isolation cells on the netlist objects specified in the **-object_list** option and connects the enable pin of each inserted cell to the enable signal specified in the **-enable** option. The tool determines the set of possible reference cells for the ISO or ELS cells by searching the link libraries for cells matching the name specified in the **-reference** option.

The tool selects the reference cell for each netlist object by using the following

strategy:

- * If the driver and all loads of the associated net operate at the same voltage, the tool selects an ISO cell that matches the operating condition of the subdesign in which the net resides. If such an ISO cell is not found, but an appropriate ELS cell is found, the tool selects the ELS cell.

- * If the driver and loads of the associated net operate at different voltages (a multivoltage net), the tool selects the first ELS cell that implements voltage-level shifting. ISO cells are not used for multivoltage nets.

The tool determines where to put each ISO or ELS cell based on the type of the netlist object:

- * If you specify a subdesign port, the tool inserts the ISO or ELS cell on the net connection inside the subdesign, next to the specified port. When you specify a driver port, the ISO or ELS cell connects to all paths driven by the port. When you specify a load port, the ISO or ELS cell connects only to the path ending at the load port.

To specify subdesign ports, use the **get_ports** command.

- * If you specify a subdesign pin, the tool inserts the ISO or ELS cell on the net connection outside the subdesign, next to the specified pin. When you specify a driver pin, the ISO or ELS cell connects to all paths driven by the pin. When you specify a load pin, the ISO or ELS cell connects only to the path ending at the load pin.

To specify subdesign pins, use the **get_pins** command.

- * If you specify a net, the tool inserts the ISO or ELS cell in net's parent subdesign.

To specify a net, use the **get_nets** command.

By default, the tool uses the <power_domain_name>_ISO_<count> naming rule to generate names for isolation cells. Each isolation cell inside its own domain gets a unique count. You can add a prefix string to the automatically generated name by specifying a value for the **isolation_cell_naming_prefix** variable.

In the following situations, the command does not insert an ISO or ESL cell on the associated net and flags the net as a violating net:

- * The tool cannot find an appropriate library cell for the net.
- * The net has multiple drivers.
- * The net is connected to a bidirectional pin.
- * The net has a dont_touch or always_on attribute.

You can use the **-force** option to force the insertion of isolation cells on such nets. However, you should not use this option when the operating conditions of the isolation cell and the subdesign do not match.

NOTE: This command is supported only on uniquified designs and designs with power domains.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example inserts a GTECH isolation cell on the IN net inside the I1 hierarchical instance. The enable pin of the isolation cell is wired to the ENABLE port of the design.

```
prompt> insert_isolation_cell -reference GTECH_ISO1_EN0 \  
        -object [get_nets {I1/IN}] \  
        -enable  [get_ports ENABLE]
```

The following example inserts a GTECH isolation cell on the net connected to port A of instance I1. Because a port is specified, the isolation cell is inserted inside the I1 block.

```
prompt> insert_isolation_cell -reference GTECH_ISO1_EN0 \  
        -object [get_ports {I1/A}] \  
        -enable  [get_ports ENABLE]
```

The following example inserts a GTECH isolation cell on the net connected to pin A of instance I1. Because a pin is specified, the isolation cell is inserted outside the I1 block.

```
prompt> insert_isolation_cell -reference GTECH_ISO1_EN0 \  
        -object [get_pins {I1/A}] \  
        -enable  [get_ports ENABLE]
```

The following example inserts an enabled level shifter on the RET net. The enable pin of enabled level shifter is wired to the en net inside the PWRCTRL hierarchical instance.

```
prompt> insert_isolation_cell -reference LVLHLEHX2 \  
        -object [get_nets RET] \  
        -enable [get_nets PWRCTRL/en]
```

SEE ALSO

```
remove_isolation_cell(2)  
get_pins(2)  
get_ports(2)  
get_nets(2)
```


insert_level_shifters

Inserts appropriate level shifters in the current design.

SYNTAX

```
status insert_level_shifters  
[-preserve]  
[-all_clock_nets]  
[-clock_net clock_name]  
[-verbose]
```

Data Types

clock_name string

ARGUMENTS

-preserve

Preserves the existing level shifters in the netlist. The command sets the **dont_touch** attribute on the appropriate load or driver net(s). The command does not remove a level-shifter cell even if it has voltage violations. The command does insert level shifters with this option but will try to find the correct operating conditions for the existing level shifters.

-all_clock_nets

Allows the addition of level-shifter cells for all of the clock nets in the design. By default, the command does not perform level shifter insertion for clock nets. This option and **-clock_net** are mutually exclusive.

-clock_net *clock_name*

Specifies the name of the clock net on which the level shifters are to be inserted. By default, the command does not perform level shifter insertion for clock nets. This option and **-all_clock_nets** are mutually exclusive.

-verbose

Displays all level shifters that the command finds in the target libraries. This option also displays all possible operating points at which each level shifter could be used by the command.

DESCRIPTION

This command inserts level shifters in the current design. The level shifters are inserted between the threshold and all of the nets that need the voltage adjustments according to the user-defined strategy. You can set the strategy and threshold using the **set_level_shifter_strategy** and **set_level_shifter_threshold** commands.

This command may fail to insert level shifters under the following conditions:

- There are no appropriate level shifters in the target library for particular input and output voltages.

- The appropriate level shifter exists in the target library but it is set with the **dont_use** attribute.
- The net is set with the **dont_touch** attribute.
- The net is driven by multiple drivers at different voltages.
- The net is a clock net and you have not used the **-all_clock_nets** or **-clock_net** options.

This command implicitly removes all existing buffer-type level-shifter cells from the design, unless they are set as dont_touch or invoked with the **-preserve** option.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example inserts level shifters in a design:

```
prompt> insert_level_shifters
```

The following example shows the result of using the **-verbose** option:

```
prompt> insert_level_shifters -verbose
```

```
Information: Analyzing target libraries for level-shifters
```

```
=====
Level Shifter Library      type   Opcond    vi->vo   P  T   Tree    calc_mode
=====
LVL8          max_ls_v7_v8  NLDM    max_v7_v8  0.7->0.8 1  125 balanced _
LVL8          max_ls_v7_v9  NLDM    max_v7_v9  0.7->0.9 1  125 balanced _
=====
```

```
Information: Found '2' level shifters.
```

```
Information: Total '5' level shifters are inserted
```

SEE ALSO

```
check_level_shifters(2)
remove_level_shifters(2)
set_level_shifter_strategy(2)
set_level_shifter_threshold(2)
```

insert_metal_filler

Fills an empty track with the metal wires to meet the metal density rules.

SYNTAX

```
int insert_metal_filler
[-out self | cellview_name]
[-purge]
[-bounding_box {{llx lly} {urx ury}}]
[-dont_overwrite]
[-timing_driven]
[-insert_as_instance instance_name]
[-tie_to_net none | ground | net_name]
[-create_floating_vias]
[-floating_via_ftr_spacing]
[-routing_space route_space]
[-from_metal from_metal_number]
[-to_metal to_metal_number]
[-width {layer width}]
[-space {layer space_between_fills}]
[-min_length {layer min_length}]
[-max_length {layer max_length}]
[-space_to_route {layer keep_from_metal_space}]
[-space_to_pg {layer keep_from_pg_nets_space}]
[-stagger {layer}]
[-x_offset {layer x_distance}]
[-y_offset {layer y_distance}]
[-dont_snap_fill_to_track]
[-fill_poly]
[-distance_to_boundary poly_to_cell_distance]
```

Data Types

<i>llx</i>	float
<i>lly</i>	float
<i>urx</i>	float
<i>ury</i>	float
<i>instance_name</i>	string
<i>route_space</i>	float
<i>from_metal_number</i>	integer
<i>to_metal_number</i>	integer
<i>layer</i>	string
<i>width</i>	float
<i>space_between_fills</i>	float
<i>min_length</i>	float
<i>max_length</i>	float
<i>keep_from_metal_space</i>	float
<i>keep_from_pg_nets_space</i>	float
<i>x_distance</i>	float
<i>y_distance</i>	float
<i>poly_to_cell_distance</i>	string

ARGUMENTS

- `-out self | cellview_name`
Specifies where to generate output. Use **self** to generate output in the current cellview. Use *cellview_name* to generate output in the specified cell view. The default value is *cell_name*.
- `-purge`
Clears up all metal fill in the current cell view. You can purge fill-track objects only on selected metal layers. You can control the fill-track objects to be purged by changing the numbers in the **-from_metal** and **-to_metal** options. By default, without specifying metal in either the **-from_metal** or **-to_metal** options, the command purges all fill-track objects in the current cell view.
- `-bounding_box {{llx lly} {urx ury}}`
Indicates a list of float values as the coordinates of the rectangle where metal fills are to be inserted. By default, the working area is the entire chip.
- `-dont_overwrite`
Specifies not to modify existing metal fill. You can use this option to incrementally insert metal fill.
- `-timing_driven`
Performs all necessary timing checks and calculations to create fills without violating timing. This can result in wider spacing for metal fills around timing-critical wires. This option might conflict with the local minimum metal density specification, making it difficult to meet both density and timing requirements in some critical areas.
- `-insert_as_instance instance_name`
Specifies the cell instance to be created for the fill view in the current design. If you use this option, you should also specify the **-out** option with the *cellview_name* argument.
- `-tie_to_net none | ground | net_name`
Specifies whether or not the fill metal is tied to the net; and, if so, specifies the type of tie. Use **none** if you want the fill metal to remain floating and not be tied to a net. Use **ground** to tie the fill metal to ground. Use *net_name* to tie the fill metal to a specific net. If there is more than one ground net, the name of the net must be specified by using a *net_name* value. The specified net must be of power or ground type. The default value is **none**.
- `-create_floating_vias`
Creates floating vias to meet the via density rule.
- `-floating_via_ftr_spacing`
Uses the *-space_to_route* value of each layer for floating vias to keep away from existing metal. Otherwise the minSpacing of each layer is used.
- `-routing_space route_space`
Specifies the space between normal routing wires and the fill metal. This value is multiplied by the minSpacing of the metal layer. The default is **1.0**.

-from_metal *from_metal_number*
 Specifies the starting metal layer from which empty tracks are filled and the uppermost layer specified by the **-to_metal** option. By default, *from_metal_number* is **1**.

-to_metal *to_metal_number*
 Fills empty tracks beginning with the specified *from_metal_number* layer and ending at the *to_metal_number* layer. By default, *to_metal_number* is **15**.

-width {*layer width*}
 Specifies the width for fill tracks. You can specify a list of layers and widths. Each layer must be a string, such as *poly*, *m1*, *m2*, and so on. By default, the command uses the `defaultWidth` from the physical library.

-space {*layer space_between_fills*}
 Specifies the space for fill tracks. You can specify a list of layers and spaces. Each layer must be a string, such as *poly*, *m1*, *m2*, and so on. Spacing refers to the lengthwise spacing between fills of the same layer only. By default, the command uses the `minSpacing` from the physical library.

-min_length {*layer min_length*}
 Specifies the minimum length for fill tracks. You can specify a list of layers and minimum lengths. Each layer must be a string, such as *poly*, *m1*, *m2*, and so on. By default, the command uses the `minWidth` from the physical library.

-max_length {*layer max_length*}
 Specifies the maximum length for fill tracks. You can specify a list of layers and maximum lengths. Each layer must be a string, such as *poly*, *m1*, *m2*, and so on. By default, the command has no limit on the maximum length.

-space_to_route {*layer keep_from_metal_space*}
 Specifies the space to keep away from existing metal. You can specify a list of layers and spaces. Each layer must be a string, such as *poly*, *m1*, *m2*, and so on. By default, the command uses the `minSpacing` of each layer.

-space_to_pg {*layer keep_from_pg_nets_space*}
 Specifies the space to keep away from power/ground nets. You can specify a list of layers and spaces. Each layer must be a string, such as *poly*, *m1*, *m2*, and so on. By default, the command uses the **-space_to_route** value of each layer.

-stagger {*layer*}
 Specifies the list of layers for which a stagger pattern of metal fill is to be generated. Each layer must be a string, such as *poly*, *m1*, *m2*, and so on. You can generate a stagger pattern by turning on this option or by specifying the `x-offset` and `y-offset`.

-x_offset {*layer x_distance*}
 Specifies the x-offset for each metal layer to be used for metal fill. Each layer must be a string, such as *poly*, *m1*, *m2*, and so on. The default is **0**.

-y_offset {*layer y_distance*}
 Specifies the y-offset for each metal layer to be used for metal fill. Each layer must be a string, such as *poly*, *m1*, *m2*, and so on. The default is **0**.

-dont_snap_fill_to_track
Skips snapping the fill to wiretracks. Use this option only if you are also using either the **-stagger** option or the **-x-offset** and **y-offset** options.

-fill_poly
Fills the poly layer. By default, the command does not fill the poly layer.

-distance_to_boundary *poly_to_cell_distance*
Specifies the spacing of the poly to the cell boundary. This distance is the specified distance plus poly-to-poly minimum spacing. The default value is 0.0.

DESCRIPTION

This command fills all of the empty tracks with the metal wires to meet the metal density rules required by most processes. This procedure is performed after routing and before layout versus schematic, design rule checking, or layout parasitic extraction (LPE).

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example fills all of empty tracks with metal wires.

```
prompt> insert_metal_filler
```

The following example fills empty tracks without overwriting the existing cellview.

```
prompt> insert_metal_filler -dont_overwrite
```

The following example fills the poly layer by maintaining spacing with the cell boundary as specified.

```
prompt> insert_metal_filler -fill_poly\  
-distance_to_boundary 2
```

The following example fills empty tracks and creates a cell instance for the fill view in the current design.

```
prompt> insert_metal_filler\  
-insert_as_instance Instance
```

The following example fills the tracks from metal layer 1 to 4 within the area specified by the bounding box.

```
prompt> insert_metal_filler\  
-bounding_box {{10 20} {100 300}}\  
-from_metal 1 -to_metal 4
```

The following example fills tracks and ties them to ground, keeping space between the routing wires and fill metals as indicated by using the **-routing_space** option.

```
prompt> insert_metal_filler -tie_to_net ground\  
-routing_space 3
```

The following example fills the tracks from metal layer 1 to 2 and ties them to VDD.

```
prompt> insert_metal_filler -tie_to_net VDD\  
-from_metal 1 -to_metal 2
```

The following example fills tracks with width 0.1 for metal 1 and spacing 0.1 for poly and length ranging from 2 to 10 for metal 1 2, and 3. It applies default values for any others that you do not specify.

```
prompt> insert_metal_filler -width {m1 0.1}\  
-space {poly 0.1}\  
-min_length {m1 2 m2 2 m3 2}\  
-max_length {m1 10 m2 10 m3 10}
```

SEE ALSO

```
insert_diode(2)  
insert_ng_filler(2)  
insert_pad_filler(2)  
insert_stdcell_filler(2)  
insert_well_filler(2)  
set_parameter(2)
```

insert_ng_filler

Generates notch and gap filling information for filling notches and gaps that are smaller than the minimum distance required between objects of the same net on the same layer.

SYNTAX

```
int insert_ng_filler
[-include_existing_notch_gap_fill_cell]
[-skip_filling_child_cell]
[-dont_apply_fat_wire_rule]
[-dont_fill_corner_to_corner]
[-out outname]
[-notch_layer_data_type notch_datatype]
[-gap_layer_data_type gap_datatype]
[-layers layer_list]
```

Data Types

<i>outname</i>	string
<i>notch_datatype</i>	integer
<i>gap_datatype</i>	integer
<i>layer_list</i>	list

ARGUMENTS

- include_existing_notch_gap_fill_cell
Specifies to use notch and gap cells that exist in the netlist. The notch and gap cells have a .FILL extension.
- skip_filling_child_cell
Specifies not to read the child cell's port data and apply filling on the child ports. By default, filling is applied on the child ports.
- dont_apply_fat_wire_rule
Specifies not to apply the fat wire rule when filling the notches and gaps. By fault, the fat wire rule is applied. See the man page for **set_fact_contact_options** for further information.
- dont_fill_corner_to_corner
Specifies not to fill the corner-to-corner notch errors. By default, the tool fills these notch errors.
- out *outname*
Specifies the error cell name from which to save any notch and gap errors. By default, notch and gap errors are not saved.
- notch_layer_data_type *notch_datatype*
Specifies the notch layer data type. Valid values are 0 through 255. If not specified, the default is 0.
See the DESCRIPTION section for more information.

`-gap_layer_data_type gap_datatype`

Specifies the gap layer data type. Valid values are 0 through 255. If not specified, the default is 0.

See the DESCRIPTION section for more information.

`-layers layer_list`

Specifies the layers to fill for notch and gap. By default, the value is {M1 M2 M3 ... M12}. The tool fills M1 to the maximum metal available in the design. The maximum metal layer supported is M12.

DESCRIPTION

The **insert_ng_filler** command generates notch and gap filling information for filling notches and gaps that are smaller than the minimum distance required between objects of the same net on the same layer. The information is stored in the FILL view cell. For example, the notch and gap cells for a top design cell named *rgb* is *rgb.FILL*. This information is used when you translate the data to GDSII stream.

The **-notch_layer_data_type** and **-gap_layer_data_type** options values from 0 through 6 are predefined as follows:

Data Type	Value
-----	-----
No data	0
Bit array	1
Two byte signed integer	2
Four byte signed integer	3
Four byte real	4 (not used)
Eight byte real	5
ASCII string	6

Values from 7 through 255 are user-defined.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

```
prompt> insert_ng_filler -include_existing_notch_gap_fill_cell \  
-skip -dont_apply_fat_wire_rule \  
-dont_fill_corner_to_corner \  
-notch_layer_data_type 2 -gap_layer_data_type 5 \  
-layer {m1 m3 m5}
```

```
prompt> insert_ng_filler -include_exisitng_notch_gap_fill_cell \  
-dont_apply_fat_wire_fule -out test.err
```

```
prompt> insert_ng_filler -skip_filling_child_cell \  
        -dont_fill_corner_to_corner -layer {M1 M2 M4 M6}  
  
prompt> insert_ng_filler -notch_layer_data_type 3 \  
        -gap_layer_data_type 1
```

SEE ALSO

```
insert_diode(2)  
insert_metal_filler(2)  
insert_pad_filler(2)  
insert_stdcell_filler(2)  
insert_well_filler(2)
```

insert_pad_filler

Fills gaps in the pad ring with instances of pad filler cells.

SYNTAX

```
int insert_pad_filler  
-cell lib_cells  
[-overlap_cell overlap_lib_cells]  
[-voltage_area voltage_area_list]  
[-bounding_box rectangle]  
[-prefix prefix]  
[-no_left]  
[-no_right]  
[-no_bottom]  
[-no_top]
```

Data Types

<i>lib_cells</i>	string
<i>overlap_lib_cells</i>	string
<i>voltage_area_list</i>	list
<i>prefix</i>	string

ARGUMENTS

-cell *lib_cells*
Specifies the names of the pad filler cells to be used. The tool adds the cells in the priority that they are specified.

-overlap_cell *overlap_lib_cells*
Specifies the names of pad filler cells that can overlap other pad filler cells. The names specified here also need to be specified in the **-cell** list.

-voltage_area *voltage_area_list*
Specifies the voltage areas where the pad fillers are to be inserted. Default is all voltage areas.

-bounding_box *rectangle*
Specifies the rectangle region where pad cells are to be inserted. By default, the working area is the entire chip.

-prefix *prefix*
Inserts an extra string of user choice into the name of pad filler cell instances created by this command. The *prefix* can later be used to identify the pad filler cells added.

-no_left
Does not insert pad filler cells on the left boundary.

-no_right
Does not insert pad filler cells on the right boundary.

`-no_bottom`

Does not insert pad filler cells on the bottom boundary.

`-no_top`

Does not insert pad filler cells on the top boundary.

DESCRIPTION

The **insert_pad_filler** command fills gaps in the pad ring with instances of pad filler cells in the library. It is recommended that you complete the routing process before you add pad filler cells.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example inserts pad filler cells PFILL_2X and PFILL_1X in the given priority order:

```
prompt> insert_pad_filler -cell "PFILL_2X PFILL_1X"
```

The following example inserts pad filler cells PFILL_2X and PFILL_1X on the pad ring and inside voltage area V1 only. The PFILL_1X cell is allowed to overlap other pad filler cells to fill gaps that are too small for any pad filler cell.

```
prompt> insert_pad_filler -cell "PFILL_2X PFILL_1X" \  
-overlap_cell "PFILL_1X" -voltage_area {V1}
```

The following example inserts pad filler cells PFILL_2X and PFILL_1X on the pad ring, skipping the top and right boundary:

```
prompt> insert_pad_filler -cell "PFILL_2X PFILL_1X" -no_top -no_right
```

SEE ALSO

`insert_diode(2)`
`insert_metal_filler(2)`
`insert_ng_filler(2)`
`insert_stdcell_filler(2)`
`insert_well_filler(2)`
`set_parameter(2)`

insert_port_protection_diodes

Inserts diodes around the specified ports to protect them. One diode is added to each specified port.

SYNTAX

```
status insert_port_protection_diodes  
[-prefix name]  
[-ignore_dont_touch ignore_dont_touch_net]  
{-diode_cell lib_cell}  
-port list
```

Data Types

<i>name</i>	string
<i>lib_cell</i>	collection of one lib cell

ARGUMENTS

-prefix *name*
Specifies the prefix used to identify the newly added diode cells. If you do not specify this option, no prefix is used and the tool will assign a cell name. This option is optional.

-ignore_dont_touch *ignore_dont_touch_net*
Specifies that even if a net is a "dont_touch" net, a port protection diode may also be added into that net. By default, when this command is inserting port_protection_diodes, if a net is "dont_touch" net, no diode will be added into that net. This option is optional.

-diode_cell *lib_cell*
Specifies the name of the library reference cell used for the new diodes. You can specify only a single library cell. This option is required.

-port *list*
Specifies the ports to which to connect protection diodes. This option is required.

DESCRIPTION

The **insert_port_protection_diodes** command performs the following tasks: 1.Creates the new diodes and adds them into the netlist. 2.Connects the new diodes to the specified ports. 3.Legalizes the diodes. 4.Marks the diodes as FIXED.

This command is supposed to be used in "pre-route" stages for the purpose of "antenna prevention". A typical usage model for this command may be like:
insert_port_protection_diodes place_opt clock_opt route_opt

Please note: if this command succeeds, all of the "inserted new diodes" will be located at legal locations and marked as FIXED. However, this does not necessarily mean that other movable standard cells will be legal. There may be a chance that

some movable standard cells are overlapping with the "inserted new diodes". As for how to remove the overlapping between the "inserted new diodes" and existing movable standard cells, it will be up to the user's option. After the port protection diodes have been added, users can run command `place_opt/clock_opt/route_opt` or just a `legalize_placement` command to make the entire design legal.

Multicorner-Multimode Support

This command uses information from all active scenarios.

EXAMPLES

The following example runs the **`insert_port_protection_diodes`** command.

```
prompt> insert_port_protection_diodes -prefix MY_DIODE \  
-diode_cell [get_lib_cell XXX] [all_inputs *]
```

SEE ALSO

```
report_port_protection_diodes(2)  
legalize_placement(2)
```

insert_redundant_vias

Replaces the specified single-cut vias with a multiple-cut via array. It can also replace a single via with another single via that has a different contact code.

SYNTAX

```
status insert_redundant_vias
-from_via list_of_from_vias
-to_via list_of_to_vias
[-to_via_x_size list_of_x-sizes]
[-to_via_y_size list_of_y-sizes]
[-via_array_no_swap]
[-optimize_level level_of_optimization]
[-num_cpus number_of_cpus]
[-auto_mode preview | insert]
```

Data Types

<i>list_of_from_vias</i>	list
<i>list_of_to_vias</i>	list
<i>level_of_optimization</i>	integer
<i>number_of_cpus</i>	integer

ARGUMENTS

-from_via *list_of_from_vias*
Specifies a list of via names to be replaced. The via names are defined in the technology file.

-to_via *list_of_to_vias*
Specifies a list of names corresponding to the new vias to be created. The via names are defined in the technology file. The *list_of_to_vias* list is usually the same as the corresponding *list_of_from_vias* list specified in the **-from_via** option. If the list is different, the layer definition of the two vias must be the same.

-to_via_x_size *list_of_x-sizes*
Specifies a list of the x-sizes for the newly created vias. For any given via, either the x-size or y-size must have a value equal to 1. The default value is **1**.

-to_via_y_size *list_of_y-sizes*
Specifies a list of the y-sizes for the newly created vias. For any given via, either the x-size or y-size must have a value equal to 1. The default value is **1**.

-via_array_no_swap
Specifies not to swap the row and column of via arrays. In other words the 1 x N and N x 1 via arrays are not considered equivalent. By default, the command allows you to swap the row and column of via arrays.

`-optimize_level level_of_optimization`

Determines how aggressively to try different positions for a line via array insertion. A value of **0** tries relatively fewer positions; and a value of **1** tries more. The default is **0**.

`-num_cpus number_of_cpus`

Specifies the number of CPUs in distributed processing. The default is **1**.

`-auto_mode preview | insert`

Specifies the mode in which to automatically generate via lists.

If you specify **preview**, all vias defined in the technology file are listed, but no redundant vias are inserted.

If you specify **insert**, redundant vias that are defined as default vias in the technology file are inserted.

This option is mutually exclusive with the **-from_via**, **-to_via**, **-to_via_x_size**, and **-to_via_y_size** options.

DESCRIPTION

This command can replace all of the specified single-cut vias in the design with a multiple-cut via array or even a different via with the same layers. From netlist connectivity point of view, using a single-cut via is no different from using a multiple-cut via array, so additional via cuts in a multiple-cut via array are regarded as redundant vias from this point of view. However, redundant vias can increase yield during manufacturing and are an important design-for-manufacturing (DFM) feature.

The single-cut via is replaced only when the new via or via array does not introduce any design rule violations.

The **insert_redundant_vias** command provides two insertion methods: manual or automatic.

The manual method (using the **-from_via**, **-to_via**, **-to_via_x_size**, and **-to_via_y_size** options) provides the flexibility to customize the redundant via insertion. During manufacturing, different vias have different yield loss impact and redundant via insertions of different vias have a different impact of the yield of the metal layers. To maximize the yield, you can customize which kinds of redundant vias are inserted (by using the **-from_via** and **-to_via** options) and how to insert these redundant vias (by using the **-to_via_x_size** and **-to_via_y_size** options).

The automatic method (using the **-auto_mode** option) provides an easy-to-use method. You can list all of the vias on all the layers (without referencing the technology file) by using **-auto_mode preview**. You can then cut and paste part or all of the listed vias into the **-to_via** and **-from_via** options to insert redundant vias. You can also use **-auto_mode insert** to insert the default vias (as defined in the technology file) for all layers.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example replaces all VIA23 single-cut vias with 1x3 VIA23 arrays and all VIA34 single-cut vias with 5x1 VIA34 arrays.

```
prompt> insert_redundant_vias -from_via {VIA23 VIA34} \  
-to_via {VIA23 VIA34} -to_via_x_size {1 5} \  
-to_via_y_size {3 1}
```

The following example replaces all VIA23 single-cut vias with 1x3 VIA23 arrays.

```
prompt> insert_redundant_vias -from_via VIA23 -to_via VIA23 \  
-to_via_x_size 1 -to_via_y_size 3
```

The following example replaces one list of vias with another list of vias of different x-sizes and y-sizes.

```
prompt> insert_redundant_vias \  
-from_via {V1 V1 V2 V2 V3 V3 V4 V4 V6} \  
-to_via {V1 V1 V2 V2 V3 V3 V4 V4 V6S} \  
-to_via_x_size {1 2 1 2 1 2 1 2 1} <HardSpace<plain  
-to_via_y_size {2 1 2 1 2 1 2 1 1}
```

SEE ALSO

```
insert_diode(2)  
insert_metal_filler(2)  
insert_ng_filler(2)  
insert_pad_filler(2)  
insert_stdcell_filler(2)  
insert_well_filler(2)  
set_parameter(2)
```

insert_spare_cells

Inserts spare cells in the legalized design.

SYNTAX

```
status insert_spare_cells
[-lib_cell lib_cell_name]
-cell_name prefix_name
[-num_instances number]
[-hier_cell hierarchy_name]
[-num_cells {lib_cell_name number}]
[-tie]
[-skip_legal]
```

Data Types

<i>lib_cell_name</i>	string
<i>prefix_name</i>	string
<i>number</i>	int
<i>hierarchy_name</i>	string

ARGUMENTS

-lib_cell *lib_cell_name*
Specifies the names of library cells for which the spares cells are to be created. If you specify more than one library cell, the instances of these cells are placed as close as possible.

-cell_name *prefix_name*
Specifies the prefix name of the spare cells to be used by the tool. An integer prefix is automatically appended to the prefix name for each instance. If a cell with a generated name already exists, the tool automatically assigns a unique name for the cell name.

-num_instances *number*
Specifies the number of instances of the library cell that are inserted as spare cells.

-hier_cell *hierarchy_name*
Specifies the level of hierarchy at which the spares cells are inserted.

-num_cells {*lib_cell_name number*}
Specifies the number of instances for each different *lib_cell_name*. This option is mutually exclusive with options *-lib_cell* and *-num_instance*.

-tie
Ties the input pins of the inserted spare cells to logic zero.

-skip_legal
Skips legalization checking. However, the command still try to insert spare cells in legalized locations.

DESCRIPTION

This command inserts a specified number of instances of the library cell as spare cells in an already legalized design. This command adds the spare cells to the logical netlist and tries to place them as evenly as possible on the already placed logic, but avoids all placement blockages. The location of the existing logic is not changed and new cells are placed according to available legal locations.

If the input netlist already has spare cells that are placed during **physopt**, this command ignores the existing spare cells and does not change their locations.

Specifying more than one library cell allows you to place the instances of these spare cells as close as possible. Thus, you can connect them manually during the ECO phase.

When you want to insert spare cells in a given hierarchy by using the *-hier_cell* option, the spare cells are placed in the maximum rectangular area consisting of all the placed cells of the specified hierarchy.

If you use the *-tie* option, all the input pins of the spare cells inserted in this instance of the command are tied to logic zero. If you do not specify this option, the inputs will not get tied to logic zero.

When the design is not legalized for some reasons, spare cells can be spared using the *-skip_legal* option. All the newly inserted spare cells would be given legal locations.

To insert spare cells with different number of instances from different library cells, use the *-num_cells* option. This option is mutually exclusive with the *-lib_cell* and *-num_instances* options.

Apply this command after all the optimization and legalization steps and user constraints are met. If the design is not legalized, use the *-skip_legal* to continue the flow. Any further legalization would likely to move the placement of spare cells unless you explicitly set them as fixed cells.

This command modifies the logical netlist of the design.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example inserts 20 instances of a library cell named *and2* with the prefix *spares* and with the input pins tied to logic zero.

```
prompt> insert_spare_cells -lib_cell and2 \  
-cell_name spares -num_instances 20 -tie
```

The following example inserts 15 instances of library cells named *and2* and *nor2* with the prefix *spares* in the sub design instance *I_SUB*.

```
prompt> insert_spare_cells -lib_cell {and2 nor2} \  
-cell_name spares -num_instances 15 -hier_cell I_SUB
```

The following example inserts 10 instances of library cell named *and1* and 5 instances of library cell named *nor1* with the prefix name *spars* in the hierarchical cell instance *I_SUB*.

```
prompt> insert_spare_cells -num_cells {and1 10 nor1 5} \  
-cell_name spares -hier_cell I_SUB
```

SEE ALSO

`legalize_placement(2)`
`spread_spare_cells(2)`

insert_stdcell_filler

Fills empty spaces in standard cell rows with instances of master filler cells only.

SYNTAX

```
status insert_stdcell_filler
[-cell_without_metal lib_cell_without_list]
[-cell_with_metal lib_cell_with_list]
[-bounding_box rectangle]
[-dont_respect_hard_placement_blockage]
[-dont_respect_soft_placement_blockage]
[-between_std_cells_only]
[-randomize]
[-respect_overlap]
[-cell_without_metal_prefix cell_without_prefix]
[-cell_with_metal_prefix cell_with_prefix]
[-avoid_layers layers_list]
[-connect_to_power Power_net_name]
[-connect_to_ground Ground_net_name]
[-voltage_area voltage_area_list]
[-vt_filler lib_cell_list]
[-check_only]
[-restore_filler_snapshot]
[-vt_filler_prefix vt_filler_prefix]
[-respect_keepout]
[-leakage_power max_total_metal_filler_cell_leakage_power]
[-metal_filler_coverage_area max_total_metal_filler_cell_coverage_area]
[-no_1x]
```

Data Types

<i>lib_cell_without_list</i>	collection
<i>lib_cell_with_list</i>	collection
<i>rectangle</i>	rectangle
<i>cell_without_prefix</i>	string
<i>cell_with_prefix</i>	string
<i>layers_list</i>	collection
<i>voltage_area_list</i>	collection
<i>lib_cell_list</i>	collection of physical lib cells
<i>vt_filler_prefix</i>	string
<i>max_total_metal_filler_cell_leakage_power</i>	float
<i>max_total_metal_filler_cell_coverage_area</i>	float

ARGUMENTS

-cell_without_metal *lib_cell_without_list*
Specifies the list of filler cells that are from the physical libraries to be used. The tool adds them in the order that you specify, so specify them from the largest to smallest. The tool assumes that the specified cells contain no metal, so it does not check for design rule violations after filler insertion.

`-cell_with_metal lib_cell_with_list`

Specifies the master filler cells that are from the physical libraries to be used. The tool adds them in the order that you specify, so specify them from the largest to smallest. The specified cells contain metal and are inserted only if no design rule violations are detected after insertion. By default, the tool uses the classical router for design rule checking. If you specify Zroute by using **set_route_mode_options**, the tool uses Zroute.

`-bounding_box rectangle`

Specifies the rectangular region (in microns) where filler cells are to be inserted. By default, the working area is the entire chip.

`-dont_respect_hard_placement_blockage`

Specifies an option that will be obsolete. The tool issues a warning message that instructs you to use the **-ignore_hard_placement_blockage** option instead. The **-dont_respect_hard_placement_blockage** option inserts filler cells in hard placement blockage areas. By considering hard and soft placement blockages separately, you can insert filler cells in soft blockages, while still respect the hard placement blockages in the design. By default, the tool prevents the insertion of filler cells in hard placement blockage areas.

`-dont_respect_soft_placement_blockage`

Specifies an option that will be obsolete. The tool issues a warning message that instructs you to use the **-ignore_soft_placement_blockage** option instead. The **-dont_respect_soft_placement_blockage** option inserts filler cells in soft placement blockage areas. By default, the tool prevents the insertion of filler cells in soft placement blockage areas.

`-between_std_cells_only`

Inserts filler cells only in an area between two standard cells. A filler cell is not inserted if there are abutting macros or if it would be near the two ends of a row.

`-randomize`

Instructs the tool not to follow the specified list order. Use this feature when the filler cell masters are the same size.

`-respect_overlap`

Considers the overlap check rectangles outside the standard cell boundary to be part of the standard cell. The tool does not fill the areas under the overlap check rectangle. By default, filler cell insertion ignores the overlap check rectangle layer(192) of standard cells.

`-cell_without_metal_prefix cell_without_prefix`

Adds a prefix of your choice to the instance names of the filler cells without metal created by this command. You can later use the prefix to identify the filler cell instances.

`-cell_with_metal_prefix cell_with_prefix`

Adds a prefix of your choice to the instance names of the filler cells with metal created by this command. You can later use the prefix to identify the filler cell instances.

`-avoid_layers layers_list`

Prevents insertion of filler cells under the specified metal layers (from the

technology file) where preroute exists. This option works the same as the **-avoid_filler_under** option except that it takes layer names from technology file. By default, the tool attempts to fill in areas under all metal layers.

-connect_to_power *Power_net_name*

Specifies the name of the existing power net to which to connect filler cells.

-connect_to_ground *Ground_net_name*

Specifies the name of the existing ground net to which to connect filler cells.

-voltage_area *voltage_area_list*

Specifies a list of the voltage areas to insert filler cells.

-vt_filler *lib_cell_list*

Specifies the threshold voltage cells to be used as threshold voltage (vt) fillers.

If you do not specify this option, all threshold voltage cells are used. If you specify an empty list, no threshold voltage cells are added.

-check_only

Runs the command in checking mode, rather than insertion mode. In checking mode, this command lists the empty spaces in standard cell rows with instances of master filler cells in the library without copying them. This option does not support the **-no_1x** mode.

-restore_filler_snapshot

Restores the filler cell placement of the snapshot previously created if the standard cell placement of the filling areas has not changed. A filler cell snapshot is created when filler cells are deleted. This option allows you to preserve the previous filler cell placement whenever possible. This helps to preserve the timing and capacitance coupling characteristics of the design.

-vt_filler_prefix *vt_filler_prefix*

Specifies a prefix for the collection of threshold voltage fillers inserted by this command.

You can use the prefix later to identify the threshold voltage filler cell instances.

If you do not specify this option, the command uses the default prefix "xofiller!my_vt_filler_prefix!".

-respect_keepout

When specified, the tool respects hard and soft keepout margins on hard macros. Keepout areas are treated as if they are occupied by a blockage. This option applies only to keepout margins on hard macros.

-leakage_power *max_total_metal_filler_cell_leakage_power*

This specifies the max leakage power contributed by metal fillers and does not included any power contribution from standard cells(unit: mW).

-metal_filler_coverage_area *max_total_metal_filler_cell_coverage_area*

This specifies the max total coverage area of metal filler cells(unit: μm^2). This option is trying to reduce the coverage area of metal filler cells by removing them until the user specified value is met. Those cells include metal fillers not only existed originally but also inserted by this command.

-no_1x

Specifies whether you want to use no-1x filler insertion. When this option is set, the no-1x rule takes precedence over other filler insertion rules:

- The tool issues warning messages for any existing 1x gaps.
 - If placing a LR filler leaves a 1x gap, the tool does not place the LR filler at that location and issues a warning message.
 - If placing a Vt, metal, or non-metal filler leaves a 1x gap, the tool does not place the Vt, metal, or non-metal filler at that location but tries the next smaller Vt, metal, or non-metal filler instead.
- By default, this option is disabled. The **-check_only** mode does not support this option.

DESCRIPTION

This command fills empty spaces in standard cell rows with instances of master filler cells in the library. Perform placement (and clock tree synthesis, if applicable) before adding filler cells.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example fills empty spaces with filler cells named *FILL_4X*, *FILL_2X*, and *FILL_1X* in the given order. If the sizes of the filler cells are such that *FILL_4X* > *FILL_2X* > *FILL_1X*, this ordering minimizes the number of filler cells added by using the larger filler cells first.

```
prompt> insert_stdcell_filler -cell_without_metal \  
{FILL_4X FILL_2X FILL_1X}
```

The following example fills empty spaces with metal filler cells given by the ordered list *FILL_4XM*, *FILL_2XM*, and *FILL_1XM* at places where design rules are satisfied. If design rule errors occur, the command uses the non-metal filler cells named *FILL_4X*, *FILL_2X*, and *FILL_1X*.

```
prompt> insert_stdcell_filler -cell_without_metal \  
{FILL_4X FILL_2X FILL_1X} -cell_with_metal} \  
{FILL_4XM FILL_2XM FILL1XM}
```

The following example fills empty spaces with filler cells *FILL_2X* and *FILL_1X* in the bounding box (10.0,10.0) (5000.0,5000.0) and the command fills only areas between two standard cells. In addition, filler cells are not inserted under metal 3 or metal 4 preroutes.


```
prompt> insert_stdcell_filler \  
-cell_without_metal {FILL_2X FILL_1X} \  
-bounding_box {{10.0 10.0} {5000.0 5000.0}} \  
-between_std_cells_only -avoid_filler_under {M3 M4}
```

The following example fills empty spaces with filler cells named *FILL_2X* and *FILL_1X* in a voltage area named *VA1* and avoids a layer named *my_metal_2*.

```
prompt> insert_stdcell_filler \  
-cell_without_metal {FILL_2X FILL_1X} \  
-voltage_area {VA1}  
-avoid_layers "my_metal_2"
```

SEE ALSO

```
insert_diode(2)  
insert_metal_filler(2)  
insert_ng_filler(2)  
insert_pad_filler(2)  
insert_well_filler(2)  
set_route_mode_options(2)
```

insert_tap_cells_by_rules

Adds tap cells while meeting the maximum spacing design rule from the diffusion to the substrate or well contact.

SYNTAX

```
status insert_tap_cells_by_rules
[-drc_spacing_check | -tap_cell_insertion]
[-tap_distance_based | -drc_spacing_based]
[-move | -freeze]
[-tap_master physical_lib_cell]
[-tap_layer layer_name]
[-tap_distance_limit distance]
[-n_well_layer layer_name]
[-p_well_layer layer_name]
[-contact_layer layer_name]
[-p_diffusion_layer layer_name]
[-n_diffusion_layer layer_name]
[-p_implant_layer layer_name]
[-n_implant_layer layer_name]
[-tap_spacing_design_rule distance]
[-tap_filler_name_identifier prefix]
[-ignore_hard_blockage]
[-ignore_soft_blockage]
[-ignore_double_back_sharing]
[-connect_to_power_net power_net]
[-connect_to_ground_net ground_net]
[-no_tap_cells_under_metal_layer metal_layers_list]
[-voltage_area voltage_area_list]
[-respect_keepout]
```

Data Types

<i>physical_lib_cell</i>	string
<i>layer_name</i>	string
<i>distance</i>	float
<i>prefix</i>	string
<i>power_net</i>	net
<i>ground_net</i>	net
<i>metal_layers_list</i>	collection
<i>voltage_area_list</i>	collection

ARGUMENTS

-drc_spacing_check
Reports spacing violations without inserting tap cells. The **-drc_spacing_check** and **-tap_cell_insertion** options are mutually exclusive. You can specify only one.

-tap_cell_insertion
Inserts tap cells while meeting design rules specified. The **-drc_spacing_check** and **-tap_cell_insertion** options are mutually exclusive.

insert_tap_cells_by_rules

You can specify only one.

-tap_distance_based

Selects the distance based method to ensure design rule compliance. This method uses a simple spacing measurement from a standard cell to a tap cell. It ignores the actual metal and contact layers of the standard cell and the tap cell. This conservative distance model may use more tap cells than necessary. Specify this option when the actual layout of the standard cell or the tap cell is not available in the cell library. The **-tap_distance_based** and **-drc_spacing_based** options are mutually exclusive. You can specify only one.

-drc_spacing_based

Selects the spacing design rule checking (DRC) based method to ensure design rule compliance. This method reads the actual layouts of the standard cells and the tap cells. The tool analyzes the diffusion layers, well layers, and contact layers to ensure spacing compliance. This results in inserting fewer tap cells because the tool recognizes an existing tap in a standard cell. The **-tap_distance_based** and **-drc_spacing_based** options are mutually exclusive. You can specify only one.

-move

Allows the moving of standard cells to avoid overlapping with the tap cells but might affect timing results of the design. The **-move** and **-freeze** options are mutually exclusive. You can specify only one.

-freeze

Disables the moving of standard cells during tap cells insertion. This option may insert more tap cells and may produce a placement with design rule violations. The **-move** and **-freeze** options are mutually exclusive. You can specify only one.

-tap_master *physical_lib_cell*

Specifies the library cell to be used as the tap cell master.

-tap_layer *layer_name*

Specifies the tap layer which contains the location of a tap in a cell. The tap layer typically has no physical correspondence to an actual mask layer.

-tap_distance_limit *distance*

Specifies the maximum spacing design rule between the standard cell and the tap cell.

-n_well_layer *layer_name*

Specifies the N-well layer only when you select the DRC spacing based tap cell insertion method.

-p_well_layer *layer_name*

Specifies the P-well layer only when you select the DRC spacing based tap cell insertion method.

-contact_layer *layer_name*

Specifies the metal to diffusion contact layer only when you select the DRC spacing based tap cell insertion method.

`-p_diffusion_layer layer_name`
 Specifies the P-diffusion layer only when you select the DRC spacing based tap cell insertion method.

`-n_diffusion_layer layer_name`
 Specifies the N-diffusion layer only when you select the DRC spacing based tap cell insertion method.

`-p_implant_layer layer_name`
 Specifies the P-Implant layer. Use this option only for a design with FRAM view in which P or N diffusions are on the same layer.

`-n_implant_layer layer_name`
 Specifies the N-Implant layer. Use this option only in the design with FRAM view in which P or N diffusions are on the same layer.

`-tap_spacing_design_rule distance`
 Specifies the maximum spacing rule from a P or N diffusion to a substrate or well tap. Use this option only when the DRC spacing based tap cell insertion method is selected.

`-tap_filler_name_identifier prefix`
 Specifies a name prefix to be used for inserted tap cells. You can use this prefix to find all inserted tap cells by name pattern matching.

`-ignore_hard_blockage`
 Ignores hard blockages and place tap cells inside them.

`-ignore_soft_blockage`
 Ignores soft blockages and place tap cells inside them.

`-ignore_double_back_sharing`
 Disables the sharing of tap cells. By default, the tool assumes that taps can be shared with an adjacent row when using a double-back row structure. This results in insertion of fewer tap cells due to sharing. Use this option when the wells of a double-back row pair are not shared, and, a tap from one row cannot be used for a different row.

`-connect_to_power_net power_net`
 Specifies a net to which the power port of a tap cell is connected after insertion.

`-connect_to_ground_net ground_net`
 Specifies a net to which the ground port of a tap cell is connected after insertion.

`-no_tap_cells_under_metal_layer metal_layers_list`
 Disables the insertion of tap cells under specified metal layers where a preroute exists. The **metal_layers_list** argument is a list of layer names from the technology file. By default, the tool attempts to fill in areas under all metal layers.

`-voltage_area voltage_area_list`
 Specifies the voltage areas where tap cells are inserted. Tap cells are not inserted outside the specified voltage areas. The value of **voltage_area_list**

must be a collection of voltage areas. By default, the tool insert tap cells in all voltage areas.

-respect_keepout

Respects hard and soft keepout margins on hard macros. The tool treats keepout areas as if they are occupied by a blockage. This option applies only to keepout margins on hard macros.

DESCRIPTION

This command inserts tap cells in a design while meeting the maximum spacing design rule between the diffusion and the tap cells. You typically use this command on a design after global placement is completed. A tap cell is a special non-logic cell with well and/or substrate ties. Tap cells are typically used when most or all of the standard cells in the library do not contain substrate or well taps. The design rule specifies the maximum spacing between every transistor of a standard cell and the well or substrate ties.

This command deploys two different insertion methods. The tap distance based method uses a simple distance model where the distance from a standard cell to a tap is not violated. This is a conservative tap insertion strategy. Use this method when the internal layout of a standard cell is not available.

The design rule checking (DRC) spacing based method reads the diffusion, well, and contact layers of the standard cell layout. By using intersection of the given layers, the P and N transistor diffusion area, the substrate and well contact locations are identified. The transistor diffusion area of a standard cell can use a tap inside another standard or tap cell. This method makes the most efficient use of tap cells and results in the insertion of fewer tap cells.

The **insert_tap_cells_by_rules** command can also connect the power or ground port of a tap cell to a specified power or ground net.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example inserts a tap cell named *MY_TAP* into a placed design where the distance from a standard cell to a tap cell is no more than 30.0 um, but does not place it under the metal layer named *metal1* where a preroute exists.

```
prompt> insert_tap_cells_by_rules -tap_master "MY_TAP" \  
-tap_cell_insertion -tap_distance_based -move \  
-tap_distance_limit 30.0 \  
-no_tap_cells_under_metal_layer {metal1}
```

SEE ALSO

`add_tap_cell_array(2)`

```
insert_stdcell_filler(2)
insert_well_filler(2)
```

insert_well_filler

Fills gaps between cells in the same row containing wells on the specified layer.

SYNTAX

```
status insert_well_filler
-layer layer_name_or_number
[-ignore_PRboundary]
[-fill_gaps_smaller_than gap_size]
[-higher_edge min | max]
[-lower_edge min | max]
-gap_type {tt | bb | tb | bt}
[-respect_blockages]
[-row_overlap {row_overlap_value}]
[-min_gap {min_gap_distance}]
[-max_gap {max_gap_distance}]
[-enclosure_only width]
```

Data Types

<i>layer_name_or_number</i>	string
<i>gap_size</i>	float
<i>row_overlap_value</i>	float
<i>width</i>	float

ARGUMENTS

- layer *layer_name_or_number*
Specifies the name or number of the well layer to be filled.
- ignore_PRboundary
Instructs the tool to ignore the PR boundary layer. By default, the tool respects the PR boundary (layer 207) and does not fill inside a PR boundary box that extends outside the cell.
- fill_gaps_smaller_than *gap_size*
Specifies that gap sizes less than the given threshold are to be filled. If a negative number or zero is entered, the default well spacing from the technology file is used. The default value is 1000.0.
- higher_edge min | max
Specifies the position of the higher edge of the well filler when the wells from the two standard cells do not line up. Use the keyword "min" or "max". If "min" is specified, the lower of the top edges of the two wells is used. If "max" is specified, the higher of the top edges of the two wells is used. The default is "min".
- lower_edge min | max
Specifies the position of the lower edge of the well filler when the wells from the two standard cells do not line up. Use the keywords "min" or "max". If "min" is specified, the lower of the bottom edges of the two wells is used. If "max" is specified, the higher of the bottom edges of the two wells is

used. The default is "min".

-gap_type {tt | bb | tb | bt}

Indicates the type of gap to fill. This will fill the gaps between rows of wells in a specified layer.

-respect_blockages

Specifies that placement blockages are not to be filled. By default placement blockages will be filled. This should be used only with gap_type option to fill between rows.

-row_overlap {row_overlap_value}

Specifies the amount of space that the gap could overlap the row. A positive number overlaps the row. A negative number adds space between the row and the gap. The default value is 0.0 and is valid only when gap_type is used.

-min_gap {min_gap_distance}

Specifies the minimum gap distance of the channel. If the channel is less than this distance then it is not filled. The default value is 0.0. This option is valid only with gap_type option.

-max_gap {max_gap_distance}

Specifies the maximum gap distance of the channel. The default value is 0.0. This option is valid only with gap_type option.

-enclosure_only width

Specifies the desired width of the enclosure.

DESCRIPTION

The **insert_well_filler** command fills gaps between cells either in the same row on the specified layer or between rows depending on which options are used. If the gap_type option is specified it fills gaps between rows otherwise gaps in the same row are filled. A well layer has a minimum spacing rule. Small gaps that violate the spacing rule need to be filled to avoid design rule errors. It is considered best practice to complete the routing before adding well fillers.

If the **-enclosure_only** option is turned on, this command adds a well enclosure around all standard cells in the design.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example fills the NWELL layer:

```
prompt> insert_well_filler -layer NWELL
```

The following example adds well fillers on layer 10 on gaps smaller than 3.0um.

If the wells from the two standard cells do not line up, the tool creates the largest fill box by using the larger top edge and the smaller bottom edge.

```
prompt> insert_well_filler -layer 10 -fill_gaps_smaller_than 3.0 \  
        -higher_edge "max" -lower_edge "min"
```

The following example inserts well fillers between rows on layer NWELL:

```
prompt> insert_well_filler -gap_type TT -layer NWELL
```

The following example invokes a special well filler mode where a well enclosure of 2.0um is inserted to the left and right side of each standard cell on the NWELL layer.

```
prompt> insert_well_filler -layer {NWELL} -enclosure_only 2.0
```

SEE ALSO

```
insert_diode(2)  
insert_metal_filler(2)  
insert_ng_filler(2)  
insert_pad_filler(2)  
insert_stdcell_filler(2)  
set_parameter(2)
```

insert_zrt_redundant_vias

Replaces single-cut vias with multiple-cut via arrays. It can also replace a single-cut via with another single-cut via that has a different contact code.

SYNTAX

```
status insert_zrt_redundant_vias
[-effort low | medium | high]
[-list_only]
[-timing_preserve_nets {collection_of_nets}]
[-timing_preserve_setup_slack_threshold slack_value]
[-timing_preserve_hold_slack_threshold slack_value]
```

Data Types

<i>collection_of_nets</i>	collection
<i>slack_value</i>	float

ARGUMENTS

-effort low | medium | high
Specifies the effort used to perform redundant via insertion. The default is medium.

-list_only
Lists the default via mappings to be used for redundant via insertion, but does not perform insertion.

-timing_preserve_nets {collection_of_nets}
Specifies the nets to preserve timing on.
If you specify this option, the command considers the specified nets as timing critical and skips redundant via insertion on them.

-timing_preserve_setup_slack_threshold slack_value
Specifies the setup slack threshold for timing critical nets.
If you specify this option, the command skips redundant via insertion on all nets with a worst setupslack less than or equal to this value.

-timing_preserve_hold_slack_threshold slack_value
Specifies the hold slack threshold for timing critical nets.
If you specify this option, the command skips redundant via insertion on all nets with a worst hold slack less than or equal to this value.

DESCRIPTION

This command replaces one set of vias in the design with another set. By default, all single-cut vias are replaced with 2-cut via arrays. You can change the behavior by using the **define_zrt_redundant_vias** command to specify the exact via mapping to be used by this command. From a netlist connectivity point of view, using a single-cut via is no different from using a multiple-cut via array, so additional via cuts in a multiple-cut via array are regarded as redundant vias from this point of view.

However, redundant vias can increase yield during manufacturing and are an important design-for-manufacturing (DFM) feature.

The vias are replaced only when the new via or via array does not introduce any design rule violations.

Multicorner-Multimode Support

This command uses information from active scenarios only.

EXAMPLES

The following example replaces all single cut vias by 2-cut via arrays.

```
prompt> insert_zrt_redundant_vias
```

The following example replaces vias named VIA23 and VIA34 with vias named VIA23 and VIA34 that have corresponding x-sizes of 1 and 5 and y-sizes of 3 and 1.

```
prompt> define_zrt_redundant_vias -from_via {VIA23 VIA34} \  
-to_via {VIA23 VIA34} -to_via_x_size {1 5} -to_via_y_size {3 1}  
prompt> insert_zrt_redundant_vias
```

The following example replaces a via named VIA23 with a via named VIA23 that has a corresponding x-size of 1 and a y-size of 3.

```
prompt> define_zrt_redundant_vias -from_via VIA23 -to_via VIA23 \  
-to_via_x_size 1 -to_via_y_size 3  
prompt> insert_zrt_redundant_vias
```

SEE ALSO

`define_zrt_redundant_vias(2)`

is_false

Tests the value of a specified variable, and returns a 1 if the value is 0 or the case-insensitive string *false*; returns a 0 if the value is 1 or the case-insensitive string *true*.

SYNTAX

```
int is_false  
value
```

Data Types

```
value      string
```

ARGUMENTS

```
value
```

Specifies the name of the variable whose value is to be tested.

DESCRIPTION

Tests the value of a specified variable, and returns a 1 if the value is either 0 or the case-insensitive string *false*. Returns a 0 if the value is either 1 or the case-insensitive string *true*. Any value other than 1, 0, *true*, or *false* generates a Tcl error message.

The **is_false** command is provided to use in writing scripts that test Boolean variables that can be set to 1, 0, or the case-insensitive strings *true* or *false*. When such variables are set to *true* or *false*, they cannot be tested in the negative in an **if** statement by simple variable substitution, because they do not constitute a true or false condition. The following example is not legal Tcl:

```
set x FALSE  
if {!$x} { set y TRUE  
}
```

This results in a Tcl error message, indicating that you cannot use a non-numeric string as the operand of "!". So, although you can test the positive condition, **is_false** allows you to test both conditions safely.

EXAMPLES

The following example shows the use of the **is_false** command.

```
prompt> set x TRUE  
TRUE
```

```
is_false  
1150
```

```
prompt> if {![is_false $x]} {  
?      set y TRUE  
?      }  
TRUE  
prompt>
```

SEE ALSO

[is_true\(2\)](#)

is_true

Tests the value of a specified variable, and returns a 1 if the value is 1 or the case-insensitive string *true*; returns a 0 if the value is 0 or the case-insensitive string *false*.

SYNTAX

```
int is_true  
value
```

Data Types

```
value      string
```

ARGUMENTS

```
value
```

Specifies the name of the variable whose value is to be tested.

DESCRIPTION

Tests the value of a specified variable, and returns a 1 if the value is either 1 or the case-insensitive string *true*. Returns a 0 if the value is either 0 or the case-insensitive string *false*. Any value other than 1, 0, *true*, or *false* generates a Tcl error message.

The **is_true** command is provided to use in writing scripts that test Boolean variables that can be set to 1, 0, or the case-insensitive strings *true* or *false*. When such variables are set to *true* or *false*, they cannot be tested in the negative in an **if** statement by simple variable substitution, because they do not constitute a true or false condition. The following example is not legal Tcl:

```
set x TRUE  
if {!$x} { set y FALSE  
}
```

This results in a Tcl error message, indicating that you cannot use a non-numeric string as the operand of "!". So, although you can test the positive condition, **is_true** allows you to test both conditions safely.

EXAMPLES

The following example shows the use of the **is_true** command.

```
prompt> set x FALSE  
FALSE  
prompt> if {![is_true $x]} {  
?      set y FALSE
```

```
?      }
```

```
FALSE  
prompt>
```

SEE ALSO

```
is_false(2)
```

is_zrt_routed_design

Checks if the current design is routed by Zroute.

SYNTAX

```
int is_zrt_routed_design
```

DESCRIPTION

This command checks if the current design is routed by Zroute. The top-level design must be open before you can run this command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command checks if the current design is routed by Zroute.

```
prompt> is_zrt_routed_design
```


legalize_fp_placement

Resolves standard cell placement conflicts after performing an initial placement.

SYNTAX

status **legalize_fp_placement**

ARGUMENTS

The **legalize_fp_placement** command has no arguments.

DESCRIPTION

Resolves std cell placement conflicts after performing an initial placement. Overlaps are removed and std cells are placed at legal sites. This command will legalize the entire design and is meant for use during the floorplanning stage of design. For final implementation, use `legalize_placement` command instead.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

SEE ALSO

`legalize_placement(2)`

legalize_placement

Executes detailed placement on a design.

SYNTAX

```
status legalize_placement
[-effort low | medium | high]
[-check_only]
[-eco]
[-incremental]
[-cells cell_objects]
[-priority low | medium | high]
[-timing]
[-coordinates {llx1 lly1 urx1 ury1 ...}]
```

ARGUMENTS

- effort low | medium | high**
Specifies the effort level for detailed placement. The default effort level is *medium*.
- check_only**
Checks whether the design and the libraries have all the necessary information to run the command.
- eco**
Derives locations for unplaced cells in the netlist. These cells are legally placed.
- incremental**
This option specifies incremental legalization. This option is mainly used at the situation that most cells have already been legalized. This option will try to move as fewer cells as possible (but may be larger distance). A suitable situation to use this option is in post-route optimization. This option can also minimize the timing critical cell displacement if it is used combined with **-timing** option.
- cells *cell_objects***
This option will legalize a group of cells which have been specified by this option. This option will need to be used combined with the option **-priority**. If **-priority low** is specified, legalizer will legalize the cells without moving other existing cells. If **-priority high** is specified, the legalizer will try to move the cells in minimum distance by moving other existing cell away. If **-priority med** is specified, the legalizer will move both these cells and their surrounding cells.
- priority low | medium | high**
This option is only used with **-cells** option together. This option specify the priority of a group of cells when this group of cells are being legalized. The default value is "low", i.e. legalizer will legalize the group of cells without moving other surrounding cells. If **-priority "med"** is specified, legalizer will move this group of cells and their surrounding cells when they

are being legalized. At this time, the cell movement will depend on their cell weights (cells are weighted by the worst pin slack as well as cell size). If `-priority "high"` is specified, legalizer will first snap these cells into their nearest legal locations, then move other cells at those locations to other legal locations.

-timing

This option is only used with `-cells` or `-incremental` option. This option will control the incremental legalization move less distance for the cells on critical paths, i.e. the cells are weighted by the worst pin slack as well as cell size. The incremental legalization will weight a cell only based on cell size if `-timing` is not specified.

-coordinates {llx1 lly1 urx1 ury1 ...}

Only used with `-cells` option. Specify the rectilinear region (multiple rectangles) so that the command only legalize the cells within the region. No cell will be moved outside the region, works like a fence to prevent cells move out of the region. The coordinate is the float_list of {x1 y1 x2 y2 ...}, unit of um.

DESCRIPTION

The **legalize_placement** command performs placement legalization on the current design, where coarse placement has already been performed. If no `-eco` specified, it is necessary to supply initial locations for every cell and blockage in the design before legalize. In addition, a **physical library** needs to be set using the `physical_library` variable. It is also necessary to specify a site array for the floorplan.

Also note that the option `-timing` can only be used together with `-cells` or `-incremental` option.

Note: After executing `legalize_placement`, all the timings are based on steiner routes of the design and any previous annotations are ignored.

Multicorner-Multimode Support

This command uses information from all active scenarios.

EXAMPLES

The following example runs the **legalize_placement** command.

```
prompt> legalize_placement -effort high
```

SEE ALSO

`current_design(2)`
`create_placement(2)`

lib2saif

Creates a forward-annotation SAIF file for a specified technology library.

SYNTAX

```
int lib2saif  
[-output file_name]  
library  
[-lib_pathname lib_path_name]
```

Data Types

<i>file_name</i>	string
<i>library</i>	list
<i>lib_path_name</i>	list

ARGUMENTS

-output *file_name*
Specifies the name of the library forward-annotation SAIF file created by **lib2saif**. The default file name is the value of the variable **power_sdpd_saif_file**. By default the value of the **power_sdpd_saif_file** variable is *power_sdpd.saif*.

library
Specifies the library name, or library file name for which the forward SAIF file is generated. If a library file name is specified then the library must be in .db format. The library must have state dependent leakage power characterization and/or pin-level state and/or path dependent internal power characterization for a library forward SAIF file to be generated.

-lib_pathname *lib_path_name*
Specifies the pathname to the simulation library information. This is optional, and is only required if the library is not in the simulation tool's current path during simulation.

DESCRIPTION

Creates a forward-annotation SAIF file that contains the state-dependent and path-dependent information for library cells. The library forward-annotation SAIF file is used in flows where a gate-level back-annotation SAIF file with state and/or path dependent switching activity is generated.

For details on the SAIF format, see the SAIF specification.

For details on state and path dependent power characterization, see the *Library Compiler Reference Manual* and the *Power Compiler Reference Manual*.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following examples illustrate the use of this command.

```
prompt> lib2saif -out a130.saif asic130_typical
```

```
prompt> lib2saif -out a130.saif ../libs/asic130.db
```

SEE ALSO

```
report_power(2)  
set_switching_activity(2)  
power_sdpd_saif_file(3)
```

license_users

Lists the current users of the Synopsys licensed features.

SYNTAX

```
status license_users  
[feature_list]
```

Data Types

feature_list list

ARGUMENTS

feature_list
Lists the licensed features for which to obtain the information. If more than one feature is specified, they must be enclosed in braces ({}). See the Synopsys *Installation Guide: UNIX-Based Platforms* for a list of features supported by the current release, or determine from the key file all of the features that are licensed at your site.

DESCRIPTION

Displays information about all of the licenses, related users, and host names currently in use. If a feature list is specified, only information about those features is displayed.

The **license_users** command is valid only when Network Licensing is enabled.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

In this example, all of the users of the Synopsys features are displayed:

```
prompt> license_users  
  
krig@node1      Design-Analyzer, Design-Compiler, LSI-Interface  
                Test-Compiler, VHDL-Compiler  
doris@node2     HDL-Compiler, Library-Compiler  
test@node3      Design-Compiler, Design-Analyzer, TDL-Interface  
  
3 users listed.
```

This example shows the users of the "Library-Compiler" or "VHDL-Compiler" feature.

```
prompt> license_users {Library-Compiler VHDL-Compiler}
```

```
krig@node1      Design-Analyzer, Design-Compiler, LSI-Interface  
                Test-Compiler, VHDL-Compiler  
doris@node2     HDL-Compiler, Library-Compiler
```

```
2 users listed.
```

SEE ALSO

```
get_license(2)  
remove_license(2)
```

link

Resolves design references.

SYNTAX

```
int link  
[-force]  
[-remove_sub_design]
```

ARGUMENTS

-force

This is icc_shell only option. Without this option **link** command will not do anything in icc_shell. This option will force **link** command to resolve design references.

DESCRIPTION

Performs a name-based resolution of design references for the current design. For a design to be complete, it needs to be connected to all of the library components and designs it references. The references must be located and linked to the current design in order for the design to be functional. The purpose of this command is to locate all of the designs and library components referenced in the current design and connect (link) them to the current design.

The **link** command uses the system variables, **link_library** and **search_path**, and the **local_link_library** design attribute to resolve design references. The **local_link_library** and **link_library** variables specify a list of design and library files. A "*" entry in the value of the **link_library** variable indicates that **link** should search all the designs loaded in dc_shell. If the **link_library** has no "*" entry, the designs loaded in dc_shell are not searched. (The default for **link_library** is {your_library.db}.) The **search_path** variable specifies a list of directory names. When using this option, place the asterisk in quotes ("*").

If the reference is to a parameterized design (and the design is not already in memory), **link** automatically builds the template with the specified parameters. (Parameterized designs can only be specified using HDL code.)

If a **local_link_library** is set on the current design, it is added to the beginning of the **link_library** before the **link_library** is searched. The **link** command looks for the design references in the files specified first, by **local_link_library**, then by **link_library**. Simple filenames (those not containing directory or pathnames) are searched for in the directories specified in **search_path**. If the referenced design is not found in the link libraries, it looks for the referenced design in the **search_path** directories.

The order of directories in the **search_path** and of the files in the link libraries is important. Search order during a link is

1. local_link_library
2. link_library

3. `search_path`

The first occurrence of a design reference is used.

Multicorner-Multimode Support

This command uses information from active scenarios only.

EXAMPLES

In all modes, if design "top" refers to design "test" in its implementation, **link** must find and connect the design "test" to this reference. The **link** command first searches the files specified in the link libraries for "test". If it does not find "test" in the link libraries, it searches the directories specified in the **search_path** variable for a file named "test.db". If it does not find it there, a warning is issued stating that the reference cannot be resolved.

Assume that the **search_path**, **local_link_library**, and **link_library** variables for the previous example are set as follows:

```
prompt> search_path = {. synopsys_root + "/libraries" ~bill/project}  
{./, /usr/synopsys/libraries, /usr/bill/project}
```

```
prompt> set_local_link_library tech1_lib.db  
Setting local link library 'tech1_lib.db' on design 'top'
```

```
prompt> link_library = {"*" tech2_lib.db ./project.db}  
{"*" tech2_lib.db, ./project.db}
```

To find "test", the following sequence is performed:

Step 1.

Read the file "tech1_lib.db". Since this is a simple filename, the **search_path** is queried for the file location.

Step 2.

Search all the designs already loaded in dc_shell for a design called "test".

Step 3.

Read the file "tech2_lib.db". Since this is a simple filename, the **search_path** is queried for the file location.

Step 4.

Read the file "project.db" from the current directory. Since this is a complex filename (the current directory is specified), the **search_path** is not used to find the file.

Step 5.

Look for "test" in the list of designs contained in "tech1_lib.db", "tech2_lib.db", and "project.db".

Step 6.

If not found in Step 5, search for the file "test.db" using the **search_path**.

The following contains examples of the **link** command for all modes.

Example 1

The **search_path** variable is set to a list of directories where designs and technology libraries reside. In the following example, "top", "counter", and "controller" designs are built using components from the **tech_lib** library. You need to read in only the top-level design. **link** automatically loads designs from both **link_library** files and, as needed, files in the **search_path**.

```
prompt> search_path = { . synopsys_root + "/libraries" /usr/bill}
{ ., synopsys_root + "/libraries", /usr/bill}

prompt> read top.db
Loading db file '/usr/bill/top.db'
{top}

prompt> link_library = {"*" tech_lib.db}
{"*", tech_lib.db}

prompt> link
Loading db file '/usr/synopsys/libraries/tech_lib.db'
Linking design:
    top
Using the following designs and libraries:
    top, tech_lib (library)
Loading db file '/usr/bill/controller.db'
Loading db file '/usr/bill/counter.db'
```

Example 2

The following example shows that reading a design into dc_shell has no effect on the **link** command if the "*" is missing from the **link_library** variable. The design "adder" is read into the dc_shell explicitly, but cannot be found on the **search_path** or in the **link_library**. Therefore, it is unresolved during **link**.

The following example sets the **search_path** without including the directory of the file "adder.db":

```
prompt> search_path = { . synopsys_root + "/libraries" /usr/bill}
{ ., /usr/synopsys/libraries, /usr/bill}
```

The following example reads design "adder" into dc_shell:

```
prompt> read /usr/bill/dc/adder.db
Loading db file '/usr/bill/dc/adder.db'
{adder, mux81}
```

The following example sets the **link_library** without including the file "adder.db" or "*":

```
prompt> link_library = {tech_lib.db}
{tech_lib.db}
```

The following example sets the **current_design** variable:

```
prompt> current_design = top  
top
```

The following example shows that **link** fails for designs in "adder.db":

```
prompt> link  
Loading db file '/usr/synopsys/libraries/tech_lib.db'  
Linking design:  
    top  
Using the following designs and libraries:  
    tech_lib (library)  
Warning: Unable to resolve reference 'adder' in design 'top'  
Warning: Unable to resolve reference 'mux81' in design 'top'
```

The following example adds "*" to the **link_library** and the design links successfully:

```
prompt> link_library = "*" + link_library  
{ "*", tech_lib.db}  
prompt> link  
Loading db file '/usr/synopsys/libraries/tech_lib.db'  
Linking design:  
    top  
Using the following designs and libraries:  
    top, tech_lib (library)  
Loading db file '/usr/bill/controller.db'  
Loading db file '/usr/bill/counter.db'
```

The following example adds "adder.db" to the **link_library**, which is another way to link successfully:

```
prompt> link_library = {tech_lib.db, /usr/bill/dc/adder.db}  
{tech_lib.db, /usr/bill/dc/adder.db}  
prompt> link  
Loading db file '/usr/synopsys/libraries/tech_lib.db'  
Linking design:  
    top  
Using the following designs and libraries:  
    tech_lib (library), /usr/bill/dc/adder.db  
Loading db file '/usr/bill/controller.db'  
Loading db file '/usr/bill/counter.db'
```

Example 3

This example shows how to handle files with multiple designs. If you need to link with a file of multiple designs, put that file in your **link_library**.

If a design is not found in the **link_library** files, **link** searches the **search_path** directories only for a file of the same name as the referenced design. If your multiple-design file has a different file name, it will not be found, even if it is located on the **search_path**.

```
prompt> search_path = {. synopsys_root + "/libraries" /usr/project}  
{., /usr/synopsys/libraries, /usr/project}
```

The following example sets the **link_library** without including "mylib.db":

```
prompt> link_library = {tech_lib.db}  
{tech_lib.db}
```

The link fails for designs in "mylib" that are neither in **link_library** files nor found on the **search_path**.

```
prompt> link  
Loading db file '/usr/synopsys/libraries/tech_lib.db'  
Linking design:  
    top  
Using the following designs and libraries:  
    tech_lib (library)  
Warning: Unable to resolve reference 'adder' in design 'top'  
Warning: Unable to resolve reference 'mux81' in design 'top'  
Warning: Unable to resolve reference 'counter' in design 'top'
```

Because some of the designs are in "mylib.db", adding this file to the **link_library** enables a successful link:

```
prompt> link_library = {tech_lib.db mylib.db}  
{tech_lib.db, mylib.db}  
prompt> link  
Loading db file '/usr/synopsys/libraries/tech_lib.db'  
Linking design:  
    top  
Using the following designs and libraries:  
    tech_lib (library), mylib, adder  
Loading db file '/usr/bill/controller.db'  
Loading db file '/usr/bill/counter.db'
```

Instead, add "*" to the **link_library** and the design links successfully:

```
prompt> link_library = "*" + link_library  
{"*", tech_lib.db}  
prompt> link  
Loading db file '/usr/synopsys/libraries/tech_lib.db'  
Linking design:  
    top  
Using the following designs and libraries:  
    mylib, adder, tech_lib (library)  
Loading db file '/usr/bill/controller.db'  
Loading db file '/usr/bill/counter.db'
```

SEE ALSO

```
current_design(2)  
set_local_link_library(2)  
which(2)  
auto_link_options(3)  
link_library(3)
```

`search_path(3)`

link_physical_library

Links the physical library with the logical library for physical synthesis.

SYNTAX

```
int link_physical_library
```

ARGUMENTS

There are no arguments to this command.

DESCRIPTION

The **link_physical_library** command links the logical library and physical library, linking library cells and pins by name. Normally, any physopt command will implicitly do the linking for you. If you want to explicitly link the libraries, use this command. If the libraries are already linked, this command will do nothing.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The **search_path** variable is set to a list of directories where designs and technology libraries reside. The following example shows how to link the logical library with the physical library.

```
prompt> set search_path = ". synopsys_root/libraries /usr/bill"
prompt> set target_library = mylib.db
prompt> set physical_library = mylib.pdb
prompt> read_db mydesign.db
prompt> link_physical_library
```

SEE ALSO

```
search_path(3)
target_library(3)
```

list_attributes

Lists the currently defined attributes.

SYNTAX

```
string list_attributes  
[-application]  
[-class class_name]  
[-nosplit]
```

Data Types

class_name string

ARGUMENTS

-application

Lists application attributes and user-defined attributes.

-class *class_name*

Limits the listing to attributes of a single class. Valid classes are **bound**, **cell**, **cell_site**, **clock**, **core_area**, **design**, **die_area**, **drc_error**, **group**, **gui_object**, **layer**, **library**, **lib_cell**, **lib_pin**, **mw_cel**, **net**, **physical_bus**, **physical_lib**, **physical_lib_cell**, **physical_lib_pin**, **pin**, **pin_guide**, **pin_shape**, **placement_blockage**, **plan_group**, **port**, **power_domain**, **power_switch**, **region**, **route_guide**, **rp_group**, **rp_group_instance**, **rp_group_keepout**, **shape**, **site_row**, **supply_net**, **supply_port**, **terminal**, **text**, **track**, **via**, and **voltage_area**.

DESCRIPTION

The **list_attributes** command displays an alphabetically sorted list of currently defined attributes. The attributes are divided into two categories: application-defined and user-defined. By default, **list_attributes** lists all user-defined attributes.

Using the **-application** option adds all application attributes to the listing. Since there are many application attributes, you can limit the listing to a specific object class by using the **-class** option.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following is an example listing of some attributes defined with the **define_user_attribute** command:

prompt> **list_attributes**

Report : List of Attribute Definitions

Design :

Properties:

A - Application-defined

U - User-defined

I - Importable from design/library (for user-defined)

Attribute Name	Object	Type	Properties	Constraints
area_coefficient	library	float	U	
capacitive_load_units	library	float	U	
current_unit	library	int	U	
in_place_swap_mode	library	int	U	
leakage_power_unit	library	int	U	
library_compiler_version	library	string	U	
library_features	library	int	U	
power_unit	library	int	U	
resistance_unit	library	int	U	
time_scale	library	int	U	
timing_model_type	lib_cell	string	U	
voltage_unit	library	int	U	

The following example adds application-defined attributes, but limits the listing to net attributes only:

prompt> **list_attributes -application -class net**

Report : List of Attribute Definitions

Design :

Properties:

A - Application-defined

U - User-defined

I - Importable from design/library (for user-defined)

Attribute Name	Object	Type	Properties	Constraints
actual_fall_transition_max				
	net	float	A	
actual_fall_transition_min				
	net	float	A	
actual_rise_transition_max				
	net	float	A	
actual_rise_transition_min				
	net	float	A	
wire_capacitance_min	net	float	A	
x_length	net	int	A	
y_length	net	int	A	

list_attributes

SEE ALSO

`define_user_attribute(2)`
`get_attribute(2)`
`remove_attribute(2)`
`report_attribute(2)`
`set_attribute(2)`

list_designs

List the designs available in memory.

SYNTAX

```
int list_designs  
[design_list]  
[-show_file]
```

Data Types

design_list list

ARGUMENTS

design_list
Specifies the designs to list. If this option is not specified, all designs are listed. When specifying designs, the wildcard characters * (asterisk) and ? (question mark) are supported. For details about using and escaping wildcard characters, refer to the **wildcards** man page.

-show_file
Specifies that designs are shown according to the file in which they reside. The **-show_file** option is typically used when two or more designs share the same name.

DESCRIPTION

The **list_designs** command displays the designs currently read in dc_shell. In the design listing, the current design is designated by an asterisk.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses **list_designs**.

```
prompt> list_designs  
ADDER                FULL_SUBTRACTOR      HALF_SUBTRACTOR      TOP  
FULL_ADDER          HALF_ADDER           SUBTRACTOR  
  
prompt> current_design TOP  
Current design is 'TOP'.
```

The following example shows that the current design is denoted with an asterisk:

```
prompt> list_designs
ADDER          FULL_SUBTRACTOR    HALF_SUBTRACTOR    TOP (*)
FULL_ADDER     HALF_ADDER          SUBTRACTOR
```

The following example lists all designs that end with `_ADDER`:

```
prompt> list_designs "*"_ADDER"
HALF_ADDER     FULL_ADDER
```

The following example uses `-show_file` to differentiate between designs having the same name:

```
prompt> list_designs
ADDER          FULL_SUBTRACTOR    HALF_SUBTRACTOR    TOP
FULL_ADDER     HALF_ADDER          SUBTRACTOR

prompt> copy_design TOP example.db:TOP
Copying design 'TOP' to 'example.db:TOP'

prompt> list_designs
ADDER          FULL_SUBTRACTOR    HALF_SUBTRACTOR    TOP
FULL_ADDER     HALF_ADDER          SUBTRACTOR          TOP (*)

prompt> list_designs -show_file
/usr/users/bill/test/designs/top.db
ADDER          FULL_SUBTRACTOR    HALF_SUBTRACTOR    TOP (*)
FULL_ADDER     HALF_ADDER          SUBTRACTOR

example.db
TOP
```

SEE ALSO

```
current_design(2)
list_instances(2)
wildcards(3)
```

list_drc_error_types

Lists error type names

SYNTAX

```
list list_drc_error_types
[-error_view mw_error_view]
[-class type_class]
[-name type_name]
[-id]
```

Data Types

<i>mw_error_view</i>	list collection
<i>type_class</i>	list
<i>type_name</i>	list

ARGUMENTS

-error_view *mw_error_view*
Specify the error view on which to list the error types. If omitted, the error types from the current top-level design cell are listed. Specifying more than one error view causes an error.

-class *type_class*
One or more of the type classes {Default, Open, OpenLocator, Short, Spacing}. Type class name keywords are case insensitive. Only the error types in the specified classes are listed. If omitted, all types are listed.

-name *type_name*
One or more type names to list. This option is useful only if you are listing type ids. Type names are case sensitive. Only the error types with the given names are listed.

-id
If this Boolean option is given, type ids, rather than type names, are listed. While it is not possible to create multiple types with the same name using the public Tcl API, there are existing error view writers which create error views with multiple error types sharing the same name. This command along with the **-name** and the **-id** option can be used to list the ids sharing type names.

DESCRIPTION

This command returns a list of error types in the given error view by name or id.

SEE ALSO

```
create_drc_error_type(2)
report_drc_error_type(2)
```

list_files

Lists the files that are loaded into memory.

SYNTAX

integer **list_files**

ARGUMENTS

The **list_files** command has no arguments.

DESCRIPTION

The **list_files** command lists the files loaded in dc_shell.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows the output from **list_files**:

```
prompt> list_files
```

File	Path
----	----
adder.db	/remote/usr4/joeuser/designs
clock.db	/remote/usr4/joeuser/designs
gtech.db	/remote/usr4/joeuser/libraries
lsi_10k.db	/remote/usr4/joeuser/libraries

SEE ALSO

list_designs(2)
list_licenses(2)

list_floorplan_data

Lists the currently defined floorplan data attribute names.

SYNTAX

```
string list_floorplan_data  
[-module top | child_cell | plan_group]  
[-view one_level | logical | physical]
```

ARGUMENTS

-module top | child_cell | plan_group
Specifies the module type on which the floorplan data attribute names are listed.
The default value of this option is **top**.

-view one_level | logical | physical
Specifies the view in which floorplan data attribute names are listed.
When you specify **one_level** (the default), the data attribute names are listed if they are defined in the first level of hierarchy in the module.
When you specify **logical** or **physical**, the data attribute names are listed if they are defined in any level of hierarchy in the module. The difference between **logical** and **physical** is that soft macros are traversed in the physical view, but not in the logical view.
This option is ignored if you specify **-module plan_group**, because plan groups have only a logical view.

DESCRIPTION

The **list_floorplan_data** command displays an alphabetically sorted list of currently defined floorplan data attribute names. To query the value of a data attribute on a specified object, use the **get_floorplan_data** command.

For more information about floorplan data attributes, see the **floorplan_data_attributes** man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example lists the names of all floorplan data attributes that have been defined on the top module in the the one-level view, and then gets the physical_area value.

```
prompt> list_floorplan_data -module top -view one_level  
hard_macro_area_percentage  
macro_area_percentage  
number_of_black_box
```

```
number_of_hard_macro
number_of_io_cell
number_of_macro
number_of_standard_cell
physical_area
```

```
prompt> get_floorplan_data -module [current_mw_cel] \
-view one_level physical_area
388.815300
```

The following example lists the names of all floorplan data attributes that have been defined on the top module in the logical view and gets the physical_area and area values. The example also tries to get the area value from the one-level view, but cannot because it does not exist.

```
prompt> list_floorplan_data -module top -view logical
area
aspect_ratio
hard_macro_area_percentage
macro_area_percentage
number_of_black_box
number_of_hard_macro
number_of_io_cell
number_of_macro
number_of_standard_cell
physical_area
```

```
prompt> get_floorplan_data -module [current_mw_cel] \
-view logical physical_area
2084312.873500
```

```
prompt> get_floorplan_data -module [current_mw_cel] \
-view logical area
5707320.000000
```

```
prompt> get_floorplan_data -module [current_mw_cel] \
-view one_level area
Error: Data name area for class mw_cel in view one_level is invalid. (MWUI-044)
```

SEE ALSO

```
get_floorplan_data(2)
report_floorplan_data(2)
floorplan_data_attributes(3)
```

list_instances

Lists the instances in the current design or current instance.

SYNTAX

```
int list_instances  
[instance_list]  
[-hierarchy]  
[-max_levels num_levels]  
[-full]
```

Data Types

<i>instance_list</i>	list
<i>num_levels</i>	integer

ARGUMENTS

instance_list
Specifies the instances to list. By default, all instances in the current design or current instance are listed.

-hierarchy
Lists all levels of instance hierarchy. By default, only the current level of hierarchy is listed.

-max_levels *num_levels*
Modifies the **-hierarchy** option by specifying a limit to the number of levels of hierarchy listed, so the **-max_levels** option is significant only if used with the **-hierarchy** option.

-full
Displays the full hierarchy. By default, if there is a submodule in multiple locations in a hierarchy, its components are listed only once, with an ellipsis (...) indicating the contents of a previously displayed module.

DESCRIPTION

The **list_instances** command lists the instances in memory. The command displays cells at the current level of hierarchy, as specified by **current_design** and **current_instance**.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses **list_instances** to display the available instances. The

```
list_instances  
1178
```


design that each instance references is shown in parentheses following the instance name.

```
prompt> current_design TOP
Current design is 'TOP'.

prompt> list_instances
U1 (ADDER)          U2 (SUBTRACTOR)

prompt> current_instance U1
Current instance is 'TOP/U1'.

prompt> list_instances
U1 (FULL_ADDER)      U2 (FULL_ADDER)
U3 (FULL_ADDER)      U4 (FULL_ADDER)

prompt> list_instances {U1, U2}
U1 (FULL_ADDER)      U2 (FULL_ADDER)

prompt> current_instance U2
"TOP/U1/U2"

prompt> list_instances "*"
U1 (HALF_ADDER)      U2 (HALF_ADDER)      U3 (OR2)
```

The following examples use **-hierarchy** to display multiple levels of the instance hierarchy. All instances at each level of hierarchy are listed as a group.

```
prompt> current_instance
Current instance is the top level of design 'TOP'.

prompt> list_instances -hierarchy
U1 (ADDER)
  U1 (FULL_ADDER)
    U1 (HALF_ADDER)
      U1 (EO)          U2 (AN2)
    U2 (HALF_ADDER)...
    U3 (OR2)
  U2 (FULL_ADDER)...
  U3 (FULL_ADDER)...
  U4 (FULL_ADDER)...
U2 (SUBTRACTOR)
  U1 (FULL_SUBTRACTOR)
    U1 (HALF_SUBTRACTOR)
      U1 (EO)          U2 (IV)          U3 (AN2)
    U2 (HALF_SUBTRACTOR)...
    U3 (OR2)...
  U2 (FULL_SUBTRACTOR)...
  U3 (FULL_SUBTRACTOR)...
  U4 (FULL_SUBTRACTOR)...

prompt> list_instances -hierarchy -max_levels 2
U1 (ADDER)
```

```

U1 (FULL_ADDER)...
U2 (FULL_ADDER)...
U3 (FULL_ADDER)...
U4 (FULL_ADDER)...
U2 (SUBTRACTOR)
  U1 (FULL_SUBTRACTOR)...
  U2 (FULL_SUBTRACTOR)...
  U3 (FULL_SUBTRACTOR)...
  U4 (FULL_SUBTRACTOR)...

prompt> current_instance U1/U1
Current instance is 'TOP/U1/U1'.

```

```

prompt> list_instances -hierarchy
U1 (HALF_ADDER)
  U1 (EO)          U2 (AN2)
U2 (HALF_ADDER)...
U3 (OR2)

```

```

prompt> list_instances -hierarchy -full
U1 (HALF_ADDER)
  U1 (EO)          U2 (AN2)
U2 (HALF_ADDER)
  U1 (EO)          U2 (AN2)
U3 (OR2)

```

SEE ALSO

```

current_design(2)
current_instance(2)
list_designs(2)
wildcards(3)

```

list_libs

Lists the libraries available in memory.

SYNTAX

```
status list_libs
[lib_list]
```

Data Types

```
lib_list      list
```

ARGUMENTS

lib_list
Specifies the libraries to list. You can use the * (asterisk) and ? (question mark) wildcard characters when specifying the libraries to list. For more details about using and escaping wildcards see the man page for the **wildcards** variable. By default, the tool lists all libraries.

DESCRIPTION

This command displays all libraries that are currently available in memory.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

This example shows a simple use of the **list_libs** command. The library flagged with M is the maximum library used for maximum delay analysis. It corresponds to the library flagged with m used for minimum delay analysis.

```
prompt> set_min_library sc_max.db -min sc_min.db
prompt> set_min_library tt_max.db -min tt_min.db
prompt> list_libs
Logical Libraries:
```

Library	File	Path
-----	----	----
cell	cells.db	/remote/usr4/joeuser/designs
gtech	gtech.db	/remote/usr4/joeuser/libraries
standard.sldb	standard.sldb	/remote/usr4/joeuser/libraries
M fs120_max	sc_max.db	/remote/usr4/joeuser/libraries
m fs120_min	sc_min.db	/remote/usr4/joeuser/libraries
M cb_max	tt_max.db	/remote/usr4/joeuser/libraries
m cb_min	tt_min.db	/remote/usr4/joeuser/libraries

SEE ALSO

`list_designs(2)`
`list_instances(2)`
`read_lib(2)`
`report_lib(2)`
`set_min_library(2)`
`wildcards(3)`

list_licenses

Displays a list of licenses currently checked out by the user.

SYNTAX

```
status list_licenses
```

ARGUMENTS

The **list_licenses** command has no arguments.

DESCRIPTION

The **list_licenses** command lists the licenses you currently have checked out. If more than one copy of a license key is checked out then the number of keys checked out is shown in parentheses following the license name.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows the output from **list_licenses**.

```
prompt> list_licenses
```

```
License in use:  
    Galaxy-Common  
    Galaxy-ICC  
1
```

SEE ALSO

```
check_license(2)  
get_license(2)  
license_users(2)  
remove_license(2)
```

list_mw_cels

Prints the list of Milkyway designs in the current Milkyway library.

SYNTAX

```
status list_mw_cels  
[-all_views]  
[-all_versions]  
[-sort]
```

ARGUMENTS

- all_views**
List all views of Milkyway designs in the current Milkyway library. If specifying this option without **-all_versions** option, the version number will not be listed.
- all_versions**
List all versions of Milkyway designs in the current Milkyway library. If specifying this option without **-all_views** option, the designs of CEL view will only be listed.
- sort**
Sort the list by alphabet.

DESCRIPTION

This command prints the list of Milkyway designs in the current Milkyway library. The library must be open. And note that it will show the design name if any view of the design exists in the library. It also prints the following information about the Milkyway design:

- **open** if it is an open Milkyway design.
- **read_only** if it is opened in read only mode.
- **current** if it is the currently active Milkyway design.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example lists the Milkyway designs in the current library:

```
list_mw_cels  
1184
```

```
prompt> list_mw_cels
top
AND
NOR
1
prompt> list_mw_cels -all_views
top.CEL
AND.CEL
NOR.CEL
NOR.FRAME
1
prompt> list_mw_cels -all_versions
top.CEL;1
top.CEL;2
AND.CEL;1
NOR.CEL;1
1
prompt> list_mw_cels -all_views -all_versions -sort
AND.CEL;1
NOR.CEL;1
NOR.FRAME;1
top.CEL;1
top.CEL;2
1
```

SEE ALSO

get_mw_cels(2)

list_partition_data

Queries the partition editor and returns data from it.

SYNTAX

```
status list_partition_data
[-top_blocks]
[-IOs]
[-hard_macros]
[-missing_modules]
[-top_hard_macros]
[-sub_blocks]
[-area cells_only | boundary | include_external_keepout]
[-power_ports]
[-ground_ports]
[-
instance_count all | std_cells | hard_macros | IOs | modules | missing | black_box]
[-keepouts]
[-utilization]
[-tcl_output]
[-verbose]
[module_names]
```

Data Types

module_names string list

ARGUMENTS

- top_blocks**
Returns the top blocks from the partition. Top blocks are blocks that have a module definition but for which there are no instances in any other module. Typically, there is a single top block, which is the root of the design. You can easily end up with multiple top blocks if, for example, the top module itself is missing.
- IOs**
Returns the I/O cells and the count of each I/O cell in the specified modules.
- hard_macros**
Returns the hard macro cells and the count of each hard macro cell in the specified modules.
- missing_modules**
Returns the missing modules in the specified modules. Missing modules are those modules that have no definition and no area in Verilog.
- top_hard_macros**
Returns the hard macros that reside at the top level of design.
- sub_blocks**
Returns the subblocks from the partition. Subblocks are modules that are

marked as physical except any top blocks. You can use this option to determine which blocks have become physical after some form of autopartitioning.

`-area cells_only | boundary | include_external_keepout`

Returns the specified area, in square microns, of the specified modules. Valid values for the **-area** option are

- **cells_only**

Includes only the area of the cells in the specified modules.

- **boundary**

Considers the utilization of the modules and also the internal keepouts set on the modules.

- **include_external_keepout**

Considers the boundary area and also the external keepouts set on the modules. You can use this information after partitioning to determine the area required for the physical blocks.

`-power_ports`

Returns the power ports of the specified modules. These are computed through an analysis of the leaf-cell library cells, so it does not work for missing modules. This is used to do a better job of determining the defaults for power network synthesis and analysis.

`-ground_ports`

Returns the ground ports of the specified modules. These are computed through an analysis of the leaf-cell library cells, so it does not work for a missing modules. This is used to do a better job of determining the defaults for power network synthesis and analysis.

`-instance_count all | std_cells | hard_macros | IOs | modules | missing | black_box`

Returns the specified instance count for the specified modules. This count is the total number of cells that are in the subhierarchy rooted at the specified modules.

Valid values for the **-instance_count** option are

- **all**

Considers all types of physical cells.

- **std_cells**

Considers only standard cells.

- **hard_macros**

Considers only hard macros.

- **IOs**

Considers only I/O cells.

- **modules**

Considers all modules.

- **missing**

Considers only missing modules. Missing modules are modules that have no definition and no area.

- **black_box**

Considers missing modules that have been assigned an area. The area is assigned by using the **set_partition_data** command.

-keepouts

Returns the external and internal keepouts for the specified modules. These keepouts can be set by the **set_partition_data** command. The default value is 0.

-utilization

Returns the utilization of the specified modules, expressed as a fraction between 0 and 1. If you have not modified the utilization values with the partition editor or by using the **set_partition_data** command, the values are the default value of 0.7 that was used when the partition was created.

-tcl_output

Outputs the results in Tcl format. See the example below for details.

-verbose

Outputs more detailed messages.

module_names

Specifies the modules from which the requested data is gathered and returned. By default, it is the top-level modules.

DESCRIPTION

This command is used to query a partition and return data from it. It is needed because much of the data on the browser is computed or derived from autopartitioning. This command is used mainly in flows to retrieve data from a partition to control subsequent steps within the flow. For example, after autopartitioning, you can use this command to return the set of subblocks within the partition as well as the area for each of the subblocks. The area can be used by subsequent top-down block shaping to keep the proper area for the subblock footprints.

If you run this command without specifying any options, the command returns the specified module names, or the top-level modules if you do not specify the module names.

EXAMPLES

The following example loads the partition editor and does an automatic partitioning, which creates the data to be queried by the **list_partition_data** command.

```
prompt> open_mw_lib myDesign.mw
{myDesign.mw}
prompt> read_partition -input_files {../ICC_DATA/Netlist/myDesign.v}
Partitioner Name: pfe0
Loading verilog file ../ICC_DATA/Netlist/myDesign.v
MYDESIGN: 73977 instances of 134 modules.
1
prompt> set_partition_data -auto_partition instance_count
```

list_partition_data

After partitioning, list of sub-blocks is BLENDER_0 BLENDER_1 BLENDER_2
BLENDER_3
1

The following example reports information about the subblocks in the partition after autopartitioning.

```
prompt> list_partition_data -sub_blocks  
BLENDER_0 BLENDER_1 BLENDER_2 BLENDER_3
```

The following example requests multiple pieces of data at the same time.

```
prompt> list_partition_data -instance_count all -area boundary \  
"BLENDER_0 PCI_CORE SDRAM_IF"  
module BLENDER_0  
    BoundaryArea: 333551 InstanceCount: 11039  
module PCI_CORE  
    BoundaryArea: 253270 InstanceCount: 7579  
module SDRAM_IF  
    BoundaryArea: 260616 InstanceCount: 8086
```

The following example returns the information in Tcl format.

```
prompt> list_partition_data -area boundary -instance_count all \  
-tcl_output "BLENDER_0 PCI_CORE SDRAM_IF"  
{  
    {BLENDER_0  
        {BoundaryArea 333551 InstanceCount 11039}}  
    {PCI_CORE  
        {BoundaryArea 253270 InstanceCount 7579}}  
    {SDRAM_IF  
        {BoundaryArea 260617 InstanceCount 8086}}  
}
```

Note that the brace indentation has been modified to make the example more readable. This version is more suited to storage within a Tcl variable for subsequent retrieval within Tcl.

SEE ALSO

```
create_partition(2)  
read_partition(2)  
set_partition_data(2)
```

lminus

Removes one or more named elements from a list and returns a new list.

SYNTAX

```
list lminus  
[-exact] the_list elements
```

Data Types

<i>the_list</i>	list
<i>elements</i>	list

ARGUMENTS

the_list
Specifies the list to copy and modify.

elements
Specifies a list of elements to remove from *the_list*.

DESCRIPTION

The **lminus** command removes elements from a list by using the element itself, rather than the index of the element in the list (as in **lreplace**). **lminus** uses the commands **lsearch** and **lreplace** to find the elements and replace them with nothing.

If none of the elements are found, a copy of *the_list* is returned.

The **lminus** command is often used in the translation of Design Compiler scripts that use the subtraction operator (-) to remove elements from a list.

EXAMPLES

The following example shows the use of the **lminus** command. Notice that no error message is issued if a specified element is not in the list.

```
prompt> set l1 {a b c}  
a b c  
prompt> set l2 [lminus $l1 {a b d}]  
c  
prompt> set l3 [lminus $l1 d]  
a b c
```

Example below illustrates the usage of *-exact* option with **lminus**.

```
prompt> set l1 {a a[1] a* b[1] b c}  
a a[1] a* b[1] b c  
prompt> set l2 [lminus $l1 a*]  
{b[1]} b c  
prompt> set l3 [lminus -exact $l1 a*]  
a {a[1]} {b[1]} b c
```

```
prompt> set 14 [lminus -exact $l1 {a[1] b[1]}]  
a a* b c
```

SEE ALSO

load_fp_rail_map

Loads a voltage (IR) drop, electromigration (EM), resistance map or instance based voltage (IR) drop, power or power density map for display.

SYNTAX

```
status load_fp_rail_map
[-nets net_name]
[-instance]
[-type map_type]
[-min min_value]
[-max max_value]
[-directory directory_name]
[-clear]
```

Data Types

<i>net_name</i>	string
<i>map_type</i>	string
<i>min_value</i>	float
<i>max_value</i>	float
<i>directory_name</i>	string

ARGUMENTS

-nets *net_name*

Enter the name of the power or ground net for which you want to load the rail analysis results for displaying the voltage drop map. Only a single net is allowed at a time. For example, if "-nets {VDD VSS}" is specified, the first VDD net is selected.

-instance

Use this option to display instance based voltage (IR) drop, power, or power density maps. This option is mutually exclusive with the **-nets** option. exclusive with "-nets" option.

-type *map_type*

Specify a keyword "IR, EM, R, P, PD or InstIR" for the type of map you want to load. By using this option, a voltage (IR) drop, electromigration (EM) violations rail, resistance (R), instance power (P), instance power density (PD), or instance IR drop (InstIR) will be loaded for display in the GUI. The default for a net based map is **IR**. The default for an instance based map is **InstIR**.

-min *min_value*

Enter a value for the minimum voltage drop limit in millivolts (mV). By default, it is automatically set based on power rail synthesis or analysis results from a specified directory. There is no limit for this value. Note that if "min_value" > "max_value", then both values are exchanged to display data correctly.

`-max max_value`

Enter a value for the maximum voltage drop limit in millivolts (mV). By default, it is automatically set based on power rail synthesis or analysis results from a specified directory. There is no limit for this value. Note that if "min_value" > "max_value", then both values are exchanged to display data correctly.

`-directory directory_name`

Specify the directory which stores IR/EM results with the file name *design_name.net_name.pw_hl.pna* and the power allocation results with the file name *design_name.net_name.power*. For an instance-based map, the file name is *design_name.inst_hl.pna*. The default is *./pna_output*.

`-clear`

Removes a map which was previously loaded. By default, this option is off.

DESCRIPTION

After rail analysis is complete, you can load a voltage drop (IR), electromigration (EM), or resistance (R) distribution in your design. You can also load instance based voltage drop (InstIR), instance power (P) or instance power density (PD) map for display through the GUI.

This command loads voltage drop values, EM values, resistance values, power or power density in the design to set increments between the maximum and minimum value limits that you specify.

The tool performs voltage drop analysis by retrieving the database objects that belong to the power and ground net to be analyzed and builds a resistive network through parasitic extraction. All cell instances are treated as equivalent current sources, and their values are derived from the power consumption values you specify. Using the current source values, the tool determines the voltage drop for each cell instance, and then it calculates the current flow and current density along each power and ground segment.

Multicorner-Multimode Support

This command is not supported in a multi-scenario design flow.

EXAMPLES

The following example shows how to load a IR map of a net VDD, with the minimum IR drop value **0.1**, the maximum IR drop value **100** and a directory which stores voltage (IR) drop results,

```
prompt> load_fp_rail_map \  
-nets VDD \  
-min 0.1 \  
-max 100 \  
-directory ./pna_output_2 \  

```

The following example shows how to load a resistance map of a net VDD.

```
prompt> load_fp_rail_map \  
-nets VDD \  
-type R \
```

The following example shows how to load a instance voltage drop map.

```
prompt> load_fp_rail_map \  
-instance \  
-type InstIR \
```

SEE ALSO

```
analyze_fp_rail(2)  
synthesize_fp_rail(2)
```


load_of

Returns the capacitance of the specified library cell pin.

SYNTAX

```
float load_of  
library_cell_pin
```

Data Types

```
library_cell_pin      string
```

ARGUMENTS

```
library_cell_pin  
    The name of the library cell pin whose capacitance is to be returned. The  
    format for library_cell_pin is similar to that of the find command:  
    library_name/lib_cell_name/pin_name. (The specified library must be already  
    loaded into dc_shell.)
```

DESCRIPTION

Returns the load of a given library cell pin. It is primarily used as an argument to the **set_load** command. If the specified *library_cell_pin* cannot be found, a warning message is issued and 0.0 is returned.

EXAMPLES

The following command returns the pin load of a lib_cell from the tech_lib.

```
prompt> load_of tech_lib/lib_cell/pin
```

The following example uses this command with the **set_load** command. It sets the load on all output ports to the value of the inverter cell input from the tech_lib.

```
prompt> set_load load_of( tech_lib/inverter/input) all_outputs()
```

SEE ALSO

```
set_load(2)
```

load_upf

Reads a script in the Unified Power Format (UPF). This command is supported only in UPF mode.

SYNTAX

```
status load_upf  
upf_file_name  
[-scope instance_name]  
[-noecho]
```

Data Types

<i>upf_file_name</i>	string
<i>instance_name</i>	string

ARGUMENTS

upf_file_name
Specifies the name of the file that contains the UPF script to be read.

-scope *instance_name*
Specifies the scope where the UPF commands contained in the specified *upf_file_name* are to be executed.

-noecho
Instructs **load_upf** not to echo the commands as the UPF file is loaded.

DESCRIPTION

The **load_upf** command sets the scope to the specified instance and executes the set of UPF commands in the specified UPF file. Upon completion, the current scope is restored to the value it was prior to the invocation of **load_upf**. When the **-scope** option is not specified, the current scope is used.

The **load_upf** command executes the constraints as they are being read. If errors are encountered when the UPF file is being read, it is possible that only some of the constraints are applied.

The **load_upf** command supports following UPF commands. Usage of some of these commands is restricted when reading the UPF file. In certain cases, some command options are not supported.

```
add_port_state  
add_pst_state  
bind_checker  
connect_supply_net  
create_upf2hdl_vct  
create_hdl2upf_vct  
create_power_domain
```

```
create_power_switch
create_pst
create_supply_net
create_supply_port
load_upf
map_isolation_cell
map_level_shifter_cell
map_power_switch
map_retention_cell
name_format
save_upf
set_scope
set_isolation
set_isolation_control
set_retention
set_retention_control
set_domain_supply_net
set_design_top
set_level_shifter
```

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example loads the UPF script file named top.upf:

```
prompt> load_upf top.upf
```

Loading UPF file top.upf with version 1.0...

End loading UPF file top.upf

1

The following example shows the usage of the **-scope** option of the **load_upf** command:

```
prompt> load_upf top.upf -scope Afp
```

Loading UPF file top.upf with version 1.0...

End loading UPF file top.upf

1

prompt>

The following example reads the UPF script file named top2.upf that contains commands not supported by UPF. The CMD-005 error messages are generated for the unsupported commands.

```
prompt> load_upf top2.upf
```

```
Loading UPF file top2.upf with version 1.0...
link_design
Error: Unknown command 'link_design' (CMD-005)
```

```
End loading UPF file top2.upf
0
prompt>
```

The following example shows the behavior of **load_upf** when the UPF file contains commands not supported by the tool. One UPF message is generated for every unsupported command. When the execution of the **load_upf** command is complete, the tool also reports a summary of all unsupported commands encountered in the UPF file.

```
prompt> load_upf t2.upf
```

```
Loading UPF file t2.upf with version 1.0...
set_logic_zero portA
Warning: Constraint 'set_logic_zero' is not supported by dc_shell. (UPF-042)
set_logic_zero portB
Warning: Constraint 'set_logic_zero' is not supported by dc_shell. (UPF-042)
set_logic_one portC
Warning: Constraint 'set_logic_one' is not supported by dc_shell. (UPF-042)
```

```
Summary of unsupported constraints:
Information: Ignored 1 unsupported 'set_logic_one' constraint. (UPF-043)
Information: Ignored 2 unsupported 'set_logic_zero' constraints. (UPF-043)
```

```
End loading UPF file t2.upf
1
prompt>
```

The following example shows the behavior of **load_upf** when unknown options or syntax errors are encountered in the UPF file:

```
prompt> load_upf 9.upf
```

```
Loading UPF file 9.upf with version 1.0...
set_scope A
create_power_domain PD1
create_supply_port o -dir out -domain PD1 -abcd
Error: unknown option '-abcd' (CMD-010)
Error: Errors reading UPF file: . Use error_info for more information. (UPF-044)
```

```
End loading UPF file 9.upf
0
```

```
prompt> load_upf 5.upf
```

```
Loading UPF file 5.upf with version 1.0...
Error: Errors reading UPF file: EOF found before command was complete.
Use error_info for more info. (UPF-044)
```

```
End loading UPF file 5.upf
0
prompt>
```

Recursion is supported by **load_upf**; however, circular recursion is not supported. The following example shows the behavior of the **load_upf** command when circular recursion is detected:

```
prompt> load_upf 8.upf
```

```
Loading UPF file 8.upf with version 1.0...
load_upf 8.upf
Error: Detected circular references to file 8.upf while loading UPF file 8.upf. (U
PF-045)
```

```
End loading UPF file 8.upf
load_upf 5.upf
```

```
Loading UPF file 5.upf with version 1.0...
set_scope A
create_power_domain PD1 -elements {B1 B2}
create_supply_port o -dir out -domain PD1
Error: Name space conflict while trying to create object A/o. (UPF-049)
load_upf 6.upf
```

```
Loading UPF file 6.upf with version 1.0...
add_port_state o -state {state_off off}
Error: Value for list 'supply_port_name' must have 1 elements. (CMD-036)
load_upf 7.upf
```

```
Loading UPF file 7.upf with version 1.0...
create_supply_port i -dir out -domain PD1
Error: Name space conflict while trying to create object A/i. (UPF-049)
add_port_state i -state {state_1_0 1.0}
Error: Value for list 'supply_port_name' must have 1 elements. (CMD-036)
```

```
End loading UPF file 7.upf
```

```
End loading UPF file 6.upf
```

```
End loading UPF file 5.upf
load_upf 8.upf
Error: Detected circular references to file 8.upf while loading UPF file 8.upf. (U
PF-045)
```

```
End loading UPF file 8.upf
```

```
End loading UPF file 8.upf
1
prompt>
```

The following example illustrates the use of the **-noecho** option:

```
prompt> load_upf 5.upf
```

```
Loading UPF file 5.upf with version 1.0...  
set_scope A  
create_power_domain PD1 -elements {B1 B2}  
create_supply_port o -dir out -domain PD1
```

```
End loading UPF file 5.upf  
1
```

```
prompt> load_upf 5.upf -noecho
```

```
Loading UPF file 5.upf with version 1.0...
```

```
End loading UPF file 5.upf  
1  
prompt>
```

The following example shows the message generated when an invalid scope is specified with **load_upf**:

```
prompt> load_upf 5.upf -scope AM
```

```
Error: Could not set the scope to AM. load_upf failed. (UPF-047)
```

```
End loading UPF file 5.upf  
0
```

SEE ALSO

```
save_upf(2)  
set_scope(2)
```

magnet_placement

Performs magnet placement. Pulls standard cells closer to macros and assigns legal locations.

SYNTAX

```
int magnet_placement
[-move_fixed]
[-mark_fixed]
[-move_soft_fixed]
[-mark_soft_fixed]
[-avoid_soft_blockages]
[-stop_by_sequential_cells]
[-exclude_buffers]
[-logical_level level]
[-stop_points object_list]
[-align]
magnet_objects
```

Data Types

<i>level</i>	integer
<i>object_list</i>	collection
<i>magnet_objects</i>	collection

ARGUMENTS

- move_fixed
Specifies that a cell marked fixed can be moved. A warning occurs when a fixed cell is being moved. By default, fixed cells cannot be moved.
- mark_fixed
Specifies that the cells are to be marked fixed after placement. By default, no cells are marked fixed.
- move_soft_fixed
Specifies that a cell marked soft fixed can be moved. A warning occurs when a soft fixed cell is being moved. By default, fixed cells cannot be moved.
- mark_soft_fixed
Specifies that the cells are to be marked fixed after placement. By default, no cells are marked fixed.
- avoid_soft_blockages
Specifies that cells should not be placed over soft blockages. By default, soft blockages will be ignored.
- stop_by_sequential_cells
Specifies that the pulling of cells closer to magnet should stop when a sequential cell is encountered during level(s) traversal. By default, all the cells in the logical level(s) of the **magnet_object** will be pulled.

-exclude_buffers

Specifies not to consider buffers and inverters when calculating the level. They will be pulled like normal cells but will be skipped over for level determination. By default, buffers and inverters are considered as one level of logic.

-logical_level level

Specifies the number of logical levels to pull. By default, *level* is 1 (only the first level cells are pulled). This option is mutually exclusive with *-stop_points*.

-stop_points object_list

Specifies to pull only the cells that are on the timing path between the **magnet_objects** and the **-stop_points** objects. The objects can be list of any pin, port or cell object(s). This option is mutually exclusive with *-logical_level*.

-align

Specifies to place the cells in a horizontal or vertical alignment with respect to the pin of the macro.

magnet_objects

Specifies the objects to become magnets. The objects can be a fixed macro cell, a pin of a fixed macro cell, or an I/O pin. Objects like Vias, blockages are not allowed to be magnets.

DESCRIPTION

The **magnet_placement** command allows you to define a set of objects as magnets and pull the neighboring cells up to a specified logical level closer to the magnet. Specify a large macro with many placeable standard cell neighbors. Magnet placement can be used on this type of a macro to ensure that the standard cells are placed close to the macro, resulting in better timing and congestion.

It is considered best practice to run the **magnet_placement** command before the **create_placement** command, so that sufficient areas are available for magnet placement operation.

If there are a large number of high fanout nets, **magnet_placement** might pull a large number of cells resulting in large run-time. In order to avoid such cases the environment variable **magnet_placement_fanout_limit** could be set to some reasonable value. The default value is 1000.

The effect of magnet placement is not preserved from one invocation to the other. For example if cells A and B are made magnets and there are cells which are connected to both of them through possibly multiple levels. Doing **magnet_placement A** and then **magnet_placement B**, could result in a different result then doing **magnet_placement [get_cells "A B"** together. Doing the two cells together would give correct result.

If **magnet_placement** is invoked after **create_placement**, there could be standard cells already placed. In such a case finding legal locations for all the cells might not be possible. Hence **magnet_placement** will place cells on top of existing cells. It is recommended that the user uses the *-mark_fixed* option to put a restriction on the

cells, so that operations like `place_opt` or `create_placement` would not move the cells. The user can then invoke these commands and run `legalize_placement` to remove the overlaps. This would move standard cells away from the magnet cells. The user might require to remove the fixed restriction on the magnet_cells and then run a second **`legalize_placement`** to legalize the magnet_cells.

If you do not want cell overlaps, set the **`magnet_placement_disable_overlap`** variable before running `magnet_placement`.

With `stop_by_sequential_cells` option, if you want to sequential cell element to also get pulled towards magnet, set the **`icc_magnetpl_stop_after_seq_cellfp`** variable before running `magnet_placement`. By default only the logic before sequential element will get pulled by magnet placement.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example performs magnet placement on the INST_1 cell:

```
prompt> magnet_placement [get_cells "INST_1"]
```

The following example performs magnet placement and marks the moved cells as dont touch placement:

```
prompt> magnet_placement -mark_fixed [get_cells "INST_2"]
```

The following example pulls two levels of cells to INST_2. Buffers (if any) are pulled but skipped over when calculating levels.

```
prompt> magnet_placement -exclude_buffers -logical_level 2 \  
[get_cells "INST_2"]
```

SEE ALSO

```
create_placement(2)  
legalize_placement(2)  
set_cell_location(2)  
set_dont_touch_placement(2)
```

man

Displays reference manual pages.

SYNTAX

```
string man  
topic
```

Data Types

```
topic      string
```

ARGUMENTS

```
topic
```

Specifies the subject to display. Available topics include commands, variables, and error messages

DESCRIPTION

The **man** command displays the online manual page for a command, variable, or error message. Users can write man pages for their own Tcl procedures and access them with the **man** command by setting the **sh_user_man_path** variable to an appropriate value. See the man page for **sh_user_man_path** for details.

EXAMPLES

The following command displays manual page for the **echo** command.

```
prompt> man echo
```

The following command displays the manual page for the error message CMD-025.

```
prompt> man CMD-025
```

SEE ALSO

```
help(2)  
sh_user_man_path(3)
```

map_freeze_silicon

Manually maps cells to the specified spare cells.

SYNTAX

```
status map_freeze_silicon  
cell_name  
spare_cell_name | -map_file map_file_name
```

Data Types

<i>cell_name</i>	string
<i>spare_cell_name</i>	string
<i>map_file_name</i>	string

ARGUMENTS

cell_name
Specifies the name of the cell to be mapped. This option accepts a collection of single elements as input. The specified cell should already exist in the design with the `eco_add` and `eco_status` attributes specified. The name of the cell is treated as an exact name, which means simple pattern matching is disabled.
This option is mutually exclusive with the **-map_file** option and is used together with the *spare_cell_name* option.

spare_cell_name
Specifies the name of the spare cell used for mapping. The option accepts a collection of single elements as input. The spare cell should already exist in the design and it should be a spare cell. The name of the spare cell is treated as an exact name, which means simple pattern matching is disabled. This option is mutually exclusive with the **-map_file** option and is used together with the *cell_name* option.

-map_file *map_file_name*
Specifies the name of the mapping file where contains the information of the cells and spare cells in pairs.

DESCRIPTION

This command replaces the `eco_add` cell with the specified spare cell. If either of the `eco_add` cell and spare cell have the `gate_array_master_type` attribute, the replacement succeeds when they have the same `gate_array_master_type` id and the width of spare cell is no less than the width of `eco_add` cell. If neither of them has the `gate_array_master_type` attribute, the replacement succeeds when they have the same cell reference.

You can specify the cell name and spare cell name in pairs with the command. Or you can specify the mapping file to be used with the **-map_file** option.

The following is the format of the mapping file: `cell_1 spare_cell_1 cell_2`

spare_cell_2

EXAMPLES

The following example uses **map_freeze_silicon** in the current design.

```
prompt> map_freeze_silicon U1 U1_spare
```

In the following example, cells are mapped to the spare cells based on the information in the mapping file called mapFile.

```
prompt> map_freeze_silicon -map_file mapFile
```

SEE ALSO

```
place_freeze_silicon(2)  
create_cell(2)  
remove_cell(2)  
change_link(2)  
eco_netlist(2)
```

map_isolation_cell

Specifies how to map or remap the isolations and enable level-shifter cells belonging to the specified isolation strategy. This command is supported only in UPF mode.

SYNTAX

```
status map_isolation_cell  
isolation_strategy  
-domain power_domain  
-lib_cells lib_cells
```

Data Types

<i>isolation_strategy</i>	string
<i>power_domain</i>	string
<i>lib_cells</i>	list

ARGUMENTS

isolation_strategy
Specifies the UPF isolation strategy name. The name of the isolation strategy must be unique within the specified power domain.

-domain *power_domain*
Specifies the name of the power domain name to which this isolation strategy belongs.

-lib_cells *lib_cells*
Specifies a list of the target library cells to be used for the isolation mapping.

DESCRIPTION

This command defines how the **compile** command performs the mapping of an isolation cell. All isolation cells that match the strategy will be mapped or remapped to one of the cells specified with the **-lib_cells** option. Resizing is done if required; but the choice is limited only to the library cells specified with the **-lib_cells** option.

You can also specify enable level-shifter cells using the **-lib_cell** option. If the isolation cells are later changed to enable level-shifter cells in the technology library, the cells can only be mapped to the one of the enable level-shifter cells specified with this command.

EXAMPLES

In the following example, the isolation cells belonging to the strategy named *isolation_1* can only be mapped to the LIB_HICLAMP0X2 or LIB_HICLAMP0X4 library cells:

```
prompt> map_isolation_cell isolation_1 -domain PD1 \  
-lib_cells {LIB_HICLAMP0X2, LIB_HICLAMP0X4}
```

SEE ALSO

```
set_isolation(2)  
set_isolation_control(2)
```

map_level_shifter_cell

Specifies that the level-shifter cells belonging to the specified strategy can only be mapped to a subset of the library cells. This command is supported only in UPF mode.

SYNTAX

```
status map_level_shifter_cell  
level_shifter_strategy  
-domain power_domain  
-lib_cells lib_cells
```

Data Types

<i>level_shifter_strategy</i>	string
<i>power_domain</i>	string
<i>lib_cells</i>	list

ARGUMENTS

```
level_shifter_strategy  
    Specifies the UPF level-shifter strategy name. The level-shifter strategy  
    name must be unique within the specified power domain.  
  
-domain power_domain  
    Specifies the power domain name to which this level-shifter strategy belongs.  
  
-lib_cells lib_cells  
    Specifies a list of library cells that can be used to map to the level-shifter  
    cells.
```

DESCRIPTION

This command defines how the **compile** command maps the level-shifter cells. All level-shifter cells belonging to the specified strategy are mapped to one of the library cells specified in the **-lib_cells** list. This subset is honored during resizing. If a valid solution cannot be reached using only the specified library cells, no level-shifter cells are inserted. For example, if the level shifters specified in the **-lib_cells** option cannot be used to shift between the required voltage levels, the tool does not insert any level-shifter cell.

For mapping the enable level-shifter cells, use the **map_isolation_cell** command.

EXAMPLES

In the following example the level-shifter cells belonging to the strategy named VL_1 can only be mapped to the LIB_LSHIX2 or LIB_LSHIX4 library cells.

```
prompt> map_level_shifter_cell LVL_1 -domain PD1 \
```

```
-lib_cells {LIB_LSHIX2, LIB_LSHIX4}
```

SEE ALSO

```
map_isolation_cell(2)  
set_level_shifter(2)
```


map_power_switch

Defines which power switch library cells to use for the mapping of the given UPF power switch.

SYNTAX

```
status map_power_switch  
switch_name  
-domain domain_name  
-lib_cells list
```

Data Types

<i>switch_name</i>	string
<i>domain_name</i>	string
<i>list</i>	list

ARGUMENTS

switch_name
Specifies the name of the power switch to be mapped using the specified library cells. This is a required option.

-domain *domain_name*
Specifies the power domain where this power switch was created. This is a required option.

-lib_cells *list*
Specifies a list of target library cells to use for the power switch cell mapping. This is a required option.

DESCRIPTION

This command is used to explicitly specify which library power switch cells are mapped for the corresponding UPF power switch.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example maps the `switch_1A` power switch in the power domain named `myPowerDomain` with the `lib2A/switch2A` library cell:

```
prompt> map_power_switch switch_1A \  
        -domain myPowerDomain -lib_cells lib2A/switch2A
```

1

SEE ALSO

`create_power_switch(2)`
`get_physical_lib_cells(2)`

map_retention_cell

Defines how to map the unmapped sequential cells to retention cells for the specified UPF retention strategy of the power domain. This command is supported only in UPF mode.

SYNTAX

```
status map_retention_cell  
retention_strategy  
-domain power_domain  
[-lib_cells lib_cells]  
[-lib_cell_type lib_cell_type]  
[-elements objects]
```

Data Types

<i>retention_strategy</i>	string
<i>power_domain</i>	string
<i>lib_cells</i>	list
<i>lib_cell_type</i>	string
<i>objects</i>	list

ARGUMENTS

retention_strategy
Specifies the UPF retention strategy name. The retention strategy name should be unique within the specified power domain.

-domain *power_domain*
Specifies the power domain name that this UPF retention strategy will be applied to.

-lib_cells *lib_cells*
Specifies a list of target library lib cells to be used for the retention mapping.

-lib_cell_type *lib_cell_type*
Specifies retention type to be used for the retention mapping.

-elements *objects*
Specifies the objects that the retention mapping only happens to the sequential cells within the specified objects. The objects can be a list of hierarchical cells, leaf cells, seqgen output signal net.

DESCRIPTION

This command defines how compile does the retention mapping. If **-elements** is not specified, then the mapping applies to all sequential cells that the specified retention strategy applies to.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows an example that maps the sequential cells under retention strategy *retenion_1* to CLK_LOW retention type.

```
prompt> map_retention_cell retention_1\  
-domain PD1\  
-lib_cell_type CLK_LOW
```

SEE ALSO

```
set_retention(2)  
set_retention_control(2)
```

mark_clock_tree

Marks clock attributes on clock objects.

SYNTAX

```
status mark_clock_tree
[-clock_trees name_or_source_pin_list]
[-clock_synthesized]
[-fix_sinks]
[-clock_net]
[-ideal_net]
[-routing_rule name_of_the_non_default_routing_rule]
[-use_default_routing_for_sinks n]
[-layer_list list_of_layer_names]
[-remove]
```

Data Types

<i>name_or_source_pin_list</i>	list
<i>name_of_the_non_default_routing_rule</i>	string
<i>n</i>	integer
<i>list_of_layer_names</i>	string

ARGUMENTS

-clock_trees *name_or_source_pin_list*
Marks only those clock trees whose names or root pins are listed in *name_or_source_pin_list*. Each element of *name_or_source_pin_list* must be either a clock tree name or a clock root (port or pin) shown in **report_clock**. By default, the command is applied to all currently defined clock trees.

-clock_synthesized
Marks all CTS related objects as synthesized by **compile_clock_tree** command.

-fix_sinks
Cooperates with **-clock_synthesized**, to enable **mark_clock_tree** to mark clock sinks as fixed placement to avoid neither sizing nor moving them.

-clock_net
Marks all clock nets as clock type. Router will identify these clock nets for clock routing. There will be no impact on the cells in the clock tree.

-ideal_net
Marks all clock nets as ideal type. Timer will reset synthesized clock tree back to ideal clock mode. There will be no impact on the cells in the clock tree.

-routing_rule *name_of_the_non_default_routing_rule*
Specifies the nondefault routing rule to be used for clock tree nets.

-use_default_routing_for_sinks *n*
Forces the default routing rule to be used on the leaf nets that drive the

clock tree sinks, and nets at the bottom $n-1$ levels of the clock tree. A net is considered as leaf net if it drives at least one flip-flop or latch. Float pins, exclude pins and dont_touch_subtree pins are treated as non-sink pins of the clock tree for this purpose. For example, a net that drives only one float pin will not be treated as leaf net for this purpose. The default is 0. This option is only active if the **-routing_rule** option is specified.

-layer_list *list_of_layer_names*

Specifies the list of layers that can be used during routing of clock nets. If the list has more than 2 elements, the min and max will be derived by the program.

-remove

Removes the specified attribute. Coordinates with **-clock_synthesized**, **-ideal_net**, **-routing_rule**, or **-layer_list**.

DESCRIPTION

This command marks clock attributes on clock objects and updates the design database with the results of the updated clock trees. The motivation of this command mainly comes from 3 parts. Firstly, synthesized clock trees imported from Astro or third-party cannot be recognized by ICC CTS or post-CTS engine due to lack of sufficient clock attributes. **mark_clock_tree -clock_synthesized** is to enable the capability to add necessary ICC CTS identity information for a clock tree synthesized by third-party. After attribute marking, the clock tree is supposed the same as built by **compile_clock_tree**. This is to smooth the third-party CTS plus ICC post-CTS flow. Secondly, **mark_clock_tree -clock_net** is to enable the capability to mark clock net type in database, which is desired for clock routing consideration. Thirdly, it is strongly required to have the capability to change non-default routing rule settings directly for a synthesized clock tree. This is to adjust the routing parameters to alleviate congestion, while re-compile the clock tree is unacceptable. However, it is a very tedious job to manually perform these NDR settings on thousands of clock nets through **set_net_routing_rule** or **set_net_routing_layer_constraints** commands. Here **mark_clock_tree -routing_rule** and **mark_clock_tree -layer_list** are to enable the NDR modification directly for a whole clock tree. This command should also be run after any post-CTS manual ECO done on clock network to mark clock attributes on newly created cells and nets.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example assumes a clock tree created by third party. Lack of sufficient ICC CTS attributes, it cannot be recognized and **report_clock_tree** is not able to report it. After **mark_clock_tree**, this clock tree can then be reported properly.

```
prompt> mark_clock_tree -clock_trees CLK1 -clock_synthesized
prompt> report_clock_tree -clock_trees CLK1
```

SEE ALSO

`report_clock_tree(2)`
`report_timing(2)`
`report_net_routing_layer_constraints(2)`
`compile_clock_tree(2)`
`optimize_clock_tree(2)`
`psynopt(2)`

mem

Reports memory usage information.

SYNTAX

```
int mem  
[-all]  
[-verbose]
```

ARGUMENTS

-all

Report the maximum of the peak memory usage among the main process and its child processes.

-verbose

Report in detail the peak memory usage of the main process and each child process, including placement, extraction, and routing.

DESCRIPTION

This command reports the peak memory usage of processes. By default, it reports the peak memory usage of the main process in kilobytes. If you specify the **-all** option, it reports the maximum of the peak memory usage among the main process and its child processes. If you specify both the **-all** and **-verbose** options, it also prints out the peak memory usage in megabytes for the main process and each child process.

Memory peak is defined as the memory high-water-mark of processes. When the tool reports a memory peak, it means the maximum memory allocated from the operating system. When child processes exist, the tool reports the maximum number among the main process and its child processes.

Note that the **mem** command reports the peak memory usage, not the swap space usage of the process. It actually reports how much memory was allocated from the operating system at the peak point. Some of the memory can be swapped out by the operating system based on the system status.

EXAMPLES

The following example shows the default command output.

```
prompt> mem  
102400
```

The following example shows the output when using the **-all** and **-verbose** options.

```
prompt> mem -all -verbose  
Main process mem-peak: 100 Mb  
Child "placement" called 3 times, mem-peak: 60 Mb  
Child "extraction" called 3 times, mem-peak: 50 Mb
```

```
mem  
1218
```


102400

SEE ALSO

`cputime(2)`

`monitor_cpu_memory(3)`

merge_clock_gates

Merges multiple compatible clock gates into one.

SYNTAX

```
status merge_clock_gates  
[-verbose]  
[-preview]
```

ARGUMENTS

-verbose
Specifies that **merge_clock_gates** should print verbose information about the merges done.

-preview
Specifies that **merge_clock_gates** should do the analysis and just print a preview of what merges are possible. No merging is done.

DESCRIPTION

This command analyzes clock gates at each level of hierarchy in the current design and replaces each set of compatible clock gates with a single clock gate. Two clock gates are considered compatible if all their input signals are logically equivalent. The merging is not done across hierarchy.

If **-preview** and **-verbose** options are specified together, only **-preview** is honored and no merging is done.

After merging, the **merge_clock_gates** command does a clean up of the design to remove any unused combinational logic.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example shows how to print information without merging.

```
prompt> merge_clock_gates -preview
```

The following example shows how to print detailed information.

```
prompt> merge_clock_gates -verbose
```

SEE ALSO

merge_flip_chip_nets

Merges the flip-chip nets into one net.

SYNTAX

```
status merge_flip_chip_nets  
-from from_nets  
-to to_net  
[-update_routing]
```

Data Types

<i>from_nets</i>	collection of net
<i>to_net</i>	string

ARGUMENTS

-from *from_nets*
Specifies a collection of the top-level nets from hierarchical net groups in the current design to be merged into one net. The nets must be of one net type. The pins, I/O ports, and terminals are disconnected from the nets in the collection and are reconnected to the merged net.

-to *to_net*
Specifies the destination net name of the merge. You can assign a new net name or you can use the name of one of the nets in the collection.

-update_routing
Specifies to update the owner_net attribute of the net shapes and vias of the specified nets in the collection. Use this option if these nets were partially routed using the **route_flip_chip** command.

DESCRIPTION

This command merges the flip-chip nets uniquified by the **assign_flip_chip_nets** command into one net. All components are disconnected from the nets in the specified collection and reconnected to the merged net. It is recommended that you use the **-update_routing** option if this command is used after flip-chip redistribution layer routing is performed using the **route_flip_chip** command.

EXAMPLES

The following example merges the VDD_# nets back to VDD.

```
prompt> merge_flip_chip_nets -from {get_nets VDD* -top_net_of_hierarchical_group -  
all} -to VDD  
-update_routing  
1
```

SEE ALSO

merge_fp_hierarchy

Creates a new level of hierarchy.

SYNTAX

```
int merge_fp_hierarchy  
[-design_name design_name]  
[-new_cell_name cell_name]  
[-cells cell_list]  
[-wrapper]  
[-input_sdc_file input_file_name]  
[-output_sdc_file output_file_name]  
[-load_internal_pins]  
[-load_back_sdc]  
[-no_port_merging]
```

Data Types

<i>design_name</i>	string
<i>cell_name</i>	string
<i>cell_list</i>	list
<i>input_file_name</i>	string
<i>output_file_name</i>	string

ARGUMENTS

- design_name** *design_name*
Specifies the name of the design containing the new level of hierarchy. This name must not already exist in the current design.
- new_cell_name** *cell_name*
Specifies the name of the new hierarchy level.
- cells** *cell_list*
Specifies a list of cells in the current design to merge into a new level of hierarchy. If more than one cell is specified, they must be enclosed in braces { }.
You can specify a collection of cells here. When necessary, you need specify full name here, like "a/b/c", to indicate the cell "c" under the module "b".
Note that the below scenarios may cause you encounter "User input data check failed" error:
- A non-existent cell is specified
 - The "new_cell_name" has already been used by other cell in the design
 - The "design_name" has already been used by other design in the design
 - The cells specified don't belong to the same parent.
 - The parent of the specified cells is multiple instantiated
- By checking your "cell_list" specification, you can pass the "input data check".
- wrapper**
Indicates to create the new cell as a wrapper.
A wrapper is used to selectively expose internal connections/dangling pins

onto the wrapper interface, so that future change on such internal objects wouldn't impact the wrapper interface.

Compared to a non-wrapper merging, a wrapper merging would create extra hier port (and hier net when necessary) on the new cell for:

1. original internal dangling hier port that has no outside connection.
2. original internal floating net that has only one port connected
2. original self-connected internal hier ports, e.g. u2/pin1 is directly connected to u2/pin2 that is on the same cell u2.
3. tie high or tie low internal hier port. Each tie high/low hier port would be connected to an hier port on the new cell.

The name of the new hier port would be same with its internal hier port.

Note a mapping file, named

"<wrapper_cell_name>_wrapper_portname.mapping_file", would be written in the working directory to show the port name change from original cells to the wrapper cell.

`-input_sdc_file input_file_name`

Specifies a sdc file that is to be updated to reflect the netlist change. Such a sdc file should contains valid sdc commands for current design.

`-output_sdc_file output_file_name`

Specifies the name of the output sdc file into which the updated sdc constraints would be written.

This option should be used with "-input_sdc_file" option.

`-load_internal_pins`

Indicates the design contains "internal pin" sdc constraints. This option is used for sdc constraints update only.

This option should be used with "-input_sdc_file" and "-output_sdc_file" options.

`-load_back_sdc`

Indicates that those updated sdc constraints should also be applied on the current design.

This option should be used with "-input_sdc_file" option.

`-no_port_merging`

This option stops the ports connected by the same interface net being merged during grouping under wrapper mode. This option only takes effects together with -wrapper option.

DESCRIPTION

This command merges any number of cells in the current design into a new cell, creating a new level of hierarchy. At the same time, you can specify the new design name for the new cell, however, remember to make this new design name unique in the current design.

This command can be used in early design planning or prototyping phase in which the designers need to create a physical hierarchy that is compatible with their floorplan structure. This new hierarchy is derived from logical hierarchy in netlist created by front end designers.

Please note the comments below:

This command doesn't automatically update SDC information stored in current design, unless you specify input SDC file and "-load_back_sdc" option. Or you can reapply SDC file later.

Currently this command couldn't merge cells located in different hierarchy level or different hierarchy module. E.g. this command can merge A/B/C and A/B/D into a new module E whose full hierarchy name is A/B/E; but it couldn't merge A/B/C and A/F/G, or merge A/B and A/F/G together.

This command also removes design constraints stored in the current design, since the merge operation would produce inconsistent with existing design constraints. So you need to reapply design constraints after running this command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

This example merges two cells into a new design:

```
prompt> merge_fp_hierarchy -cells {u1 u2} \  
-design_name NEW -new_cell_name U
```

This example specifies a collection of cells for the merge command:

```
prompt> merge_fp_hierarchy -cells [get_cells u*] \  
-design_name NEW -new_cell_name U
```

In the following example, the input SDC file is updated according to the netlist change. Note that the netlist change is committed at the same time.

```
prompt> merge_fp_hierarchy -design_name ANALOG_CELLS \  
-new_cell_name new -cells {u1 u2} \  
-input_sdc_file top.sdc -output_sdc_file out_top.sdc
```

The following example first changes the netlist, then updates the input SDC file, at the end applies the updated SDC file on the changed netlist.

```
prompt> merge_fp_hierarchy -design_name ANALOG_CELLS -\  
new_cell_name new -cells {u1 u2} \  
-input_sdc_file top.sdc -output_sdc_file out_top.sdc -load_back_sdc
```

SEE ALSO

flatten_fp_hierarchy(2)

merge_net_shapes

Merges a collection of net shapes and returns the collection of net shapes after merge.

SYNTAX

```
collection merge_net_shapes  
shape_collection
```

Data Types

```
shape_collection      list
```

ARGUMENTS

```
shape_collection  
    Specifies net shape names or a collection returned by the get_net_shapes  
    command.
```

DESCRIPTION

This command merges a collection of net shapes in the current design. It returns a collection handle (identifier) of net shapes after merge. If no objects be merged, the returned collection has the same size with the given collection, the objects in collection are the same too.

Please note that only interconnected net shapes which belong to a same net, on a same metal layer, has same width and end type are eligible to merge. Rectangle net shape and non-rectilinear ending segment of path won't be taken care.

For information about collections and the quering of objects, see the collecitons man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The example merges 2 net shapes to 1 net shape.

```
prompt> merge_net_shapes {PATH#91648 HWIRE#91392}  
{ "PATH#91648" }
```

SEE ALSO

```
create_net_shape(2)  
get_net_shapes(2)
```

```
remove_net_shape(2)
```

```
merge_net_shapes  
1228
```

merge_saif

Reads a list of SAIF files with their corresponding weights, computes the merged toggle rate and static probability, and annotates the switching activity for nets, pins, and ports in the current design. The command then generates a merged output SAIF file.

SYNTAX

```
integer merge_saif  
-input_list saif_file_and_weight_list  
-instance_name inst_name  
[-output merged_saif_name]  
[-simple_merge]  
[-ignore ignore_name]  
[-ignore_absolute ig_absolute_name]  
[-exclude exclude_file_name]  
[-exclude_absolute ex_absolute_file_name]  
[-unit_base unit_value]  
[-scale scale_value]  
[-khrate khrate_value]  
[-map_names]  
[-rtl_direct]  
[-strip_module annotated_instance_name]
```

Data Types

<i>saif_file_and_weight_list</i>	list
<i>inst_name</i>	string
<i>merged_saif_name</i>	string
<i>ignore_name</i>	string
<i>ig_absolute_name</i>	string
<i>exclude_file_name</i>	string
<i>ex_absolute_file_name</i>	string
<i>unit_value</i>	string
<i>scale_value</i>	integer
<i>khrate_value</i>	float
<i>annotated_instance_name</i>	string

ARGUMENTS

-input_list *saif_file_and_weight_list*
Specifies the name and the corresponding weight of the Switching Activity Interchange Format (SAIF) file list. The *saif_file_and_weight_list* argument is a list of items in the following format:

```
{-input name.saif -weight number}
```

where **-input** and **-weight** are keywords; *name* identifies a specific .saif file, and $0 < \textit{number} < 100$. For example, in the following statement, the gate_back_1.saif and gate_back_2.saif files are weighted by 20% and 80%, respectively and the sum of all weights is equal to **100**.

```
prompt> {-input gate_back_1.saif -weight 20 \
-input gate_back_2.saif -weight 80}
```

-instance_name *inst_name*

Specifies the instance name as it appears in each SAIF file of the current design instance to annotate. The command annotates the instance itself and all subinstances in the hierarchy of the specified instance.

-output *merged_saif_name*

Specifies the name of the output merged SAIF file.

-simple_merge

Specifies that all SAIF files are to annotate 100% of the nets, ports and pins of the current design and that the command is to perform a simple SAIF file data merge. You can use the **report_saif** command to verify that each SAIF file annotates 100% of the current design.

-ignore *ignore_name*

Specifies the name of an instance for which to ignore switching activity. Use this option to ignore switching activity within the hierarchy of that instance in each SAIF file. The ignore names are recognized after the -**instance_name** option.

-ignore_absolute *ig_absolute_name*

Specifies the name of an instance for which to ignore switching activity, but which is not affected by the use of the -**instance_name** option. The command determines whether a net name is under the "ignore hierarchy" before applying the -**instance_name** option is applied.

-exclude *exclude_file_name*

Specifies the name of a file that contains a list of names to ignore. Use this option to ignore switching activity for instances specified by the *exclude_file_name* argument. The file specified by the *exclude_file_name* argument must contain each "ignore name" on a separate line, without the -**exclude** switch. The ignore names are recognized after the -**instance_name** option is applied.

-exclude_absolute *ex_absolute_file_name*

Specifies the name of a file that contains a list of absolute names to be ignored but which are not affected by the use of the -**instance_name** option.

-unit_base *unit_value*

Specifies the base synthesis timing unit for all SAIF files. The synthesis timing unit is obtained by using the *scale_value* argument of the -**scale** option to scale this base unit. The valid values for *unit_value* are **ns**, **us**, or **ps**. The default is **ns**.

-scale *scale_value*

Specifies the value of the scaling factor for the base synthesis timing unit for all SAIF files. The synthesis timing unit is obtained by using *scale_value* to scale the base unit. The valid values for *scale_value* are **1**, **10**, or **100**. The default is **1**.

-khrate *khrate_value*

Specifies a default derating factor value to be used for internal problems for all SAIF files. If you do not use the **-khrate** option, the command uses a default derating factor of **0.5**.

-map_names

Specifies to use the SAIF name mapping mechanism to match objects in the SAIF file with design objects.

-rtl_direct

Indicates that the SAIF file has been generated from RTL simulation without reading in an RTL forward SAIF file. When using this option, ensure that the SAIF file obtained from simulation has been done on an RTL design and that no RTL forward SAIF files have been used. The following criteria must be met:

- When using the vpower PLI/FLI, the **read_rtl_saif** command has not been performed and the net monitoring has been set to *rtl_on* by using the **set_gate_level_monitoring** command in the programmable language interface (PLI), and the **set_net_monitoring_policy** command in FLI.
- When using the vcd2saif utility, the **-rtl** argument has not been used. For more information about using vpower PLI for Verilog, the power FLI interface for the MTI VHDL simulator, and the vcd2saif application, see the Power Compiler manuals.
Please note that flows involving forward RTL SAIF files and flows involving the **merge_saif** command with the **-rtl_direct** option will be made obsolete in future releases. Use flows involving the **saif_map** and **merge_saif -map_names** instead.

-strip_module *annotated_instance_name*

Specifies the instance name to be annotated in the current design. Note that this option is obsolete; use the **-instance** option instead.

DESCRIPTION

The **merge_saif** command reads a list of SAIF files with their corresponding weights, computes the merged toggle rate and static probability, and annotates the switching activity **toggle_rate** and **static_probability** attributes for nets, ports, and pins of the current design. The **merge_saif** command can also generate a merged output SAIF file.

To identify the instance (and corresponding subinstances) to annotate, use the **-instance_name** option. To set the synthesis timing unit, use the **-unit_base** and **-scale** options, unless you are certain that the synthesis timing unit is 1ns. To specify a default derating factor other than 0.5, use the **-khrate** option.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reads and merges weighted SAIF files, annotates switching activity for the current design, and generates an output SAIF file:

```
prompt> merge_saif -input_list \  
{-input gate_back1.saif -weight 10 \  
-input gate_back2.saif -weight 20 \  
-input gate_back3.saif -weight 30 \  
-input gate_back4.saif -weight 40} \  
-instance E/UUT -output merged_saif.saif
```

SEE ALSO

```
read_saif(2)  
report_saif(2)
```

move_mw_cel_origin

Moves the origin of a Milkyway design.

SYNTAX

```
status move_mw_cel_origin  
-to point  
[mw_cel_list]
```

Data Types

<i>point</i>	list
<i>mw_cel_list</i>	list

ARGUMENTS

-to *point*

Specifies the new location of the origin. The number is specified in user units which are determined by the technology information.

mw_cel_list

Specifies the Milkyway designs for which to move the origin. You can specify the Milkyway design by name, pattern, or by the mw_cel collection's name. By default, the command uses the current_mw_cel.

DESCRIPTION

This command sets the origin of the Milkyway design to a new location. It changes the coordinates of all physical objects in the Milkyway design.

For example, suppose the die area of the Milkyway design is {{10 10} {100 100}} and you want to make the lower-left point of die area at {0 0}; then you can move the mw_cel origin by specifying the following:

```
move_mw_cel_origin -to {10 10}
```

After the move, the die area is {{0 0} {90 90}} and all coordinates of the physical objects in this Milkyway design are decreased by 10.

If you use this command multiple times in a session, the origin for each subsequent command is based on the previous command, not the original origin that started the session.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example moves the current `mw_cel` location and sets the lower-left point at `{0 0}`.

```
prompt> get_attribute [get_die_area] bbox
{{10.000 10.000} {100.000 100.000}}
prompt> move_mw_cel_origin -to {10 10}
Moving cell origin ...
1
prompt> get_attribute [get_die_area] bbox
{{0.000 0.000} {90.000 90.000}}
```

The following example moves a Milkyway design's location by giving the `mw_cel` collection.

```
prompt> set my_design [get_mw_cels top_design]
prompt> move_mw_cel_origin -to {-10 -10} $my_design
Moving cell origin ...
1
```

SEE ALSO

`get_mw_cels(2)`

move_objects

Moves one or more objects to the specified location.

SYNTAX

```
status move_objects
[-delta vector]
[-from point]
[-to point]
[-x real]
[-y real]
[-keep_placement]
[-keep_pad_to_core_distance]
[-ignore_fixed]
objects
```

Data Types

objects collection

ARGUMENTS

-delta *vector*

Specifies the displacements for objects to move through. This option cannot be used with -to, -from, -x or -y options. By default, the command uses no displacement.

-from *point*

Specifies the reference point on the object or objects to be moved for the -to option. By default, the command uses the lower-left corner of the bounding box of the given object (or collection of objects) as the reference point. You can use the -from option only if you also specify the -to option.

-to *point*

Specifies the new location of the reference point for the object or objects. This option cannot be used with the -delta, -x or -y options.

-x *real*

Specifies the x coordinate for the given object (or collection of objects). All objects will be moved so that their left edge is at this x coordinate. This option cannot be used with the -delta, -to or -from options.

-y *real*

Specifies the y coordinate for the given object (or collection of objects). All objects will be moved so that their bottom edge is at this y coordinate. This option cannot be used with the -delta, -to or -from options.

-keep_placement

This option is effective only when applied to the core or the die. It will be ignored if applied on any other object.
All objects within the core or the die are moved together with the core or the die. Their relative locations are preserved.

The default is **false**.

-keep_pad_to_core_distance

This option is effective only when applied to the core or the die. It will be ignored if applied on any other object.

This maintains the distance between the core and the pad cells and by implication the distance between the core and the die, thus forcing changes to the core and the die to be made in tandem with each other. If pad cells are absent then the core-to-die distance will still be maintained.

The default is **false**.

-ignore_fixed

Normally fixed objects will not be moved. If this flag is supplied then fixed objects will also be moved.

objects

Specifies the objects to be moved.

DESCRIPTION

This command moves one or more objects to a specified location.

See `get_edit_property(2)` man page for details on which objects can be moved.

NOTES

Fixed objects will not be moved unless `-ignore_fixed` is specified.

Snapping is done automatically using global snap settings.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example moves all the route guides so that the left hand corner of the bounding box of all the route guides is at the coordinate {1400 1600}.

```
prompt> move_objects -to {1400 1600} [get_route_guides *]
```

The following example moves all selected objects so that their left hand edges are at `x = 0`.

```
prompt> move_objects -x 0 [get_selection]
```

SEE ALSO

`remove_objects(2)`

`resize_objects(2)`

`rotate_objects(2)`

`move_objects`

```
align_objects(2)
distribute_objects(2)
expand_objects(2)
flip_objects(2)
set_object_snap_type(2)
get_object_snap_type(2)
undo(2)
redo(2)
```

move_pins_on_edge

Moves a set of pins along the side of a cell or macro.

SYNTAX

```
status move_pins_on_edge  
-distance real  
[-overlap_policy overlap_policy_type]  
[-layer_policy layer_policy_type]  
pins
```

Data Types

<i>real</i>	float
<i>overlap_policy_type</i>	string
<i>layer_policy_type</i>	string
<i>pins</i>	collection

ARGUMENTS

-distance *real*
Specifies the distance to move pins along the edge of a cell or macro.
A positive distance indicates counter clockwise movement; a negative distance indicates clockwise movement.

-overlap_policy *overlap_policy_type*
Specifies the action the tool is to take if the move would cause pins to overlap.
The valid values of the *overlap_policy_type* argument and their descriptions are the following:

- **ignore** - ignore overlaps
- **warn** - warn the user
- **fail** - output an error message and don't move pins
- **fix** - use `remove_fp_pin_overlaps` command to fix overlaps
They are mutually exclusive; choose only one.
The default value is **warn**.

-layer_policy *layer_policy_type*
Specifies the action the tool is to take if the move places pins on new sides that have different preferred layers.
The valid values of the *layer_policy_type* argument and their descriptions are the following:

- **keep** - keep current layer
- **use_preferred** - use preferred layer for new side
They are mutually exclusive; choose only one.
The default value is **use_preferred**.

`pins`

Specifies the collection of pins to be moved.

They can be top level pins or macro pins, but they must all have the same parent cell or macro.

DESCRIPTION

This command moves a set of pins along the side of a cell or macro.

Note that supplied pins must all belong to the same cell or soft macro.

You can move the pins counter clockwise by using a positive distance or clockwise by using a negative distance.

You can also specify how overlaps are handled and whether layers are changed if the pin is moved to a new side with different preferred layers.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example moves pins **100nm** counter clockwise and fails if it causes overlaps:

```
prompt> set pins [get_pins -filter "is_hierarchical==true" *clk*]  
prompt> move_pins_on_edge -distance 100 -overlap fail $pins
```

SEE ALSO

`remove_fp_pin_overlaps(2)`

mw_cel_collection

Describes the methodology for using mw_cel collection.

DESCRIPTION

How to specify mw_cel by name

A unique mw_cel can be specified by the following:

```
A[.B[;C]]
```

Where:

A is the base name of the mw_cel, and the maximum length of A is 1023.

B is the view name of the mw_cel, and the maximum length of B is 31.

C is the version number. The version number starts at 1.

The following characters . (period) and ; (semicolon) are invalid in the base name, view name, or version of an mw_cel name.

The version of an mw_cel name is usually omitted. If the version is not specified, the latest version is assumed unless an individual command has a special interpretation. The latest version is the version with highest version number.

Because ; is a keyword in Tcl language, if the version is to be specified for an mw_cel, you must use proper quoting, such as double quotation marks, for example:

```
"top.CEL;1".
```

If you do not use double quotation marks, the command might appear to succeed, but might produce unexpected results. For example, in the following command:

```
prompt> open_mw_cel test1.CEL;1
Info: Opened "test1.CEL;2" from "/root/data/design" library
[1]
```

1 is considered as a separate command, and the **open_mw_cel** command actually opens the latest version, not version 1.

You can omit the view name. When you do this, you must also omit the version. If you do not specify the view name, the command will assume the CEL view and the latest version unless the individual command has a special interpretation.

Currently supported views

Currently, 5 kind of views are supported: CEL, FRAM, err, FILL and ILM. You can not specify the mw_cel of other views through the command line. However, some commands may have the ability to operate on other views internally.

Display of the mw_cel collection

The display of an mw_cel collection does not contain view and version. For example:

```
prompt> open_mw_cel test1
Info: Opened "test1.CEL;2" from "/root/data/design" library
{test1}
```

To get the view and version information, use the attribute version and view_name.

```
prompt> get_attribute [current_mw_cel] view_name
CEL
prompt> get_attribute [current_mw_cel] version
2
```

SEE ALSO

```
close_mw_cel(2)
collections(2)
copy_mw_cel(2)
create_mw_cel(2)
current_mw_cel(2)
get_mw_cel(2)
open_mw_cel(2)
remove_mw_cel(2)
rename_mw_cel(2)
save_mw_cel(2)
```

name_format

Specifies the name format for the isolation cells and level shifters.

SYNTAX

```
status name_format
[-isolation_prefix name]
[-isolation_suffix name]
[-level_shift_prefix name]
[-level_shift_suffix name]
```

Data Types

name string

ARGUMENTS

-isolation_prefix *name*
Specifies the prefix to be used for the isolation cell names. The default is "".

-isolation_suffix *name*
Specifies the suffix to be used for the isolation cell names. The default is "_UPF_ISO".

-level_shift_prefix *name*
Specifies the prefix to be used for the level shifter cell names. The default is "".

-level_shift_suffix *name*
Specifies the suffix to be used for the level shifter cell names. The default is "_UPF_LS".

DESCRIPTION

This command specifies the suffix and prefix for the isolation and level shifter cells. The name of the element that is being isolated or level shifted by the isolation or the level shifter cell is used in the middle. For example, an isolation cell that isolates a port named p1 will be named p1_UPF_ISO by default.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following is an example of the **name_format** command:

```
name_format
1242
```



```
prompt> name_format -isolation_prefix "MY_ISO" -level_shift_prefix "MY_LS"
```

SEE ALSO

```
set_isolation(2)  
set_level_shifter(2)
```

open_mw_cel

Opens a Milkyway design.

SYNTAX

```
collection open_mw_cel
[-readonly]
[-library library]
[-version version]
[-not_as_current]
mw_cel_name
```

Data Types

<i>library</i>	string
<i>version</i>	int
<i>mw_cel_name</i>	list

ARGUMENTS

-readonly

Specifies to open the Milkyway design for reading, but not for writing. You cannot modify and save the Milkyway design.

-library *library*

Specifies the library in which the Milkyway design exists. This option will be ignored if there is an opened Milkyway library.

If you also specified the **-readonly** option, and the *library* you specified was not already opened, the *library* will be opened as read-only. The side effect is that if you want to open another Milkyway design for writing in the same library later, you must first close the *library*, because it will still be open as read-only at that time.

If a library is already open in read-only mode, the Milkyway design is opened in read-only mode, even if it is not explicitly specified.

By default, the command assumes that the specified Milkyway design exists in the current library.

-version *version*

Specifies the version of the Milkyway design to open. A valid value of *version* is an integer value. This option will override the version specification given in the *mw_cel_name* command. For information on how to specify an *mw_cel_name*, see the *mw_cel_collection* man page.

-not_as_current

Specifies to open the Milkyway design in the background (that is, the Milkyway design gets opened but is not made current). The current design stays the same. This option is used only via a GUI command to overlay a different view of the Milkyway design on the current Milkyway design.

mw_cel_name

Specifies the name of the Milkyway design to open.

You can specify the Milkyway design name by using two parts: the name and the

open_mw_cel

view name. For example, *top.CEL* specifies a Milkyway design named *top* in the *CEL* view. If you give it a simple name, such as *test*, the command assumes you are looking for the Milkyway design name *test.CEL*.

You can give a Milkyway design name list or a Milkyway design collection as the argument. This means you can open several Milkyway designs at the same time. If successful, the last opened Milkyway design is set as the current Milkyway design, and then you can switch the current Milkyway design among them by using the **current_mw_cel** command.

For information on how to specify the *mw_cel_name*, see the *mw_cel_collection* man page.

DESCRIPTION

This command opens a Milkyway design and automatically sets it as the current Milkyway design. If you open a Milkyway design as read-only, you can retrieve only information from it; you are not allowed to change that Milkyway design.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example opens one Milkyway design:

```
prompt> open_mw_cel DUT
{"DUT"}
```

SEE ALSO

```
close_mw_cel(2)
copy_mw_cel(2)
create_mw_cel(2)
current_mw_cel(2)
get_mw_cels(2)
mw_cel_collection(2)
remove_mw_cel(2)
rename_mw_cel(2)
save_mw_cel(2)
```

open_mw_lib

Opens a Milkyway library.

SYNTAX

```
collection open_mw_lib  
[-readonly | -write_ref]  
mw_lib
```

Data Types

mw_lib string

ARGUMENTS

-readonly

Specifies to open the Milkyway library only for reading. You cannot modify and save the library in this mode.

The **-readonly** and **-write_ref** options are mutually exclusive. The default is to open the main library as read/write and all reference libraries as read only.

-write_ref

Specifies to open the reference library for writing. This option implies that the main library is opened for writing.

The **-readonly** and **-write_ref** options are mutually exclusive. The default is to open the main library as read/write and all reference libraries as read only.

mw_lib

Specifies to open the Milkyway library.

DESCRIPTION

This command opens a Milkyway library. The two access permissions for an opened library are read only or read/write. Specifying the write permission implies that read permission is granted. If you specify the **-readonly** option, the command opens the main library and all reference libraries as read only. If you specify the **-write_ref** option, it opens the main library and all reference libraries as read/write.

If you open the library as read only, you can read from the library but you cannot write to it.

Specifically, opening the Milkyway cel to write in it is not permitted. If you open the library as read/write, you can modify it.

The opened Milkyway library is automatically set to be the current Milkyway library.

You cannot open more than one main library at the same time. To open another main library, you must first close the previous main library.

This command returns a collection of Milkyway libraries if it succeeds.

See the man pages for the **set_mw_lib_reference** and **set_mw_technology_file** commands for information on setting or changing reference libraries and technology files.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example opens a Milkyway library named *access05* for writing in the current session:

```
prompt> open_mw_lib access05  
{"access05"}
```

SEE ALSO

```
close_mw_lib(2)  
copy_mw_lib(2)  
create_mw_lib(2)  
current_mw_lib(2)  
open_mw_lib(2)  
rebuild_mw_lib(2)  
rename_mw_lib(2)  
report_mw_lib(2)  
set_mw_lib_reference(2)  
set_mw_technology_file(2)  
write_mw_lib_files(2)
```

optimize_clock_tree

Performs clock tree optimization.

SYNTAX

```
status optimize_clock_tree
[-clock_trees clock_trees]
[-buffer_relocation]
[-buffer_sizing]
[-gate_relocation]
[-gate_sizing]
[-delay_insertion]
[-operating_condition min | max | min_max]
[-premesh]
[-postmesh]
[-mesh_net mesh_net_name]
[-routed_clock_stage none | global | track | detail | detail_with_signal_routes]
[-search_repair_loop loop_number]
[-no_clock_eco_route]
```

Data Types

<i>clock_trees</i>	collection
<i>mesh_net_name</i>	string
<i>loop_number</i>	integer

ARGUMENTS

-clock_trees *clock_trees*
Specifies the clock name or a collection of clock objects to optimize.

-buffer_relocation
Enables buffer relocation.

-buffer_sizing
Enables buffer sizing.

-gate_relocation
Enables gate relocation.

-gate_sizing
Enables gate sizing. Gate sizing performs sizing on clock gates and buffers in the clock tree to improve insertion delay and skew.

-delay_insertion
Enables delay insertion.

-operating_condition min | max | min_max
Specifies the operating condition to use. The default is **max**.

-premesh
Runs optimization on the premesh part of the clock tree from the root of the

clock tree to the mesh net.

-postmesh

Runs optimization on low-latency branches that are driven by clock mesh to reduce skew.

-mesh_net *mesh_net_name*

Specifies the name of the clock mesh net.

-routed_clock_stage none | global | track | detail | detail_with_signal_routes

Specifies the routing state of the design before running the

optimize_clock_tree command.

Specify one of the following values:

- **none** (the default)

Specifies that the clock nets are not routed. In this case, the command performs preroute clock tree optimization using the integrated clock global router and does not do any routing after the optimization is complete.

- **global**

Specifies that the clock nets are global routed. In this case, the command performs postroute clock tree optimization and performs track assignment and detail routing of the clock nets after the optimization is complete.

- **track**

Specifies that the clock nets are track assigned. In this case, the command performs postroute clock tree optimization and performs detail routing of the clock nets after the optimization is complete.

- **detail**

Specifies that the clock nets are detail routed. In this case, the command performs postroute clock tree optimization and performs ECO routing of the clock nets after the optimization is complete.

- **detail_with_signal_routes**

Specifies that both the clock nets and signal nets are detail routed. In this case, the command performs postroute clock tree optimization and performs ECO routing of the clock nets and signal nets after the optimization is complete. If you do not specify this option, the tool performs preroute optimization.

-search_repair_loop *loop_number*

Specifies the number of clock detail route search and repair loops. By default, the tool performs 2 search and repair loops.

-no_clock_eco_route

Skips ECO routing and extraction during postroute clock tree optimization. By default, postroute clock tree optimization performs clock routing based on the postroute stage and does extraction.

DESCRIPTION

This command performs clock tree optimization to improve the skew and insertion delay of the clock tree.

You can specify the optimization techniques to use by choosing from the following options: **-buffer_relocation**, **-buffer_sizing**, **-gate_relocation**, **-gate_sizing**, and **-delay_insertion**. If you do not specify any optimization techniques, the tool performs all of them.

You can also run this command when all clock nets have been routed or when all clock and signal nets have been routed. In either of these cases, you can use only the buffer and gate sizing options. In the postroute situation, unless you specify the **-no_clock_eco_route** option, the tool performs ECO routing and extraction based on the setting of the **-routed_clock_stage** option after the optimization stage is complete. The **-search_repair_loop** option specifies the number of loops to use in search and repair.

In a clock mesh flow, the **split_clock_net** command automatically creates clusters with reasonable balance and estimated latency. However, mesh clock trees might not be completely optimized because of the tight timing constraints imposed by mesh designs. Branches that are driven by a mesh with poor skew might impact the overall clock skew. Therefore, use the **-postmesh** when you specify **-mesh_net** to optimize the branches of a mesh clock tree, where the mesh net should be the high fanout net of the clock tree. The command models the mesh as ideal and restructures the branches by resizing, splitting up into two sections, and displacing the buffers of the branches to increase latency.

The **optimize_clock_tree** command can balance skew in multiple process corners. You activate the multicorner option with the **set_clock_tree_optimization_options** command. For more information about the **set_clock_tree_optimization_options** command, see the man page.

Multicorner-Multimode Support

This command uses information from the clock tree synthesis scenario.

EXAMPLES

The following example performs clock tree optimization by using the buffer sizing and buffer relocation options.

```
prompt> optimize_clock_tree -buffer_sizing -buffer_relocation
```

The following example performs clock tree optimization for the clock tree named Clk using only the delay insertion option.

```
prompt> optimize_clock_tree -clock_tree Clk -delay_insertion
```

The following example performs postroute clock tree optimization at the global route stage:

```
prompt> optimize_clock_tree -routed_clock_stage global
```


SEE ALSO

`compile_clock_tree(2)`
`set_clock_tree_options(2)`
`set_clock_tree_optimization_options(2)`

optimize_dft

Performs placement-aware or clock-aware scan reordering.

SYNTAX

```
status optimize_dft  
[-clock_buffer]  
[-plan_group]
```

ARGUMENTS

-clock_buffer

Invokes clock buffer-aware scan reordering, which aims to minimize the number of clock buffer crossings in the scan chain for better hold violation reduction along scan path.

-plan_group

The option is designed for Design Planning to enable plan group aware scan chain repartition and reordering after plan groups are defined. it will optimize the design to minimize top-level scan wires, scan ports on the blocks and scan net feedthroughs. This option is expected to be used prior to global routing and pin assignment.

DESCRIPTION

The **optimize_dft** performs scan chain reordering. By default it performs placement aware scan reordering. It aims to reduce the scan chain wire-length, minimize congestion and improve routability. Prior to using this option you must first use either the **read_def** command or **trace_scan_chain** command to read in scan chain information of the design, and have performed placement on the design.

When invoked with the **-clock_buffer** option, **optimize_dft** performs clock buffer-aware scan reordering. The reordering aims to minimize the number of buffer crossings in the scan chain. Minimizing the number of buffer crossings can reduce hold time violations in the scan chain. For best results you should first perform placement aware scan reordering.

For best results we recommend to use **place_opt -optimize_dft** option and **clock_opt -optimize_dft** commands instead of **optimize_dft** command. The **optimize_dft** command is provided for advanced users who want to have finer control over the placement and clock tree synthesis processes.

The standalone **optimize_dft** command does not update extraction or timing data so the user must do that. For example, use **psynopt** after **optimize_dft**.

Moreover, before performing any optimization, we strongly recommend to perform the **check_scan_chain** command since only "VALIDATED"V scan chains will be optimized with the **optimize_dft** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

SEE ALSO

`read_def(2)`
`trace_scan_chain(2)`
`place_opt(2)`
`clock_opt(2)`
`check_scan_chain(2)`
`report_scan_chain(2)`

optimize_flip_chip_route

Beautifies the flip-chip routing patterns.

SYNTAX

```
status optimize_flip_chip_route
[-nets collection_of_nets
  | -nets_in_file nets_file]
-layer tech_layer_number | layer_name
[-change_route_type user_enter | signal_route]
[-split_net bool]
```

Data Types

nets_file string

DATA TYPES

tech_layer_number integer
mX string
MX string
nets_file string
layer_name string

ARGUMENTS

-nets *collection_of_nets*
Specifies the nets to be optimized. By default, all flip-chip nets are optimized.
This option and the **-nets_in_file** option are mutually exclusive.

-nets_in_file *nets_file*
Specifies the name of a file that contains the list of nets to be optimized.
This option and the **-nets** option are mutually exclusive.

-layer tech_layer_number | layer_name
Specifies the metal layer on which to optimize the flip-chip wires. You specify the metal layer by using its metal number or technology file layer number.
This option is required.

-change_route_type user_enter | signal_route
Changes all existing flip-chip wires, paths and vias route type to "user_enter" or "signal_route".

-split_net *bool*
Splits a 45-degree routed wide net into two parallel thin nets. Use the **set_route_flip_chip_options** command to specify the required width and spacing of the thin nets before using this function. All the 45-degree nets are routed as paths and will remain paths after they are split.

DESCRIPTION

This command improves and enhances the flip-chip routing patterns by performing L-shape and Z-shape optimizations on the flip-chip wires.

EXAMPLES

The following example optimizes the flip-chip routing patterns on the top metal layer.

```
prompt> optimize_flip_chip_route
```

The following example changes the flip-chip routing route-type to user_enter.

```
prompt> optimize_flip_chip_route -change_route_type user_enter
```

SEE ALSO

```
set_route_flip_chip_options(2)  
remove_flip_chip_route(2)  
create_stack_via_on_pad_pin(2)  
route_flip_chip(2)  
write_flip_chip_nets(2)  
push_flip_chip_route(2)  
display_flip_chip_route_flylines(2)
```

optimize_fp_timing

Performs timing optimization on the design.

SYNTAX

```
status optimize_fp_timing  
[-effort medium | high]  
[-fix_design_rule]  
[-area_recovery]  
[-report_qor]  
[-feedthrough_buffering_only]  
[-keep_global_routes]
```

ARGUMENTS

- effort medium | high**
Specifies how much effort this command should use to optimize the design. If you select an effort of high more effort is expended to improve the timing of the design and therefore more iterations will be performed on the design. Hence a higher effort results in more CPU time. By default, the effort is medium.
- fix_design_rule**
Enables design rule violations fixing. By default, design rule fixing is not enabled.
- area_recovery**
Enables area recovery for the cells not on the timing critical paths. By default, area recovery is not enabled.
- report_qor**
Reports on the WNS and TNS of the design. Reports utilizations of each plan group and top level. Reports the number of buffers added and the number of cells sized for each plan group and top level. By default, report qor is off.
- feedthrough_buffering_only**
Adds buffers for feedthrough nets only. Optimization is not performed and timing is not updated. Using this option minimizes design changes at the late stage of design planning. You must run the **analyze_fp_routing** command before using this option; otherwise, the tool issues an error message and stops. By default, this option is off.
- keep_global_routes**
After running the **optimize_fp_timing** command, global routes, if any, are invalid and incomplete. Measuring QoR or congestion maps based on incomplete global routes will cause correlation issues. Therefore, global routes are removed when the **optimize_fp_timing** command is run. Use this option if you want to preserve the global routes.
To obtain a congestion map, you should do global route from scratch.

DESCRIPTION

The **optimize_fp_timing** command performs timing optimization on the design. The output of this command is a legally placed netlist. This command is plan group aware.

Multicorner-Multimode Support

This command uses information from active scenarios only.

EXAMPLES

The following example optimizes the design timing and fixes design rule violations.

```
prompt> optimize_fp_timing -fix_design_rule
```

The following example optimizes the design timing and reports the plan group related QoR after optimization.

```
prompt> optimize_fp_timing -report_qor
```

The following example optimizes the design timing from both WNS and TNS points of view.

```
prompt> optimize_fp_timing -tns_opt
```

SEE ALSO

psynopt(2)
analyze_fp_routing(2)

optimize_netlist_hierarchy

Modifies the netlist tree structure contained in the input netlist to create hierarchy nodes that can be physically partitioned in the top level such that the size of the nodes are relatively well-balanced, while minimizing the number of top-level nets that need to connect to the hierarchy block interface.

SYNTAX

```
status optimize_netlist_hierarchy
-netlist_files filelist
-top topName
[-opaque_modules modulelist]
[-max_partition_to_chip_ratio sizeRatio]
[-keep_top_hier modulelist]
[-hier_marker symbol]
[-max_hier_mod_depth Level]
[-max_num_nodes Count]
[-size_metric {simple | instcnt | area}]
[-filename string]
```

Data Types

<i>filelist</i>	Tcl list
<i>topName</i>	String
<i>modulelist</i>	Tcl list
<i>sizeRatio</i>	float (>0)
<i>symbol</i>	Char
<i>Level</i>	integer (>0)
<i>Count</i>	integer (>0)

ARGUMENTS

```
-netlist_files filelist
    Specify a list of Verilog design filenames for input. The list should follow
    the standard Tcl list format.

-top topName
    Specify the module name that should be treated as the top module.

-opaque_modules modulelist
    Specify a list of module names whose instances should retain (as given in
    input Verilog) their lower-levels of hierarchy all the way down to leaf cells.
    The default value is none.

-max_partition_to_chip_ratio sizeRatio
    This must be a positive floating value. The largest hierarchical node created
    will be smaller than ( $\text{sizeRatio} * \text{total chip size}$ ). The size metric used is
    dependent on the -size_metric option value. This ratio has an impact on the
    number of new top level hierarchical nodes that are created. If the value
    specified is larger than 1, then no new hierarchy nodes are created. The
    output Verilog will contain the top module which is completely flattened
    except for the instances of cells specified in opaque module list constraint.
```


While the actual number of new hierarchical nodes created for a given sizeRatio can vary depending on the original hierarchy, a generally reliable way to estimate when the sizeRatio is between 0 and 1 is: number of new hierarchical nodes = $((4/3)*(1/\text{sizeRatio}) + 0.5)$, rounded to the nearest integer.

-keep_top_hier *modulelist*

Specify list of module names whose instances in the design must be fixed in the top hierarchy, irrespective of their original hierarchy in the input Verilog. This is useful for example, to pull the I/O pad cells to the top so that floor-planning tools can work correctly.

-hier_marker *symbol*

By default, when hierarchical nodes are flattened, the names are encoded with the original full hierarchy using the "/" character. However, if the use of "/" character is not suitable in certain flows, you can optional specify a different character. No check is made if this optional specified character is indeed an unused character in the input Verilog. So you need to take care in using appropriate characters. If the hier_marker is a character that is already used in input Verilog, this command is likely to abort when writing output Verilog file.

-max_hier_mod_depth *Level*

Modifies the hierarchy from the root down to this depth level. Levels of hierarchy deeper than this depth are preserved as in the original netlist.

-max_num_nodes *Count*

Loads the netlist and internally breaks it into small units called "nodes". By setting a limit on the number of nodes, the command will automatically determine how to map the entire design into a specified number of maximum nodes. The value of the *count* specified is in thousands. The command will use only fewer or at most equal to specified number of nodes. From a user perspective, controlling this number impacts how much memory the command uses. When memory is limited and the design is large, this threshold should be lowered so that the command can execute. In general, the higher the threshold, the greater the quality of partitions but also more memory and runtime is used, which will modify the hierarchy more extensively.

-size_metric {simple | instcnt | area}

The command uses this flag to know how to measure the partition sizes. Both "simple" and "instcnt" metrics refer to the counting of leaf nodes while measuring the hierarchy node sizes. When using "simple" metrics, all opaque_modules are treated as a single standard cell where as in "instcnt" metric, opaque modules with 100 standard cells are treated as 100 times larger than a single standard cell. "Area" metric refers to measuring a hierarchy node by the area it will need to place all the leaf cells it contains. This is better when there are large sized hard macros in the design. By default, the "instcnt" metric is used.

DESCRIPTION

The gate-level netlist hierarchy structure in an input design (Verilog file) can be considered a tree-type structure with a top module as the root of the tree and the leaf nodes of the tree as standard cells, hard macros, I/Os, or black-boxes. To

create design convergence physical partitions, it is necessary to identify candidate hierarchy nodes in the tree that can be converted into well-balanced physical partitions with the least possible number of inter-partition nets to minimize the net cut cost. Quite often the logical hierarchy tree is not well balanced tree. This leads to poor quality physical partitioning, which in turn, results in overall hierarchical chip design QoR and run time degradation. In such cases, this command can be used to modify the incoming hierarchy structure into a more well-balanced physical partitions. The flat level netlist in the output is guaranteed to be equivalent to the input design.

The command operates on an input design specified in Verilog format files (**-netlist_files** option). Optionally, the command can preserve some part of the original hierarchy (**-opaque_modules** option). Therefore, the output netlist contains two components, one with a modified hierarchy structure, and the other with preserved hierarchy. The output netlist is separated into three ASCII files. The first is a Verilog format file that contains the modified hierarchy component of the design. The second file, written to the run directory as `opaque_modules.v`, contains the preserved hierarchy component of the design. The command skips writing the **opaque_modules.v** file if it is empty, but otherwise, will create (or overwrite if preexisting) the **opaque_modules.v** file. The third file, written into the run directory as `partition_cmds.tcl`, contains the sequence of ungroup and group operations that were applied when the **optimize_netlist_hierarchy** command was run. The third file provides an alternate way to generate the netlist contained in the other two output files. However, this is inefficient because the other two output files already contain the resultant netlist. This file is primarily intended for generating a new SDC timing constraints file given the original timing constraints file which may not work with the modified hierarchical netlist. The file uses callback functions such as, `ungroup_cb` and `group_cb`. Before sourcing this script in IC Compiler or DC Compiler, you must define the tcl callback functions. (Refer to the appropriate user guide for information on how to convert SDC files so that they work with an optimized hierarchical netlist.)

The qualitative aspects of the command functionality is briefly described here. The number of **physical partitioning -ready** hierarchy nodes created in the top level is implicitly defined by the constraint you specify on the maximum size of a partition (**-max_partition_to_chip_ratio** option). The size itself could refer to leaf instance count or area (**-size_metric** option). As described earlier, the **-opaque_modules** option allows you to specify a list of module reference names whose instances will preserve their lower level hierarchy. This is an effective way to control the command if the number of opaque modules are small and can be easily determined. Besides manually listing opaque modules, you can also use the command to auto-identify opaque modules. For the command to auto-identify the appropriate opaque modules, you can use the **-max_hier_mod_depth** and **-max_num_nodes** options as guidance. For example, consider a case where the hierarchy must be preserved on all nodes at hierarchy level 3 or deeper. It is a bit cumbersome to open a hierarchy browser and manually collect all the reference names at hierarchy level 3 and mark them as **-opaque_modules**. Instead, you could achieve the same objective by setting **-max_hier_mod_depth** to 3. Another use of opaque modules is related to memory consumption in the command. The **-max_num_nodes** option implicitly puts a limit on the maximum memory the command tries to consume. It does this by intelligently identifying a list of opaque modules. However, an error message occurs if incompatible options such as, **-opaque_list**, **-max_hier_mod_depth**, and **-max_num_nodes** are combined in the same call. It is important to recognize that the instances of opaque modules may be put into a different hierarchy than the original (only their lower levels of hierarchy are preserved). Finally, you can use the **-keep_top_hier**

option to specify that in the final output certain master cell's instances should always be placed in the top hierarchy irrespective of their original hierarchy level in the input. Such cell instances will not be pushed into any hierarchy node the command creates at the top level. One application of this option may be to relocate embedded I/O pad cells to the top.

Note the following limitations. In the input Verilog file, bus names should be a valid net name plus a suffix of the form "[%d]" to indicate the bus-bit index. Non-buses should not have the "[" and "]" characters in the name. (for example, *abc[23]1* is not a supported net name). If a bus port is declared as a split-bus port in the reference design, then in the instance pin-map, the split-bus port must be listed in sequence (MSB to LSB). If they are listed out-of-sequence, the output Verilog files may be incorrect (the output design will be different from the input) and may also trigger link/bind errors when loading in IC Compiler.

EXAMPLES

The following example shows how to generate an hierarchy optimized netlist files {output.v, opaque_modules.v} given input.v. partition_cmds.tcl is output in the run directory as well.

```
prompt> optimize_netlist_hierarchy -netlist_files input.v -top MYTOP -  
max_partition_to_chip_ratio 0.3  
-keep_top_hier {PDB08DGZ PDIANA2PC PDIDGZ} -max_num_nodes 500 -size_metric area -  
output nlout.v
```

SEE ALSO

optimize_power_switch

Optimizes IR drop by sizing header and footer cells up or down.

SYNTAX

```
status optimize_power_switch  
-real_pg_net real_pg_net_name  
-virtual_pg_net virtual_pg_net_name  
[-lib_cell lib_cells_or_power_switches]  
[-cells instances]  
[-legalize_placement]  
[-remove_preroute]  
[-preroute_mode_real_pg_port rail | tie | net]  
[-no_preroute_real_pg_port]
```

Data Types

<i>real_pg_net_name</i>	collection
<i>virtual_pg_net_name</i>	collection
<i>lib_cells_or_power_switches</i>	collection
<i>instances</i>	collection

ARGUMENTS

- real_pg_net *real_pg_net_name*
Specifies a real power or ground net to perform power network analysis. Real power nets connect to the real power pins of the MTCMOS cells. The real power net is not powered down. The *real_pg_net_name* argument can be a collection or list of one item.
- virtual_pg_net *virtual_pg_net_name*
Specifies a virtual power or ground net on which to perform IR drop analysis. Virtual power nets connect to the virtual power pins of the MTCMOS cells. The virtual power net can be powered down by the MTCMOS header or footer cells to reduce leakage power. The *virtual_pg_net_name* argument can be a collection or list of one item.
- lib_cell *lib_cells_or_power_switches*
Specifies the header or footer multithreshold CMOS (MTCMOS) lib cells to use for optimization. Use power switches for Unified Power Format (UPF) mode, and use lib cells for non-UPF mode. The *lib_cells_or_power_switches* argument can be a collection or list. By default, the command gets all the header and footer lib cells of the current design, whose power and ground type is the same as that of the real power and ground net.
- cells *instances*
Specifies the header and footer cells to be optimized. The *instances* argument can be a collection or list. By default, the command gets all header and footer cells that are connected to the real power and ground net and virtual power and ground net.

-legalize_placement
Legalizes cell placement after each sizing loop.

-remove_preroute
Removes the standard cell connection user via type of preroute after each sizing.

-preroute_mode_real_pg_port rail | tie | net
Specifies the preroute mode for the real power and ground port of MTCMOS cells to pass to preroute_standard_cells. The default is **rail**.

-no_preroute_real_pg_port
Skips the preroute real power and ground port of MTCMOS cells.

DESCRIPTION

This command optimizes the target IR drop by sizing up or down the MTCMOS header and footer cells in the design. The command attempts to size up cells in the neighborhood that exceeds the IR drop target. It sizes down cells in the neighborhood where the IR drop is much less than the specified target.

Prerequisites

Before executing this command, you must perform the following tasks, in order, that are necessary for the Power Network Analysis (PNA) engine to compute the IR drop.

1. Use the **set_attribute** command to set the "mtcmos_resistance" attribute to define power-on resistance of the header or footer cell.
2. Use the **set_attribute** command to set the "mtcmos_pin_layers" attribute to specify the metal layer of the power pins.
3. Use the **create_fp_virtual_pad** command to define the power pad or virtual power pad.
4. Use the **set_mtcmos_pna_strategy** command to define the target IR drop.
5. Use the **derive_pg_connection** command, if necessary, to ensure that the virtual and real power and ground nets of all standard cells have been properly connected logically in the netlist. A standard cell to be powered down should connect to a virtual power and ground net. A standard cell that always stays on should be connected to a real power and ground net.
6. Use the **set_mtcmos_pna_strategy** command to define the PNA-related option.

EXAMPLES

The following example optimizes the IR drop of the design by sizing the MTCMOS header and footer cells. The IR drop target is 150mV. The power nets to be analyzed are VIRTUAL_VDD and VDD.

```
prompt> set_mtcmos_pna_strategy -target_voltage_drop 150
1
prompt> optimize_power_switch -virtual_pg_net VIRTUAL_VDD\
-real_pg_net VDD
1
```

SEE ALSO

`create_power_switch_array(2)`
`analyze_fp_rail(2)`
`create_fp_virtual_pad(2)`
`derive_pg_connection(2)`
`explore_power_switch(2)`
`replace_power_switch(2)`
`report_mtcmos_pna_strategy(2)`
`set_attribute(2)`
`set_mtcmos_pna_strategy(2)`

optimize_pre_cts_power

Performs power optimization before the clock tree synthesis stage.

SYNTAX

```
status optimize_pre_cts_power  
[-operating_condition min | max(default) | min_max]  
[-update_clock_latency]  
[-verbose]
```

ARGUMENTS

-operating_condition [min | max(default) | min_max]
Instructs the clock tree synthesis engine to use the specified operating condition.

-update_clock_latency
Instructs to invoke **update_clock_latency** before running power optimization.

DESCRIPTION

The command enables you to perform power optimization that includes power aware placement and clock gating optimization before clock tree synthesis (CTS) stage. This command only performs pre-CTS power optimization, thus no clock trees are seen at the completion of **optimize_pre_cts_power**. However, this command internally utilizes the CTS engine for clock tree estimations. Any ideal clock latency and/or clock uncertainty, specified on pins in the clock structure, are removed by the under-the-hood CTS engine. At the end of this command but before the actual CTS stage, if you need to report the timing based on the previously imposed ideal clock latency and clock uncertainty, reload them manually.

The **set_power_options** command is used to control the behavior of the pre-CTS power optimization by enabling or disabling options such as **-low_power_placement** and **-clock_gating**. When **-low_power_placement** is enabled, switching activity based on power-aware placement is utilized in an incremental way. For the maximum power optimization performance from the power-aware placement, you should specify the same coarse placement settings that are used during place_opt or include the settings in the psyn_constraint_file before running this command. The detailed options for ICG optimization strategy can be specified by using the **set_optimize_pre_cts_power_options** command.

If you specify **-psyn_constraint_file**, **-cts_constraint_file**, or both with the **set_optimize_pre_cts_power_options** command, the options will be sourced for physical synthesis and internal clock tree synthesis steps during the **optimize_pre_cts_power** command. Note that the constraints defined in either Tcl files will be persistent in the design, so they can affect future operations on the design. Ensure that the constraints for CTS are applied before running any actual CTS commands.

If **-update_clock_latency** is turned on, the **update_clock_latency** will be called before the **psynopt** command.

All the options for **optimize_pre_cts_power** are reported by the **report_optimize_pre_cts_power_options** command.

Multicorner-Multimode Support

This command uses information from the clock tree synthesis scenario, as well as the active scenarios.

EXAMPLES

The following example shows how to invoke pre-CTS power optimization before CTS.

```
prompt> set_power_options -low_power_placement TRUE -clock_gating TRUE  
prompt> set_optimize_pre_cts_power_options -honor_dont_touch  
prompt> optimize_pre_cts_power  
prompt> compile_clock_tree
```

SEE ALSO

```
set_power_options(2)  
set_optimize_pre_cts_power_options(2)  
report_optimize_pre_cts_power_options(2)  
compile_clock_tree(2)  
clock_opt(2)
```


optimize_wire_via

Optimizes the routing to minimize wire length.

SYNTAX

```
status optimize_wire_via
[-nets collection_of_nets]
[-exclude_nets collection_of_nets]
[-search_repair_loop num]
[-run_time_limit num]
[-num_cpus num]
```

Data Types

num integer

ARGUMENTS

- nets *collection_of_nets***
Specifies the nets that will be optimized. The **-nets** and **-exclude_nets** options are mutually exclusive. If you do not specify either option, all nets are optimized.
- exclude_nets *collection_of_nets***
Specifies the nets that will not be optimized. The **-nets** and **-exclude_nets** options are mutually exclusive. If you do not specify either option, all nets are optimized.
- search_repair_loop *num***
Specifies the number of search and repair loops to be run. The default value is 50. Range is 0 to 1000.
- run_time_limit *num***
Specifies the cpu runtime limit in minutes. Default is no limit (-1). Range is -1 to 500.
- num_cpus *num***
Specifies the number of CPUs for distributed routing. Number must be >= 1. The default value is 1. Range is 0 to 500.

DESCRIPTION

This command invokes route optimization. It will optimize the routing to minimize wire length for all signal nets and clock nets.

Some options are set by using the **set_route_options command**. Please see the **set_route_options documentation**.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

```
prompt> optimize_wire_via -search_repair_loop 3  
1
```

SEE ALSO

set_route_options(2)

order_rp_groups

Orders the relative placement groups created by the **extract_rp_group** command in a sequential manner.

SYNTAX

```
int order_rp_groups  
-group_name group_name  
[-output file_name]  
[-append]  
[-nosplit]  
[-apply]  
rp_groups
```

Data Types

<i>group_name</i>	string
<i>file_name</i>	string
<i>rp_groups</i>	list or collection

ARGUMENTS

-group_name *group_name*
Specifies a name for the relative placement group. Each relative placement group in the design must have a unique name.

-output *file_name*
Specifies the name of the output file to write the constraints to. If you do not specify this option, the constraints are written to the screen.

-append
Specifies that the constraints should be appended to the file specified in the **-output** option. By default, if the specified file exists, it is overwritten.

-nosplit
Specifies that the tool should not split the lines when writing out the relative placement constraints.

-apply
Specifies that the relative placement constraints generated by the **order_rp_groups** command are applied to the design. By default, the constraints are written out, but are not applied to the design.

rp_groups
Specifies a list of existing relative placement groups to include in the created group. These groups are added to the new group in the order in which they are specified.

DESCRIPTION

The **order_rp_groups** creates a hierarchical relative placement group that contains the specified groups in the order in which they are specified.

The command returns 1 if the group was created successfully, 0 otherwise.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following examples show the creation of relative placement groups using the `order_rp_groups` command.

```
prompt> order_rp_group -  
group_name group0 {top::misc0 top::misc1 top::misc2 top::misc3 top::misc4  
top::misc5 top::misc6 top::misc9 top::misc03 top::misc04 top::misc05 top::  
misc06 top::misc07 top::misc08} -apply  
1  
prompt> order_rp_group -  
group_name group0 {top::misc0 top::misc1 top::misc2 top::misc3 top::misc4  
top::misc5 top::misc6 top::misc9 top::misc03 top::misc04 top::misc05 top::  
misc06 top::misc07 top::misc08} -output order.rp -append  
1
```

SEE ALSO

`extract_rp_group(2)`
`create_rp_group(2)`
`get_rp_groups(2)`
`remove_rp_groups(2)`
`write_rp_groups(2)`

pack_fp_macro_in_area

Automatically packs macros in the specified area.

SYNTAX

```
status pack_fp_macro_in_area  
[-preferred_edges edge_string]  
[-objects collection]  
[-area list_of_points]
```

Data Types

<i>edge_string</i>	string
<i>collection</i>	collection
<i>list_of_points</i>	list

ARGUMENTS

-preferred_edges *edge_string*

edge_string can be any combination of l (for left), r (right), t (top) and b (bottom). If set, the command will prefer the edges listed in *edge_string*. For example, if *edge_string* is "tlb", the macros will be placed in a "C" shape. Note that specifying no edges is not the same as specifying all edges. For example, if all macros are being pulled toward the bottom by wire length considerations, if no edge is specified, macros will end up in several layers along the bottom edge; if all edges are specified, they will fill up all the edges, thus producing an "O" shape. By default (if the option is omitted), no edges are specified.

-objects *collection*

The behavior depends on the collection.

If the collection is empty (or the option is not supplied) plan group macros will be packed inside the corresponding plan group boundaries; top level macros will be packed in the remaining area. If a plan group boundary is outside core, macros belonging to that plan group will be ignored. Note that if there are no plan groups, all macros will be packed in the core area.

If the collection consists of plan groups, macros belonging to those plan groups will be packed inside the plan group boundaries. Top level macros and macros belonging to other plan groups (i.e. plan groups not in the collection) will be ignored.

If the collection consists of macros, the macros in the collection will be packed in the rectangle/polygon determined by the **-area** option.

-area *list_of_points*

Specifies a polygon/rectangle that is the boundary of the area to pack in. If only two points are specified, they are assumed to be lower left and upper right of a rectangle; else the list of points is assumed to be the list of points in a polygon.

If this option is not specified, the command will use the core boundary.

EXAMPLES

The following example will pack macros belonging to plan group named group1 inside its boundary.

```
prompt> pack_fp_macro_in_area -objects "group1"
```

The following example will pack selected macros in the rectangle with lower left in {450 450} and upper right in {1500 1500}.

```
prompt> pack_fp_macro_in_area -objects [get_selection] \  
        -area {{450 450} {1500 1500}}
```

The following example will pack macro named macro1 in the polygon determined by the list of points following the -area argument.

```
prompt> pack_fp_macro_in_area -objects [get_cells "macro1"] \  
        -  
area {{450 450} {2000 450} {2000 1000} {1200 1000} {1200 2200} {450 2200}}
```

DESCRIPTION

Automatically pack macros along edges of the chosen area.

place_flip_chip_array

Creates flip-chip bumps and places them in a two-dimensional matrix configuration.

SYNTAX

```
status place_flip_chip_array
-physical_lib_cell {phy_lib_cels}
-prefix prefix
-start_point start_point
-number num_bump
-delta {x y}
-repeat {i j}
[-orientation N | W | S | E | FN | FE | FS | FW]
[-cell_origin lower_left | center]
```

Data Types

<i>phy_lib_cels</i>	collection
<i>prefix</i>	string
<i>start_point</i>	point
<i>num_bump</i>	integer
<i>x</i>	distance
<i>y</i>	distance
<i>i</i>	integer
<i>j</i>	integer

ARGUMENTS

-physical_lib_cell {*phy_lib_cels*}
Specifies a collection of physical library reference cells. The collection must contain only one reference cell of the flip chip bump cells to be placed.

-prefix *prefix*
Specifies a string prefix to be used when naming bump cells.

-start_point *start_point*
Specifies the lower-left X- and Y-coordinates of the array.

-number *num_bump*
Specifies the number of flip chip bump instances to create and place. The *num_bump* must be greater than 0.

-delta {*x y*}
Specifies the X-pitch and Y-pitch of the bump array in microns.

-repeat {*i j*}
Specifies the X- and Y-dimensions of the array (*i* columns and *j* rows). The *i* and *j* arguments must be positive integers. If *i*j* does not match the specified *num_bump*, the tool chooses the smaller number between *num_bump* and *i*j* to create and place bumps.

-orientation N | W | S | E | FN | FE | FS | FW

Specifies the orientation of the bump cells. Default is N.

-cell_origin lower_left | center

Specifies the origin of the bump cell. Default is lower_left which means that the lower-left corner of a bump is placed at the specified matrix location. If the center option is used, the bump's center is placed at the specified matrix location.

DESCRIPTION

Creates flip chip bumps in the design and places them in a two-dimensional array pattern. The size of the array and the distances between two bumps are specified by the user. A flip chip bump is a special cell that generally consists of the topmost metal layer only. It is used to form electrical connections to the chip package.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates 24 flip chip bumps in an array of 2 by 12, starting at coordinate (100.500, 100.500) with an X-pitch of 1300um and Y-pitch of 200um between adjacent bumps.

```
prompt> place_flip_chip_array \  
-physical_lib_cell [get_physical_lib_cells BUMP] \  
-prefix "BUMP_CELL_" -number 24 \  
-start_point {100.500 100.500} -delta {1300 200} -repeat {2 12}
```

SEE ALSO

place_flip_chip_ring(2)

place_flip_chip_ring

Creates flip-chip bump cells and places them in a ring configuration.

SYNTAX

```
status place_flip_chip_ring
-physical_lib_cell {cell_name}
-prefix prefix
-number num_bump
-bump_spacing spacing
-ring_number ring_number
-ring_spacing ring_spacing
-boundary boundary_box
[-left_orientation N | W | S | E | FN | FS | FW | FE]
[-stagger_offset offset]
[-extra_spacing extra_spacing]
[-num_extra_spacing num_extra]
```

Data Types

<i>cell_name</i>	collection
<i>prefix</i>	string
<i>num_bump</i>	integer
<i>spacing</i>	distance
<i>ring_number</i>	integer
<i>ring_spacing</i>	distance
<i>boundary_box</i>	rectangle
<i>offset</i>	distance
<i>extra_spacing</i>	distance
<i>num_extra</i>	integer

ARGUMENTS

```
-physical_lib_cell {cell_name}
    Specify a collection of physical library cell reference. The collection must
    contain only one cell reference of the flip chip bump cells to be placed.

-prefix prefix
    Specifies a prefix string to be used for newly created bump cells.

-number num_bump
    Specifies a positive integer number of bump cells to be created and placed.

-bump_spacing spacing
    Specifies the minimum spacing between adjacent flip chip bump cells of the
    same ring in micron.

-ring_number ring_number
    Specifies a positive integer number of rings to be created.

-ring_spacing ring_spacing
    Specifies the spacing between adjacent rings in micron.
```

`-boundary boundary_box`
 Specifies the lower-left coordinate {llx lly} and up-right coordinate {urx ury} of the bounding box of the outermost ring in the format of {{llx lly} {urx ury}}.

`-left_orientation N | W | S | E | FN | FS | FW | FE`
 Specifies the orientation of bumps placed on the left edge of the ring. The top, right and bottom edge have an additional 90, 180 and 270 degrees of rotation respectively. Default is the N orientation.

N	rotate 0
W	rotate 90
S	rotate 180
E	rotate 270
FN	rotate 0 and mirror X
FS	rotate 0 and mirror Y
FW	rotate 90 and mirror X
FE	rotate 90 and mirror Y

`-stagger_offset offset`
 Specifies the stagger offset value between the first bump cells of two adjacent rings. The default is zero.

`-extra_spacing extra_spacing`
 Specifies additional spacing for a group of bump cells in micron. This option must be used together with `-num_extra_spacing` option to specify, for example, an addition spacing of 10um for every 5 bump cells. By default, there will be no additional spacing.

`-num_extra_spacing num_extra`
 Specifies the size of bump cell group for which additional spacing is inserted. This option should be used together with `-extra_spacing` option to specify, for example, an addition spacing of 10um for every 5 bump cells. By default, there will be no additional spacing. The `num_extra` should be a positive integer.

DESCRIPTION

The `place_flip_chip_ring` command creates flip chip bumps in the design and places them in a ring configuration. A flip chip bump is a special cell consists of a continuous metal pin placed on the topmost layer metal only. It is used to form electrical connection to the corresponding package bump on the chip packaging.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates 24 flip chip bumps and places them in 2 rings. The bump spacing is 10um and the ring spacing is 15um with the outer ring at coordinates (100,100), (900,900).

```
prompt> place_flip_chip_ring -physical_lib_cell {BUMP} -prefix "BUMP_CELL_"  
-number 24 bump_spacing 10 -ring_number 2 -ring_spacing 15 -boundary {{100  
100} {900 900}}
```

SEE ALSO

`place_flip_chip_array(2)`

place_fp_pins

Performs pin assignment for soft macros or at the block level.

SYNTAX

```
status place_fp_pins  
[-block_level]  
[-effort low | high]  
[-output_feedthroughs {nets | blocks}]  
[-verbose]  
[soft_macros]
```

ARGUMENTS

-block_level

Performs placement of terminals on currently opened cell, considering the connections within the cell. If cell placement is not done the command will attempt to distribute the terminals with equal spacing along each cell edge. Pin physical constraints (see `set_pin_physical_constraints`) for the terminals are honored, if specified and enabled (see `set_fp_pin_constraints`). By default, this option is not in effect and the command performs soft macro pin placement from the top-level, considering the top-level connections and cell placement inside soft macros.

-effort low | high

Specifies the method used to determine pin assignment. If you specify low effort (the default), a flyline-based algorithm is used. For top-level pin assignment (**-block_level** not specified), this algorithm considers all top-level connections for a soft macro pin when choosing its placement location. For block-level pin assignment (**-block_level** specified), if cell placement has been done, the algorithm considers all connections within the cell when choosing a terminal's location; if cell placement has not been done, the command will attempt to distribute the terminals with equal spacing along each cell edge. If you specify high effort, a global route-based algorithm is used. For top-level pin assignment, soft macro pins are placed guided by global routing, which takes into account cell and route congestion, along with connectivities within soft macros and between soft macros on the top block. For block-level pin assignment, terminals are placed guided by global routing, which takes into account cell and route congestion, along with connectivities within the cell. NOTE: If you require feedthrough creation (top-level pin assignment only), you must use high effort.

-output_feedthroughs {nets | blocks}

This option causes the **place_fp_pins** command to output the names of feedthrough nets across soft macros or feedthrough ports on soft macros into a text file or a set of text. The values for this option are "nets" and "blocks". If "nets" is specified, a file called `feedthrough_nets`, which lists all nets with feedthroughs is created. Note that the names are of the original nets that have been split into multiple components. Also listed with the nets are the corresponding blocks on which the feedthrough ports have been

created. Therefore, an entry in `feedthrough_nets` has the following format:
<net> <block1> <block2>

If "blocks" is specified, a number of feedthrough port files are created. A feedthrough port file is created for each soft macro on which feedthrough ports have been created and pins have been placed for these ports. It is named in the format <sm>.feedthrough_ports, where <sm> is the name of a soft macro instance.

The feedthrough port file contains a list of created feedthrough ports on the soft macro. Related feedthrough ports are grouped together. The feedthrough ports are printed out in the format <sm>/<port_name>. Nets and blocks can be specified at the same time. In all cases, if an identically named file is already present in the current working directory, a warning is issued and the existing file is overwritten. An error will be issued and the file skipped if it cannot be overwritten; the command will neither abort nor error out. The `-output_feedthroughs` option lists only the feedthrough ports and nets created by the current run of the **place_fp_pins** command. Preexisting feedthroughs, including those created by previous runs of the **place_fp_pins** command, are not included.

This option is not effective and a warning is issued if the `-block_level` option is also used. In addition, it is not effective for blocks which are not feedthrough enabled (i.e. the `set_fp_pin_constraints -allow_feedthrough on` option has not been previously applied to the blocks).

`-verbose`

Prints more information during command execution.

`soft_macros`

Specifies the soft macros on which to perform pin assignment.

If you do not specify this argument, pin assignment is performed on all soft macros in the current design.

DESCRIPTION

This command either places terminals along top cell boundary or soft macro pins along soft macro boundaries, depending on whether you specify the `-block_level` option. This command supports both rectangular and rectilinear soft macros.

For top-level pin assignment, the tool considers the top-level connections to plan groups, macros, and pads when determining soft macro pin locations. Pin assignment data are stored in both the top-level design and child soft macro cells -- if you open a soft macro cell after pin assignment to view the result and then close it without saving it, the pins are discarded.

For block-level pin assignment, the tool considers the internal cell placement and connections when determining the terminal locations. Terminals are assigned along top cell boundary.

Pin physical constraints are honored, if specified and enabled (see `set_fp_pin_constraints` and `set_pin_physical_constraints`).

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses top-level pin assignment to assign pins to soft macros macroA and macroB using the global route-based method.

```
prompt> place_fp_pins -effort high [get_cells "macroA macroB"]
```

SEE ALSO

```
set_fp_pin_constraints(2)  
set_pin_physical_constraints(2)
```

place_freeze_silicon

Automatically places new cell instances by swapping out spare cells. This command is typically used in a freeze silicon ECO flow.

SYNTAX

status **place_freeze_silicon**

DESCRIPTION

The **place_freeze_silicon** command automatically places cell instances added during ECO. It does not move previously placed cells. Only new cells added by ECO are placed. ICC replaces the new cell instances with existing spare cells of the same reference library cell, or of the same gate_array_master_type attribute. First, ICC runs a fast placement of the new cell instances and then places each new instance at its closest matching spare cell.

An ECO routing command can then be used to complete the routing. The freeze silicon capability can be used to implement metal-only ECO changes an existing layout.

PREREQUISTES

The ECO changes need to be applied before running this command. The **read_mw_eco_list** or **update_mw_design_eco** commands can be used for this purpose. Set matching spare library cell and eco library cell to have the same gate_array_master_type.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

SEE ALSO

set_attribute(2)
read_mw_eco_list(2)
route_eco(2)

place_io_pads

Adjusts the placement of unconstrained pins and pads.

SYNTAX

```
status place_io_pads
[-io_style io_style | flipchip | center | core | distribute | abut | old]
[-adjust_core no_change | macro_and_region | offset_from_pads]
[-left_to_core float]
[-right_to_core float]
[-top_to_core float]
[-bottom_to_core float]
[-min_pad_height]
```

ARGUMENTS

-io_style *io_style* | flipchip | center | core | distribute | abut | old
Specifies the style to be used in distributing the I/O pads. The default is *distribute*. Below is a list of supported IO styles
netweighted - Optimizes I/O pads based on the net length and net weight.
flipchip - Optimize the location of the Flip Chip driving cell, based on the corresponding bump.
center - Pulls pads to the center of each side of the core area.
core - Distributes pads evenly on each side within the area defined by the core.
distribute - Distributes pads evenly on each side of the core area.
abut - Distributes pads evenly on each side of the core and places the pads next to each other without space between them. Pads are placed abut on the left or the right and on the top or the bottom of each other.
old - Keeps the pads locations the same as in the initial floorplan that meet placement constraints without biasing.

-adjust_core *no_change* | macro_and_region | offset_from_pads
Specifies the method you want to use in determining the core area. The default is *no_change*.
no_change - Uses the existing core area.
macro_and_region - Adjusts the core area to the minimum size required that encloses all the macros and cell regions.
offset_from_pads - Adjusts the core area while maintaining offsets from the pads.

-left_to_core *float*
Specifies the distance between the core and the pads on the left side of the core area. The tool uses this value to adjust the cell boundary.

-right_to_core *float*
Specifies the distance between the core and the pads on the right side of the core area. The tool uses this value to adjust the cell boundary.

-top_to_core *float*
Specifies the distance between the core and the pads on the top of the core area. The tool uses this value to adjust the cell boundary.

`-bottom_to_core float`
Specifies the distance between the core and the pads on the bottom of the core area. The tool uses this value to adjust the cell boundary.

`-min_pad_height`
Sets the core area based on the minimum pad height.

DESCRIPTION

This command adjusts the placement of unconstrained pins and pads. Optionally, this command also allows you to adjust the size of the core area and the size of the cell boundary.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example runs the command in default settings.

```
prompt> place_io_pads
1
```

place_opt

Performs simultaneous placement, routing, and optimization on the design.

SYNTAX

```
status place_opt
[-effort low | medium | high]
[-area_recovery]
[-optimize_dft]
[-congestion]
[-power]
[-cts]
[-num_cpus number_of_cpus]
```

Data Types

number_of_cpus int

ARGUMENTS

- effort low | medium | high**
Specifies the effort level for **place_opt**. If you specify **high**, **place_opt** spends more time to further improve the quality of results (QoR). If you specify **low**, **place_opt** spends less time on improving the quality of results (QoR). The default effort level is **medium**.
- area_recovery**
Enables area recovery for the cells not on the timing critical paths. By default, this option is off.
- optimize_dft**
Enables placement aware scan reordering. The reordering is based on SCANCHAINS section of input DEF file, and aims to reduce the scan chain wirelength, minimize congestion and improve routability. Prior to using this option you must first use the **read_def** command to read in scan chain information of the design.
- congestion**
Enables congestion removal algorithms for improved routability. By default, this option is off.
- power**
Enables different power optimization in **place_opt**. The power optimization features are determined by the command's **set_power_options**. By default the tool runs leakage power optimization when you use the **-power** option. This option can be combined with all other current options to **place_opt**. This includes the **-area_recovery** option and the **-effort (low | medium | high)** option. By default, this option is off.
- cts**
Enables clock tree compilation, optimization, and routing within the **place_opt** flows. It allows the optimization engine to work with propagated

clock network delays. When you specify this option, it expects to have all clock related constraints specified before invoking the **place_opt** command.

-num_cpus *number_of_cpus*

Specifies the number of CPUs used in parallel during coarse placement. The *number_of_cpus* is an integer value that is less than or equal to the number of free CPUs on your machine and greater than or equal to . The default is **1**.

DESCRIPTION

This command performs simultaneous placement, routing, and optimization on the current design. The output of this command is a legally placed netlist. If **-cts** is specified, then the output also includes fully synthesized, optimized, routed and extracted clock network.

Multicorner-Multimode Support

This command uses information from active scenarios only.

EXAMPLES

The following example runs the **place_opt** command in timing-driven mode.

```
prompt> place_opt
```

SEE ALSO

```
clock_opt(2)  
create_buffer_tree(2)  
create_placement(2)  
extract_rc(2)  
identify_clock_gating(2)  
legalize_placement(2)  
psynopt(2)  
route_opt(2)  
set_power_options(2)  
set_place_opt_cts_strategy(2)
```

place_opt_feasibility

Performs placement optimization feasibility study on the design.

SYNTAX

```
status place_opt_feasibility  
[-congestion]  
[-stage initial_placement | drc | first_opt]  
[-skip initial_placement | drc | first_opt]  
[-optimize_dft]  
[-power]  
[-num_cpus number_of_cpus]
```

Data Types

number_of_cpus int

ARGUMENTS

- congestion**
Enables congestion removal to improve routability. By default, this option is off.
- stage initial_placement | drc | first_opt**
Specifies to which stage the command will run. The internal stages of the command include initial placement (**initial_placement**), fixing HFS and logic DRC violations (**drc**), timing optimization (**first_opt**), congestion removal if **-congestion** is specified, and timing optimization if **-congestion** is specified.
If **-stage initial_placement** is specified, the command finishes at the initial placement stage. If **-stage drc** is specified, the command finishes at the fixing DRC violation stage. By default, the command runs all the necessary stages.
- skip initial_placement | drc | first_opt**
Specifies which stage the command will skip. See the **-stage** option for the internal stages.
If **-skip initial_placement** is specified, the command skips the initial placement stage and start from the fixing DRC violations stage. If **-skip drc** is specified, the command skips both the initial placement and fixing DRC violation stages and starts from the timing optimization stage. By default, the command skips no stage.
- optimize_dft**
Enables scan chain reordering if scan chains exist.
- power**
Enables leakage power optimization.
- num_cpus *number_of_cpus***
Specifies the number of CPUs used in parallel during coarse placement. The *number_of_cpus* is an integer value that is less than or equal to the number

of free CPUs in your machine.

DESCRIPTION

The command performs simultaneous placement and optimization for feasibility study on the design. The output of this command is a legally placed netlist. The resulting database can be used for any subsequent flow in IC Compiler.

You can specify the **-congestion** option if the design is congested and needs to remove congestion. The runtime will increase because of the additional congestion removal and timing optimization steps.

You can use the **-stage** option to run command for only a few steps and save the database for analysis purpose.

Use the **-skip** option to skip steps to save the runtime. For example, specify **-stage initial_placement** to run the initial placement and then save the placed design for analysis. Later, you use **-skip initial_placement** to run the rest of the steps.

You should use **place_opt_feasibility** at early stage of incompleting constraints. This command runs faster than the **place_opt** command with the same settings. After running **place_opt_feasibility** command, you can use the **create_qor_snapshot** generate categorized timing report, which helps identify most common timing violations, such as large I/O delays, large macro delays, and so forth. You can use **create_qor_snapshot -zero_wire_load** to generate ZWL (Zero-Wire-Load) timing report. If you see ZWL timing violations after running the **place_opt_feasibility** command, those violations are likely infeasible paths.

The resulting database from this command is valid for any subsequent optimization process, such as clock tree synthesis and post optimization.

EXAMPLES

The following example runs the **place_opt_feasibility** command with congestion removal.

```
prompt> place_opt_feasibility -congestion
prompt> #ZWL report#
prompt> create_qor_snapshot -name myrun -max_paths 10 -zero_wire_load
prompt> #Normal report#
prompt> create_qor_snapshot -name myrun -max_paths 10
prompt> report_qor_snapshot
prompt> save_mw_cel -as myrun
prompt> #subsequent flow from here#
```

SEE ALSO

place_opt(2)
create_qor_snapshot(2)
report_qor_snapshot(2)

preroute_instances

Connects pins in instances to power and ground targets.

SYNTAX

```
status preroute_instances
[-ignore_macros]
[-ignore_pads]
[-ignore_cover_cells]
[-consider_driver_cells]
[-connect_instances all_but_specified | specified | specified]
[-cells {collection_of_cells}]
[-target_directions four_sides | vias_only | vias_only]
[-skip_left_side]
[-skip_right_side]
[-skip_bottom_side]
[-skip_top_side]
[-skip_vias_to_lower_layer]
[-skip_vias_to_higher_layer]
[-
select_net_by_type pg | tieup_and_tiedown | pg_and_tieup_tiedown | specified | tieu
p_and_tiedown | pg_and_tieup_tiedown | specified]
[-nets {collection_of_nets}]
[-route_pins_on_layer layer_from_tech_file]
[-primary_routing_layer preferred | pin | specified | pin | specified]
[-preferred_routing_layer high | low]
[-specified_horizontal_layer h_layer]
[-specified_vertical_layer v_layer]
[-pad_pin_to_pad_boundary distance]
[-one_d_pin one_d_pin_dist]
[-boundary_pad_to_top_cell_boundary distance]
[-macro_pin_to_macro_boundary distance]
[-customize]
[-routing_width connection_width]
[-connect_to_pins_at center | low_end | high_end | low_end | high_end]
[-offset pin_connection_offset]
[-extend_to_boundaries_and_generate_pins]
[-force_extend_to_boundaries_and_generate_pins]
[-route_small_pins_using_wider_dimension]
[-route_boundary_coinciding_edges_only]
[-skip_pad_pins_touching_pad_side_boundaries]
[-advanced_via_rules]
[-special_rules special_rules_name]
[-extend_for_multiple_connections]
[-extension_gap space_threshold]
[-undo]
```

Data Types

<i>collection_of_cells</i>	collection
<i>collection_of_nets</i>	collection
<i>layer_from_tech_file</i>	string
<i>h_layer</i>	string

preroute_instances

<i>v_layer</i>	string
<i>one_d_pin_dist</i>	float
<i>connection_width</i>	float
<i>pin_connection_offset</i>	float
<i>special_rules_name</i>	string
<i>space_threshold</i>	float

ARGUMENTS

-ignore_macros
Prevents connection of pins in macro cells.

-ignore_pads
Prevents connection of pins in pad cells.

-ignore_cover_cells
Prevents connection of pins in cover cells.

-consider_driver_cells
Connects pins in flip-chip driver cells.

-connect_instances *all_but_specified* | *specified* | *specified*
Specifies the instances to connect as **all_but_specified** or **specified**. You can use the **-cells** option to specify names, if necessary. The default is **all_but_specified**.

-cells {*collection_of_cells*}
Specifies a collection of instances to use for the **-connect_instances** option.

-target_directions *four_sides* | *vias_only* | *vias_only*
Specifies how to connect pins. Use the **four_sides** argument to connect pins in the directions of the four cell sides. Use the **vias_only** argument to connect pins only by dropping vias to the upper-layer and lower-layer targets. The default is **four_sides**.

-skip_left_side
Selects to skip the left side of the instance you do not want connected when you use the **-target_directions** option with the **four_sides** argument.

-skip_right_side
Selects to skip the right side of the instance you do not want connected when you use the **-target_directions** option with the **four_sides** argument.

-skip_bottom_side
Selects to skip the bottom side of the instance you do not want connected when you use the **-target_directions** option with the **four_sides** argument.

-skip_top_side
Selects to skip the top side of the instance you do not want connected when you use the **-target_directions** option with the **four_sides** argument.

-skip_vias_to_lower_layer
Prevents the dropping of vias to the lower layer, when you use the **-target_directions** option with the **vias_only** argument.

-skip_vias_to_higher_layer

Prevents the dropping of vias to the higher layer, when you use the **-target_directions** option with the **vias_only** argument.

-select_net_by_type pg | tieup_and_tiedown | pg_and_tieup_tiedown | specified | tieup_and_tiedown | pg_and_tieup_tiedown | specified

Specifies the type of nets to connect instances, using one of four valid values. The default is **pg**.

-nets {collection_of_nets}

Specifies nets to be used for the **-select_net_by_type** option. If there is more than one net in the collection, separate each one by a space.

-route_pins_on_layer layer_from_tech_file

Connects only pins on specified layer. Valid values for the *layer_from_tech_file* argument are **layerNumber** or **maskName** from the technology file. The default is to connect pins on all layers.

-primary_routing_layer preferred | pin | specified | pin | specified

Specifies how to pick the layer on which to route. The valid arguments are **preferred**, **pin**, and **specified**. Use **preferred** to route on preferred metal layers. You can refine this choice by specifying the **-preferred_routing_layer** option. Use **pin** to route on the same layer on which the pins are located. Use **specified** to route on layers specified by the **-specified_horizontal_layer** and **-specified_vertical_layer** options.

-preferred_routing_layer high | low

Specifies whether to route first on lower from pin, or higher from pin preferred metal layer when you use the **-primary_routing_layer** option with the **preferred** argument. Valid arguments are: **high** and **low**. Specify **high** to route on the next available higher preferred layer. If a higher preferred layer is not available, route on an available lower preferred layer. Specify **low** to route on the next available lower preferred layer. If a lower preferred layer is not available, route on an available higher preferred layer.

-specified_horizontal_layer h_layer

Specifies the horizontal layers to route when you use the **-primary_routing_layer** option with the **specified** argument. Valid values for the *layer_from_tech_file* argument are **layerNumber** or **maskName** from the technology file. The default is the first horizontal layer.

-specified_vertical_layer v_layer

Specifies the vertical layers to route when you use the **-primary_routing_layer** option with the **specified** argument. Valid values for the *layer_from_tech_file* argument are **layerNumber** or **maskName** from the technology file. The default is the first vertical layer.

-pad_pin_to_pad_boundary distance

Specifies the nearness ratio of pin to pad boundary. Pins routed by the **preroute_instances** command must be near the boundary across which they are to be routed. The *nearness* test is determined by the bounding box of a pin and pad and the corresponding *routing_ratio*. You can assume that pads are rectangular (but remember that if they are not, their bounding boxes are used). For example, for a left boundary pad, the command routes all pins to the right if the distance between a pin's right edge and that of the pad is

less than 0.1 times the distance between the left and right edges of the pad. The default 0.1.

Note that for macro cells, only "small" pins are considered. Small pins are those with x-dimension and y-dimension that both fail the `-one_d_pin` pin test.

`-one_d_pin one_d_pin_dist`

Specifies the nearness ratio of either the x-dimension and y-dimension of a pin to macro cell dimension. If exactly one dimension of the bounding box of a multi-rectangle pin is greater than or equal to 0.75 times the corresponding dimension of the macro, the pin is a 1-D pin. A 1-D pin is routed along its length, unless you turn on the **-customize** option to switch to routing along its width. The default is 0.75.

`-boundary_pad_to_top_cell_boundary distance`

Specifies the nearness ratio of a pad to the top cell boundary. For example, a pad near the top cell's left boundary is a boundary pad if the distance between its left edge and the top cell's left boundary is less than or equal to 0.05 times the distance between its left and right edges. Non-boundary pads are treated as macros. When a boundary pad is routed toward the core, only one side is routed. Corner pads are boundary pads that pass this test for two top cell boundaries, and they are not routed toward the core. Pad rings can be generated only for boundary pads. The default is 0.05.

`-macro_pin_to_macro_boundary distance`

Specifies the nearness ratio of a small pin to a macro boundary. The default is 0.1.

`-customize`

Customizes connections to `one_d_pin` pins.

`-routing_width connection_width`

Specifies the width of the connection to the macro or pad pins when using the **-customize** option. You might need to customize connections if you have 1-D pins.

`-connect_to_pins_at center | low_end | high_end | low_end | high_end`

Specifies where to connect wires to pin when using the **-customize** option. Valid values are **center**, **low_end**, and **high_end**. Use *center* to connect to the center of the pins. Use *low_end* to connect to the left or bottom end of the pins. Use *high_end* to connect to the right or top end of the pins. The default is **center**.

`-offset pin_connection_offset`

Specifies the distance by which you want to offset the pin connection from the center, low end, or high end when using the **-customize** option. The default is 0.0.

`-extend_to_boundaries_and_generate_pins`

Creates a power and ground extension wire to the cell boundary and generates a power and ground pin for the extension wire, if the top cell does not have pads. The pin is marked as fixed to prevent it from being moved by placement and routing operations. To extend wires, the following conditions must be met:

- The power and ground pin cannot be already connected to power and ground in the direction where the **preroute_standard_cells** command would create the extension wire.
- Creating the power and ground extension wire does not cause a design rule violation.

-force_extend_to_boundaries_and_generate_pins

Forces execution of the **-extend_to_boundaries_and_generate_pins** option, even if the power and ground pin is already connected to power and ground.

-route_small_pins_using_wider_dimension

Specifies for the **preroute_instances** prerouter to always connect to the wider side of a small pin. A small pin is routed only if it is close enough to at least one boundary. If so, the pin is routed toward and, most likely, out of the closest boundary to the first target and the connection is made to the corresponding side of the pin. As a result, the prerouter might connect to the narrower or the wider side of a small pin. This option directs it to the wider side.

-route_boundary_coinciding_edges_only

Controls exactly the widths and locations of the connections to the pins. This option forces the prerouter to connect only the edges of the pins that are co-linear with a boundary, using the edge width of the pin.

-skip_pad_pins_touching_pad_side_boundaries

Skips those pad pins that touch the side boundaries of pads.

-advanced_via_rules

Uses advanced via rules that have been previously set.

-special_rules *special_rules_name*

Refers to a set of special rules specified by the **set_preroute_special_rules** command. The value of the *special_rules_name* argument must be exactly the same as the string specified in the **-name** option of that command.

-extend_for_multiple_connections

Extends normal pin connections to reach more targets on the same net. The extension continues until either of the following conditions are true:

- There are no more targets in the respective directions.
- The next target exceeds the spacing threshold set by using the **-extension_gap** option.

-extension_gap *space_threshold*

Specifies the space threshold for multiple connections. Adjacent targets closer than the value of the *space_threshold* argument are connected together.

-undo

Removes connections created by the last run of the **preroute_instances** command.

DESCRIPTION

This command connects pins in macros and pads to trunks or other pins on the same net. The width of the connection is the same as the width of the originating pin, unless otherwise changed. You can change the internal application of design rule checking (DRC) rules by using the **set_preroute_drc_strategy** command.

Prerequisites

Specify port-to-net connections for power and ground nets that are not in the netlist by using the **derive_pg_connection** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example preroutes all macro, pad, and cover instances, except those specified in the **-cells** option, with nets VDD and VSS, only for pins on layer M3 only.

```
prompt> preroute_instances\  
-connect_instances all_but_specified\  
-cells {hier_1/i_ram0/i_testA/uMEM hier_1/i_ram7/i_testB/uMEM}\  
-nets {VDD VSS}\  
-route_pins_on_layer M3
```

The following example connects left and right I/O pads with a ring following special rules specified by the **set_preroute_special_rules** command.

```
prompt> preroute_instances\  
-nets VDD\ -ignore_macros\  
-ignore_cover_cells -skip_bottom_side\  
-skip_top_side -primary_routing_layer pin\  
-route_pins_on_layer PM\  
-special_rules specIOhor
```

See the **set_preroute_special_rules(2)** man page, *4th example*) for more detail.

SEE ALSO

```
create_pad_rings(2)  
create_power_straps(2)  
create_preroute_vias(2)  
create_rectangular_rings(2)  
derive_pg_connection(2)  
preroute_standard_cells(2)  
report_preroute_drc_strategy(2)  
set_preroute_advanced_via_rule(2)
```

```
set_preroute_drc_strategy(2)
set_preroute_special_rules(2)
verify_pg_nets(2)
```

preroute_standard_cells

Connects power and ground pins in the standard cells to the power and ground rings or straps, and connects power and ground rails in the standard cells.

SYNTAX

```
status preroute_standard_cells
[-mode rail | tie | net]
[-connect horizontal | vertical | both]
[-nets {collection_of_nets}]
[-route_pins_on_layer layer]
[-within {{llx lly} {urx ury}}]
[-no_routing_outside_working_area]
[-special_via_rule]
[-offset_both_sides_for_special_via]
[-special_via_x_offset distance]
[-special_via_y_offset distance]
[-special_via_x_size distance]
[-special_via_y_size distance]
[-special_via_x_step distance]
[-special_via_y_step distance]
[-advanced_via_rules]
[-start_share_via_stack layer]
[-skip_macro_pins]
[-skip_pad_pins]
[-remove_floating_pieces]
[-pin_width_by_extreme_edges]
[-pin_width_by_most_extended_pin]
[-tie_mode_max_route_width distance]
[-extend_to_boundaries_and_generate_pins]
[-force_extend_to_boundaries_and_generate_pins]
[-avoid_merging_vias]
[-optimize_via_locations]
[-snap_shapes_to_tracks]
[-do_not_route_over_macros]
[-fill_empty_rows]
[-cut_out_empty_spans]
[-extend_for_multiple_connections]
[-extension_gap distance]
[-max_fanout int_number]
[-h_layer name_of_metal_layer]
[-v_layer name_of_metal_layer]
[-h_width distance]
[-v_width distance]
[-port_filter pattern]
[-cell_master_filter pattern]
[-cell_instance_filter pattern]
[-voltage_area_filter pattern]
[-port_filter_mode off | select | skip]
[-cell_master_filter_mode off | select | skip]
[-cell_instance_filter_mode off | select | skip]
[-voltage_area_filter_mode off | select | skip]
[-undo]
```

[-fill_empty_sites]

ARGUMENTS

-mode rail | tie | net

Specifies one of three modes for standard cell connections:

- * **rail** to connect standard cells by rails
 - * **tie** to connect standard cells by tying all the power and ground pins to nearby targets
 - * **net** to connect the pins of standard cells to a nearby power ring or strap by creating Steiner tree with restrictions (see options -max_fanout, -h_layer, -v_layer, -h_width, -v_width below).
- The default is **rail**.

-connect horizontal | vertical | both

Specifies the direction of the power and ground connections: **horizontal** | **vertical** | **both**.

The default is **horizontal**.

-nets {collection_of_nets}

Specifies the nets used to connect standard cells.

By default, power and ground nets are used.

-route_pins_on_layer layer

Specifies to connect pins on the specified metal layer only. You can specify the value either as the layerNumber or the maskName from the technology file.

By default, all pins on all layers are connected.

-within {{llx lly} {urx ury}}

Specifies the area within which to connect pins.

By default, the top cell boundary box is used.

-no_routing_outside_working_area

Prevents standard cell connections from extending beyond the specified working area. The default is false, so that if this option is skipped, connections can extend beyond the area specified in the **-within** option.

-special_via_rule

Creates a matrix of small vias instead of one large via at wire intersections. Using smaller vias could free some routing space. You must specify the x and y offsets, size, and step by using the **-offset_both_sides_for_special_via**, **-special_via_x_offset**, **-special_via_y_offset**, **-special_via_x_size**, **-special_via_y_size**, **-special_via_x_step**, and **-special_via_y_step** options.

-offset_both_sides_for_special_via

Applies the x- and y-offsets to both the lower left and upper right corners of the wire intersection area.

By default, the offsets are applied only to the lower left corner of the wire intersection area.

-special_via_x_offset distance

-special_via_y_offset distance

Specifies the x- and y-offset from the lower left corner of the wire

intersection area where you want to create the first special via.
The default is 0.0.

`-special_via_x_size distance`

`-special_via_y_size distance`

Specifies the size of the special vias. The number you type is the x- or y-length in your units.

`-special_via_x_step distance`

`-special_via_y_step distance`

Specifies the x- and y-distance between the special vias. The distance is in user units, between two left sides of adjacent via cuts in the x-direction, and between two bottom sides of adjacent via cuts in the y-direction. When there are design rule violations, some vias might not be created.

`-start_share_via_stack layer`

Sets the request to share upper vias of two neighboring via stacks if they produce DRC violations. Vias, located on layers above the specified metal layer, should be common to both stacks. You can specify the value either as the layerNumber or the maskName from the technology file. This option may be specified along with the **-mode rail** option only. By default, no shared vias are allowed.

`-skip_macro_pins`

Specifies not to connect standard cells to macro pins.

`-skip_pad_pins`

Specifies not to connect standard cells to pad pins.

`-remove_floating_pieces`

Specifies to remove floating (unconnected) rail segments. They are kept by default.

`-pin_width_by_extreme_edges`

Determine pin widths by the width of the pin at its extreme edges.

`-pin_width_by_most_extended_pin`

Determine pin widths by the width of the most extended pin.

`-tie_mode_max_route_width distance`

Specifies the maximum routing width for **tie** mode. The default of 0.0 means no limit.

`-extend_to_boundaries_and_generate_pins`

Creates a power and ground extension wire to the cell boundary and generates a power and ground pin for the extension wire, if the top cell does not have pads. The pin is marked as fixed to prevent it from being moved by placement and routing operations. To extend wires, the following conditions must be met:

- a) The power and ground pin is not already connected to power and ground in the direction where **preroute_standard_cells** would create the extension wire.
- b) Creating the power and ground extension wire does not cause a design

rule violation.

`-force_extend_to_boundaries_and_generate_pins`

Forces the **-extend_to_boundaries_and_generate_pins** option, even if the power and ground pin is already connected to power and ground.

`-avoid_merging_vias`

Prevents the merging of existing vias.

`-optimize_via_locations`

Adjusts the vias by small distances horizontally or vertically so that more tracks are saved for detail routing.

`-snap_shapes_to_tracks`

Allows you to request that all newly created shapes are placed (snapped) onto tracks of appropriate metal layers.

This option can be used only if the standard cell pins are connected using **-mode net**. If a connection cannot meet the requirement, the command should leave the pins unconnected.

`-do_not_route_over_macros`

Prevents the prerouter from generating wires over macro cells. By default, this option is off. If you enable it, the tool removes the parts of a route that go over a macro.

`-fill_empty_rows`

Fills all rows, even those without cells, with power and ground rails.

`-cut_out_empty_spans`

Prevents creation of rails between neighboring power straps or rings of the net if no standard cell pins of the net are found in that area.

`-extend_for_multiple_connections`

Extends the low and high ends of the standard cell connections to reach more targets on the same net. The extension continues until either there are no more targets in the respective directions, or the next target exceeds the spacing threshold set in the **-extension_gap** option. This option is off by default.

`-extension_gap distance`

Specifies the space threshold for multiple connections. Adjacent targets closer than this value are connected together.

`-max_fanout int_number`

Specifies the maximum number of pins in a tree connecting them to a ring or strap (for **-mode net** only). The default is 10.

`-h_layer name_of_metal_layer`

Specifies the routing layer to be used for horizontal segments (for **-mode net** only). If the layer is not set (or is not a routing layer), the command sets the layer automatically.

`-v_layer name_of_metal_layer`

Specifies the routing layer to be used for vertical segments (for **-mode net** only). If the layer is not set (or is not a routing layer), the command sets

the layer automatically.

-h_width *distance*

Specifies the width of horizontal segments (for **-mode net** only). If the width is 0.0 (the default value), the command uses a width of the thinnest pin in a set. If the width does not correspond to the maximum or minimum width of the layer, it is adjusted automatically.

-v_width *distance*

Specifies the width of vertical segments (for **-mode net** only). If the width is 0.0 (the default value), the command uses a width of the thinnest pin in a set. If the width does not correspond to the maximum or minimum width of the layer, it is adjusted automatically.

-port_filter *pattern*

Specifies list of patterns to be tested against a pin name to decide whether it should be routed. Each pattern can contain a regular expression. Patterns must be separated by commas. Default is **.***

-cell_master_filter *pattern*

Specifies list of patterns to be tested against a reference cell name to decide whether its pins should be routed. Each pattern can contain a regular expression. Patterns must be separated by commas. Default is **.***

-cell_instance_filter *pattern*

Specifies list of patterns to be tested against a cell name to decide whether its pins should be routed. Each pattern can contain a regular expression. Patterns must be separated by commas. Default is **.***

-voltage_area_filter *pattern*

Specifies list of patterns to be tested against the name of the voltage area where a cell placed to decide whether its pins should be routed. Each pattern can contain a regular expression. Patterns must be separated by commas. Default is **.***

-port_filter_mode *off | select | skip*

Specifies one of three modes for the port filter:

- * **off** ignores the filter (this is the default mode)
- * **select** selects a pin for routing if its name corresponds to at least one pattern of the filter
- * **skip** excludes a pin from routing if its name corresponds to at least one pattern of the filter.

-cell_master_filter_mode *off | select | skip*

Specifies one of three modes for the cell master filter:

- * **off** ignores the filter (this is the default mode)
- * **select** selects pins of a cell for routing if the name of its reference corresponds to at least one pattern of the filter
- * **skip** excludes pins of a cell from routing if the name of its reference corresponds to at least one pattern of the filter

-cell_instance_filter_mode *off | select | skip*

Specifies one of three modes for the cell instance filter:

- * **off** ignores the filter (this is the default mode)
- * **select** selects pins of a cell for routing if its name

- corresponds to at least one pattern of the filter
- * **skip** excludes pins of a cell from routing if its name corresponds to at least one pattern of the filter

`-voltage_area_filter_mode off | select | skip`

Specifies one of three modes for the voltage area filter:

- * **off** ignores the filter (this is the default mode)
- * **select** selects pins of the cell for routing if the name of the voltage area in which it is placed corresponds to at least one pattern of the filter
- * **skip** excludes pins of a cell from routing if the name of the voltage area in which it is placed corresponds to at least one pattern of the filter

`-undo`

Removes connections created by the last **preroute_standard_cells** run.

DESCRIPTION

The **preroute_standard_cells** command connects power and ground pins in the standard cells to the power and ground rings or straps, and connects power and ground rails in the standard cells.

You can change the internal application of DRC rules by using the **set_preroute_drc_strategy** command.

Prerequisites:

Specify port-to-net connections for power and ground nets that are not in the netlist by using the **derive_pg_connection** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

To preroute standard cells in both direction by rail (the default mode) with the VDD and VSS nets, for pins on the metal 3 layer only, within the boundaries of (10.0 10.0) (1400.0 1400.0), enter

```
prompt> preroute_standard_cells\  
-nets {VDD VSS} -connect both\  
-route_pins_on_layer M3\  
-within {{10.0 10.0} {1400.0 1400.0}}
```

SEE ALSO

```
set_preroute_drc_strategy(2)  
report_preroute_drc_strategy(2)  
derive_pg_connection(2)  
create_rectangular_rings(2)  
create_preroute_vias(2)
```

preroute_standard_cells
1300

```
create_pad_rings(2)
preroute_instances(2)
create_power_straps(2)
verify_pg_nets(2)
```

print_message_info

Prints information about diagnostic messages which have occurred or have been limited.

SYNTAX

```
string print_message_info  
[-ids id_list] [-summary]
```

Data Types

id_list list

ARGUMENTS

-ids *id_list*

List of message identifiers to report. Each entry can be a specific message or a glob-style pattern which matches one or more messages. If this option is omitted and no other options are given, then all messages which have occurred or have been limited will be reported.

-summary

Generate a summary of error, warning and informational messages which have occurred so far.

DESCRIPTION

The **print_message_info** command enable you to print summary information about error, warning, and informational messages which have occurred or have been limited with the **set_message_info** command. For example, if the following message is generated, information about it is recorded.

```
Error: unknown command 'wrong_command' (CMD-005)
```

It is useful to be able to summarize all recorded information about generated diagnostic messages. Much of this can be done using the **get_message_info** command but you need to know a specific message id. By default, **print_message_info** will summarize all of the information. It will provide a single line for each message which has occurred or has been limited, and one summary line which shows the total number of errors, warnings, and informational messages which have occurred so far. If an *id_list* is given, then only messages matching those patterns will be displayed. If **-summary** is given, then a summary will be displayed.

Using a pattern in the *id_list* is intended to show a specific message prefix, for example, "CMD*". Note that this will not show all messages with that prefix. Only those which have occurred or have been limited will be displayed.

EXAMPLES

The following example uses **print_message_info** to show a few specific messages.

```
print_message_info  
1302
```

```
prompt> print_message_info -ids [list "CMD*" APP-99]
```

Id	Limit	Occurrences	Suppressed
CMD-005	0	7	2
APP-99	1	0	0

At the end of the session, you might want to generate some information about a set of interesting messages, such as how many times each occurred (which includes suppressions), how many times each was suppressed, and whether a limit was set for any of them. The following shows how to use `print_message_info` in this way.

```
prompt> print_message_info
```

Id	Limit	Occurrences	Suppressed
CMD-005	0	12	0
APP-027	100	150	50
APP-99	0	1	0

Diagnostics summary: 12 errors, 150 warnings, 1 informational

Note that the suppressed count is not necessarily the difference between the limit and the occurrences, since the limit can be dynamically changed with **`set_message_info`**.

SEE ALSO

```
get_message_info(2)  
set_message_info(2)  
suppress_message(2)
```

print_proc_new_vars

Check for new variables created within a Tcl procedure.

SYNTAX

string **print_proc_new_vars**

DESCRIPTION

The **print_proc_new_vars** command is used in a Tcl procedure to show new variables created up to that point in the procedure. If the **sh_new_variable_message_in_proc** and **sh_new_variable_message** variables are both set to true, this command is enabled. If either variable is false, the command does nothing. The result of the command is always empty.

When enabled, the command will issue a CMD-041 message for each variable created in the scope of the procedure so far. Arguments to the procedure are excluded.

For procedures which are in a namespace (that is, not in the global scope), **print_proc_new_vars** also requires that you have defined some aspect of the procedure using **define_proc_attributes**.

Warning: This feature has a *significant* negative impact on CPU performance. This should be used only when developing scripts or in interactive use. When you turn the feature on, the application issues a message (CMD-042) to warn you about the usage of this feature.

EXAMPLES

Given that the appropriate control variables are true, the following commands show new variables created within procedures:

```
prompt> proc new_proc {a} {                set x $a
      set y $x
      unset x
      print_proc_new_vars
}
prompt> new_proc 67
=New variable report for new_proc:
Information: Defining new variable 'y'. (CMD-041)
=END=
prompt>
prompt> proc ns::p2 {a} {                  set x $a
      print_proc_new_vars
}
prompt> define_proc_attributes ns::p2
prompt> ns::p2 67
=New variable report for ns::p2:
Information: Defining new variable 'x'. (CMD-041)
=END=
```

SEE ALSO

`define_proc_attributes(2)`
`sh_new_variable_message(3)`
`sh_new_variable_message_in_proc(3)`

print_suppressed_messages

Displays an alphabetical list of message ids that are currently suppressed.

SYNTAX

string **print_suppressed_messages**

DESCRIPTION

The **print_suppressed_messages** command displays all messages that you suppressed using **suppress_message**. The display lists in alphabetical order. You only can suppress informational and warning messages. The result of **print_suppressed_messages** is always the empty string.

EXAMPLES

This example shows the output from **print_suppressed_messages**.

```
prompt> print_suppressed_messages
No messages are suppressed
prompt> suppress_message {XYZ-001 CMD-029 UI-1}
prompt> print_suppressed_messages
The following 3 messages are suppressed:
    CMD-029, UI-1, XYZ-001
```

SEE ALSO

`suppress_message(2)`
`unsuppress_message(2)`

printenv

Prints the value of environment variables.

SYNTAX

```
string printenv  
[variable_name]
```

Data Types

variable_name string

ARGUMENTS

variable_name
Optional name of a single environment variable to print.

DESCRIPTION

Prints the values of the environment variables inherited from the parent process or set from within the application with **setenv**. With no arguments, all environment variables are listed. With a single *variable_name*, if that variable exists, its value is printed. There is no useful return value from **printenv**.

To retrieve the value of a single environment variable, use **getenv**.

EXAMPLES

These examples show the output from **printenv**.

```
prompt> printenv SHELL  
/bin/csh  
prompt> printenv EDITOR  
emacs
```

SEE ALSO

getenv(2)
setenv(2)

printvar

Prints the values of one or more variables.

SYNTAX

```
string printvar  
[pattern] [-user_defined] [-application]
```

Data Types

pattern string

ARGUMENTS

pattern

Variable names which match *pattern* will be printed. The optional *pattern* argument can include the wildcard characters "*" and "?". If omitted, all variables are printed.

-user_defined

Show only user-defined variables. If combined with -application, the behavior is as though neither option was used.

-application

Show only application variables. If combined with -user_defined, the behavior is as though neither option was used.

DESCRIPTION

The **printvar** command prints the values of one or more variables. If the *pattern* argument is not specified, the command prints out the values of all the variables (both application and user defined variables). The -user_defined option limits the variables to those which you have defined. The -application option limits the variables to those created by the application. If both options are used, they cancel each other out.

Some variables are suppressed from the output of **printvar**. For example, the Tcl environment variable array *env* is not shown. In order to see the environment variables, use the **printenv** command. Also, the Tcl variable *errorInfo* is not shown. To see the error stack, use the **error_info** command.

EXAMPLES

The following command prints the values of all the variables.

```
prompt> printvar
```

The following command prints the values of all application variables that match the pattern sh*a*.

```
prompt> printvar -application sh*a*  
sh_arch           = "sparcOS5"  
sh_command_abbrev_mode = "Anywhere"  
sh_enable_page_mode   = "false"  
sh_new_variable_message = "false"  
sh_new_variable_message_in_proc = "false"  
sh_source_uses_search_path = "false"
```

The following command prints the value of the variable `search_path`.

```
prompt> printvar search_path  
search_path       = ".  /designs/newcpu/v1.6  /lib/cmos"
```

The following command prints the value of the user defined variables.

```
prompt> printvar -user_defined  
a                 = "6"  
designDir          = "/u/designs"
```

SEE ALSO

proc_args

Displays the formal parameters of a procedure.

SYNTAX

```
string proc_args  
proc_name
```

Data Types

```
proc_name      string
```

ARGUMENTS

```
proc_name  
    Specifies the name of the procedure.
```

DESCRIPTION

The **proc_args** command is used to display the names of the formal parameters of a user defined procedure. This command is essentially a synonym for the Tcl builtin command **info** with the *args* argument.

EXAMPLES

This example shows the output of **proc_args** for a simple procedure.

```
prompt> proc plus {a b} {return [expr $a + $b]}  
prompt> proc_args plus  
a b  
prompt> info args plus  
a b  
prompt>
```

SEE ALSO

```
info(2)  
proc_body(2)
```

proc_body

Displays the body of a procedure.

SYNTAX

```
string proc_body  
proc_name
```

Data Types

```
proc_name      string
```

ARGUMENTS

```
proc_name  
    Specifies the name of the procedure.
```

DESCRIPTION

The **proc_body** command is used to display the body (contents) of a user defined procedure. This command is essentially a synonym for the Tcl builtin command **info** with the *body* argument.

EXAMPLES

This example shows the output of **proc_body** for a simple procedure.

```
prompt> proc plus {a b} {return [expr $a + $b]}  
prompt> proc_body plus  
    return [expr $a + $ b]  
prompt> info body plus  
    return [expr $a + $ b]  
prompt>
```

SEE ALSO

```
info(2)  
proc_args(2)
```

process_particle_probability_file

Encrypts or decrypts the particle probability function file.

SYNTAX

process_particle_probability_file

-key
-input_file
[-output_file]

ARGUMENTS

- key encryption_key
This is the key used to encrypt/decrypt the particle probability function file. The value of string could be a combination of alphabets and numbers. It should be at least 8 characters long. The key specified for encryption of a file must be identical to that specified for decryption. This command fails if they do not match. It is a mandatory option.
- input_file filename
Specify the particle probability file name. If input file is in ascii format, output is an encrypted file. If input file is an encrypted file, output is an ascii file.
- output_file filename
Specify the output file. To specify this is optional. If it is not specified, this name is derived by appending "_out" to the input file name.

DESCRIPTION

This command encrypts or decrypts the particle probability function file. This encryption protects the content of the particle probability file. An encrypted particle probability file can be specified as input to the **report_critical_area** command. There is no need to specify the key for that command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command performs encryption of particle probability function file.

```
prompt> process_particle_probability_file \  
-key abc123456 -input_file "datafile.txt" \  
-output_file "datafile.encr"
```

SEE ALSO

`report_critical_area(2)`

propagate_constraints

Propagates timing constraints from lower levels of the design hierarchy to the current design.

SYNTAX

```
status propagate_constraints
[-design design_list]
[-all]
[-clocks]
[-disable_timing]
[-dont_apply]
[-false_path]
[-gate_clock]
[-ideal_network]
[-ignore_from_or_to_port_exceptions]
[-ignore_through_port_exceptions]
[-max_delay]
[-min_delay]
[-multicycle_path]
[-operating_conditions]
[-power_supply_data]
[-output file_name]
[-port_isolation]
[-verbose]
[-case_analysis]
[-target_library_subset]
```

Data Types

<i>design_list</i>	list
<i>file_name</i>	string

ARGUMENTS

- design *design_list***
Specifies a list of names of lower-level designs from which constraints are to be propagated. The constraints are propagated from every instance of the designs in this list. By default, constraints are propagated from every lower-level design in the hierarchy.
- all**
Specifies to propagate clocks, **gate_clock** checks, **false_path** exceptions, **multicycle_path** exceptions, **max_delay** and **min_delay** exceptions, instance specific operating conditions, power supply data, and **disable_timing** attributes. This is the default. Issuing the command without options has the same effect as specifying the **-all** option.
- clocks**
Specifies to propagate only clocks previously created by using the **create_clock** command. Virtual clocks are not propagated. By default, this option is off.

- disable_timing**
Specifies to propagate only **disable_timing** attributes previously set by using the **set_disable_timing** command. By default, this option is off.
- dont_apply**
Specifies that the propagated constraints are not to be applied to the current design. Warnings are still reported. You can use this option with the **-verbose** option to see the effect of running the command, without actually running it. If you do not use the **-verbose** option, the command displays warnings regarding conflicting exceptions. By default, this option is off.
- false_path**
Specifies to propagate only **false_path** exceptions previously specified by using the **set_false_path** command. By default, this option is off.
- gate_clock**
The **propagate_constraints -gate_clock** option will be obsolete in a future release. Clock-gating insertion with **compile_ultra -gate_clock** automatically propagates clock gate setup and hold time.
Specifies to propagate upwards only setup and hold clock gating checks previously specified by using the **set_clock_gating_check** command. RTL clock gating automatically imposes setup and hold time constraints during elaboration. By default, this option is off.
- ideal_network**
Specifies to propagate ideal networks previously created by using the **set_ideal_network** command. By default, this option is off.
- ignore_from_or_to_port_exceptions**
Specifies to propagate exceptions for ports on *from_lists* and/or *to_lists* of commands that set timing constraints. By default, the command converts from port and to port exceptions into through exceptions and propagates them to the current design. By default, this option is off.
- ignore_through_port_exceptions**
Specifies not to propagate exceptions for ports on *through_lists* of commands that set timing constraints. By default, these exceptions are propagated.
- max_delay**
Specifies to propagate only **max_delay** exceptions previously specified by using the **set_max_delay** command. By default, this option is off.
- min_delay**
Specifies to propagate only **min_delay** exceptions previously specified by using the **set_min_delay** command. By default, this option is off.
- multicycle_path**
Specifies to propagate only **multicycle_path** exceptions previously specified by using the **set_multicycle_path** command. By default, this option is off.
- operating_conditions**
Specifies to propagate only **operating_conditions** previously specified by using the **set_operating_condition** command. It propagates the operating conditions specified on instances as well as those specified on lower level sub-designs. By default, this option is off.

-power_supply_data
 Specifies to propagate only power intent data, including UPF (supply network, power state table, and various power intent strategies) and operating voltages on the supply nets. By default, this option is off.

-output *file_name*
 Specifies to write the command output to file specified by *file_name*. By default, this option is off.

-port_isolation
 Specifies to propagate exceptions previously set by the **set_isolate_ports** command. By default, this option is off.

-verbose
 Echoes equivalent commands that are applied to the current design by using the **propagate_constraints** command. Following each propagated constraint, the name enclosed in */* */* is the name of the lower-level design from which the constraint is propagated. By default, this option is off.

-case_analysis
 Specifies to propagate user case values previously set by using the **set_case_analysis** command. By default, this option is off.

-target_library_subset
 Specifies to propagate target library subsets that have been specified by using the **set_target_library_subset** command. It propagates the target library subset specified on instances as well as those specified on subdesigns. By default, this option is off.

DESCRIPTION

This command collects constraints from instances of lower-level designs and applies those constraints to the current design for use during timing analysis and compilation. Once the constraints are propagated, they can be removed, overridden, or reported like any other constraints that are set on the current design. Issuing the command without options is the same as issuing it with the **-all** option.

The command produces a warning in some cases where it does not propagate constraints because of one of the following conditions:

- Clock name conflict: multiple clocks from different designs have the same name.
- Clock source conflict: an object is specified as a clock source of more than one clock.
- Clock exception from/to unpropagated clock: if a clock is not propagated, exceptions from or to the clock are not propagated. The command does not propagate a clock if you do not use the **-clock** option, if the clock is a virtual clock, or if propagating the clock creates a conflict.

The command does not propagate a constraint that creates a conflict and is echoed in comments using the object names in the current design.

You must propagate timing **gated_clock** checks before compilation if automatic clock

gating has been performed during elaboration.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example propagates only setup and hold time constraints, from all lower-level designs to the current design:

```
prompt> propagate_constraints -gate_clock
```

The following example propagates only **multicycle_path** exceptions from the *SUB1* subdesign to the current design:

```
prompt> propagate_constraints -design SUB1 -multicycle_path
```

The following example propagates clocks, exceptions, **gated_clock** checks, and **disable_timing** arcs to the current design from all instances of lower-level designs. The output is shown in non-verbose mode. Clock warnings are issued in this mode.

```
prompt> propagate_constraints -all
```

```
/******  
  
Created by propagate_constraints() on Thu Jun 24 06:02:01 1999  
  
*****/  
  
/* Propagate Constraints from cell mid_left/ (MID_LEFT) */  
  
/* Propagate Constraints from cell mid_right/ (MID_RIGHT) */  
  
/* Propagate Constraints from cell mid_right/in_left/ (IN_LEFT) */  
  
/* Propagate Constraints from cell mid_right/in_right/ (IN_RIGHT) */  
  
/** Warning: Clock clk was already found in design: OUTIE */  
/** The following clock definition in design IN_RIGHT*/  
/** is ignored :*/  
/* create_clock -period 10 -waveform {0 5} find(port,"clk")  
*/  
/** Warning: clock exception(s) in design IN_RIGHT is(are) ignored */  
/** because clock clk_virtual1 was not propagated */  
/** Warning: The following clock exceptions in design IN_RIGHT are ignored */  
/* set_false_path <HardSpace*/  
/* -from find(clock,"clk_virtual1") <HardSpace*/  
/* -to find(pin,"mid_right/in_right/F1/D") */  
*/ /* IN_RIGHT */
```

The following example displays all constraints that would have been propagated by the command, but does not apply them to the current design. The output is shown in verbose mode.

```
prompt> propagate_constraints -all -verbose -dont_apply
```

```
*****/
Created by propagate_constraints() on Thu Jun 24 06:02:02 1999
*****/

/* Propagate Constraints from cell mid_left/ (MID_LEFT) */

/** Warning: clock exception(s) in design MID_LEFT is(are) ignored */
/** because clock virtual_clkm was not propagated */
/** Warning: The following clock exceptions in design MID_LEFT are ignored */
/* set_false_path
   -to find(clock,"virtual_clkm")
*/ /* MID_LEFT */

/* Propagate Constraints from cell mid_right/ (MID_RIGHT) */

set_false_path
  -from find(pin,"mid_right/in_left/F2/CP")
  -to find(pin,"mid_right/in_right/F2/D")
/* MID_RIGHT */
set_disable_timing find(pin,"mid_right/in_right/F2/Q") /* MID_RIGHT */

/* Propagate Constraints from cell mid_right/in_left/ (IN_LEFT) */

create_clock -name "clkin_1" -period 10
-waveform {0 5} find(pin,
"mid_right/in_left/clk
") /* IN_LEFT */
set_false_path
  -to find(clock,"clkin_1")
/* IN_LEFT */
set_false_path
  -through {find(pin,"mid_right/in_left/i1")
find(pin,"mid_right/in_left/clk")}

  -to find(clock,"clkin_1")
/* IN_LEFT */
set_false_path
  -through find(pin,"mid_right/in_left/i1")
  -to find(pin,"mid_right/in_left/F1/D")
/* IN_LEFT */
set_disable_timing find(cell,"mid_right/in_left/F1") /* IN_LEFT */
set_disable_timing find(cell,"mid_right/in_left/F2")
-from "CP" -to "Q" /* IN_LEFT */

/* Propagate Constraints from cell mid_right/in_right/ (IN_RIGHT) */
```

```

/**** Warning: Clock clk was already found in design: OUTIE */
/**** The following clock definition in design IN_RIGHT*/
/**** is ignored :*/
/* create_clock -period 10 -waveform {0 5} find(port,"clk")
*/
/**** Warning: clock exception(s) in design IN_RIGHT is(are) ignored */
/**** because clock clk_virtual1 was not propagated */
/**** Warning: The following clock exceptions in design IN_RIGHT are ignored */
/* set_false_path
   -from find(clock,"clk_virtual1")
   -to find(pin,"mid_right/in_right/F1/D")
*/ /* IN_RIGHT */
1

```

SEE ALSO

```

create_clock(2)
report_timing_requirements(2)
set_clock_gating_check(2)
set_disable_timing(2)
set_false_path(2)
set_ideal_network(2)
set_max_delay(2)
set_min_delay(2)
set_multicycle_path(2)
set_operating_conditions(2)
set_target_library_subset(2)

```

propagate_ilm

Propagates specified information of the interface logic model (ILM) blocks to the top-level design.

SYNTAX

```
status propagate_ilm  
[-clock_mesh_annotation]  
[-verbose]  
[cell_list]
```

ARGUMENTS

-clock_mesh_annotation

Propagates clock mesh annotations from ILM to top-level. This option is used in the hierarchical clock-mesh flow using ILMs.

-verbose

Turn verbose messaging on.

cell_list

List of ILM blocks for which information must be propagated. Each cell in the list must be hierarchical ILM cell at any level in the logical hierarchy of the current design.

DESCRIPTION

This command propagates specified information for ILM blocks to the current design. Issuing the command without any options is considered as legacy style usage and interpreted as user intention to propagate keepout, placement, delay from ILMs. Since keepout, placement and delay information is auto-propagated, tool will not take any action in such a usage with no options specified.

The current design before executing this command must be the top design. This command links the current design before propagating the information.

-clock_mesh_annotation

Propagates clock mesh annotations from ILM to top-level. The clock mesh annotations are stored during block-level clock_mesh flow when analyze_subcircuit command is run. The same annotation is retained in the ILM during create_ilm command and is propagated to TOP level when this option is used. The clock mesh annotation data can be MCMM scenario specific. Therefore the propagate_ilm command is recommended to be run after activating required scenarios if the design has MCMM scenarios. Once propagated, the clock_mesh annotation gets stored in the top-level design when the design is saved to Milkyway. Subsequently, when the saved design is opened, the earlier propagated clock_mesh annotation will get restored. Therefore there is no need to run this command with this option after each time the design is opened. Refer to Hierarchical Clock Mesh related documentation in IC Compiler User Guide for further details on usage of this feature within the whole clock-mesh flow.

Multicorner-Multimode Support

This command uses information from active scenarios only.

SEE ALSO

`propagate_constraints(2)`
`read_sdc(2)`
`report_annotated_delay(2)`
`report_cell(2)`

propagate_switching_activity

Forces a propagation of the power switching activity information.

SYNTAX

```
int propagate_switching_activity  
[-effort low | medium | high]  
[-verbose]  
[-infer_related_clocks]
```

ARGUMENTS

-effort low | medium | high

Specifies the effort level used during switching activity propagation. The tool uses more randomly generated switching activities to propagate the switching activity on higher effort levels. When this option is not specified, the value of the **power_sa_propagation_effort** variable is used as the effort level.

-verbose

Specifies that the switching activity mechanism is run in verbose mode, displaying displays more information and warning messages

-infer_related_clocks

Specifies that only related clock information is to be inferred. When this option is used, static probability and toggle rate information is not propagated; but related clock information is inferred for design objects that request a related clock.

DESCRIPTION

The **propagate_switching_activity** command forces a propagation of the power switching activity information.

During switching activity propagation, the information annotated using the **read_saif**, **set_switching_activity**, **merge_saif** commands is used to estimate the activity on unannotated nets. All commands that need switching activity information on all design objects use the switching activity propagation mechanism automatically. In most cases, you do not need to explicitly call the **propagate_switching_activity** command. The **propagate_switching_activity** command allows you to access propagated switching activity values directly without using a command such **report_power**, which propagates the switching activity as a side-effect.

The mechanism to propagate switching activity uses the static probability and toggle rate information you annotate to estimate the static probability and toggle rate information on unannotated design objects.

The mechanism used by the tool is based on a stochastic simulation method, which is that random activity based on the annotated switching activity is generated on the annotated design objects, and this activity is propagated across the design using a zero delay simulator. This is repeated for a few thousand times, depending on the

effort level.

The **-infer_related_clocks** option is used to restrict the action of the **propagate_switching_activity** command to inferring only the related clock information on design objects on which it was requested. Use the **-clock ""** argument of the **set_switching_activity** command to specify that the related clock on the specified design object will be inferred automatically by the tool. Related clock information is inferred when commands that need switching activity information (such as **report_power**) are used. Use the command with the **-infer_related_clocks** option to infer only the related clock information. The related clock is placed in the **related_clock** attribute on the design objects.

The method in which the related clock information is described in the steps below.

First, the tool determines a starting set of inferred related clocks using the following rules:

- The inferred related clock on an object with a user set related clock, is the user set related clock.
- The inferred related clock on a clock source port or pin is the clock.
- If the **power_rclock_inputs_use_clocks_fanout** variable is set to **true**, then primary input nets that do not have an inferred related clock according to the rules above, then the following applies. If the transitive fanout of the input contains a clock network object, then the inferred related clock on the input is the clock driving the network. A clock network is the transitive fanout of a clock port or pin, stopping and including sequential cells. If more than one of this type of a clock object exists, then the fastest clock of this type is chosen as the related clock.

The related clocks are then inferred on design objects by the following rules:

- For flip-flop cells, if the **power_rclock_use_async_inputs** variable is set to **false**, then the inferred related clock on the cell's outputs is the inferred related clock on the cell's clock pin. If **power_rclock_use_async_inputs** is set to **true**, then the inferred related clock on the cell's outputs is the fastest inferred related clock on the cell's clock pin and the cell's asynchronous input pins.
- The inferred related clock on clock-gating cell outputs is the inferred related clock on the cell's clock input pin.
- For non-flip-flop and non-clock-gating cells, the inferred related clock on the cell's outputs is the fastest related clock on the cell's inputs.

The inferred related clocks are then made to be the design object's related clock for objects that requested a related clock. The inferred related clock information on design objects that did not request a related clock is discarded. For design objects that requested a related clock, but no clock was inferred by the above rules, the fastest design clock is set as the related clock if the **power_rclock_unrelated_use_fastest** variable is set to **true**.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example propagates the power switching activity information with high effort:

```
prompt> propagate_switching_activity -effort high
```

SEE ALSO

```
merge_saif(2)
read_saif(2)
report_saif(2)
reset_switching_activity(2)
set_switching_activity(2)
power_sa_propagation_effort(3)
power_sa_propagation_verbose(3)
power_rclock_inputs_use_clocks_fanout(3)
power_rclock_unrelated_use_fastest(3)
power_rclock_use_asynch_inputs(3)
```

psynopt

Performs incremental synthesis on the design.

SYNTAX

```
int psynopt
[-area_recovery]
[-only_area_recovery]
[-congestion]
[-no_design_rule | -only_design_rule]
[-only_hold_time]
[-in_place_size_only]
[-size_only]
[-on_route]
[-preserve_footprint]
[-use_annotation]
[-power]
[-only_power]
```

ARGUMENTS

- area_recovery**
Enables area recovery for the cells not on the timing critical paths. By default, area recovery is not enabled.
- only_area_recovery**
Specifies that only area recovery is performed. Timing optimizations and design rule checking are not performed. This option cannot be used with the **-only_design_rule** or the **-only_hold_time** option.
- congestion**
Enables congestion removal algorithms for improved routability. By default, congestion is not enabled.
- no_design_rule | -only_design_rule**
Specifies whether or not the tool fixes design rule violations before exiting. If **-no_design_rule** is specified, the tool exits before fixing design rule violations. You can check the results in a constraint report before fixing the violations. If **-only_design_rule** is specified, the tool performs only design rule fixing, and timing optimizations are not performed. The default is to perform both design rule fixing and timing optimizations before exiting.
The **-no_design_rule** and **-only_design_rule** options are mutually exclusive.
- only_hold_time**
Specifies that only hold time violations are fixed. This option cannot be used with the **-no_design_rule** option. Use the **set_fix_hold** command to perform hold time fixing.
- in_place_size_only**
Restricts optimization to sizing changes only. Optimization procedures that insert new cells and remove cells are disabled. With the **-in_place_size_only**

option, sizing changes are further constrained for minimal engineering change order (ECO) placement changes. For example, a cell is sized to improve timing or design rule costs only if the newly-sized cell can fit into any available space adjacent to the original cell location. The resulting transformation is verified to ensure it is legal.

The **-in_place_size_only** option is recommended for postroute flow, so optimization changes are constrained for minimal ECO changes. The **psynopt** command with the **-in_place_size_only** option is to be run as a final step after the design is finalized and legal, in order to provide extra quality of results (QoR) without causing significant ECO changes. Since the optimization on the entire design is highly constrained, significant QoR changes is not to be expected with **-in_place_size_only**.

-size_only

Restricts optimization to sizing changes only. Optimization tricks that insert new cells and remove cells are disabled. The **-size_only** and **-in_place_size_only** options are mutually exclusive. See the description for the **-in_place_size_only** option for additional information on how to constrain sizing changes more.

-on_route

Invokes the on-route optimization flow. RC extraction is performed automatically, if the design is not extracted. Optimization is restricted to sizing on free space and buffering on the existing route.

-preserve_footprint

Specifies to perform only footprint preserving sizing. Footprint preserving sizing is done by using footprint attribute added by vendors. When the tool sizes the cell, it checks the number of pins is the same, the name of pins is the same and the footprint attribute is the same as well as the size of the cell is the same. However, the tool does not check the physical pin location is the same and the tool assumes it is done by library vendor when it added footprint attribute.

-use_annotation

Invokes back annotation based optimization flow.

-power

Enables power optimization in **psynopt**. The type of power optimization will be determined by the command **set_power_options**. If the command **set_power_options** is not specified, then by default the tool will run leakage power optimization.

-only_power

Enables only power optimization in **psynopt**. The type of power optimization will be determined by the command **set_power_options**. If the command **set_power_options** is not specified, then by default the tool will run leakage power optimization.

DESCRIPTION

The **psynopt** command performs incremental preroute or postroute synthesis on the current design. The output of this command is a legally placed netlist.

Multicorner-Multimode Support

This command uses information from active scenarios only.

EXAMPLES

The following example runs the **psynopt** command in preroute flow:

```
prompt> psynopt
```

SEE ALSO

```
create_buffer_tree(2)  
create_placement(2)  
legalize_placement(2)  
place_opt(2)  
set_power_options(2)
```

push_down_fp_objects

Transfers physical objects from the top level to the soft macro level.

SYNTAX

```
status push_down_fp_objects
[-object_type {routing | route_guides | blockages | cells | rows | shapes}]
[-cut_type push_down | cut_down | copy_down]
[-no_overlap_checking]
[-connect_copy_down_wires]
[-wire_net_type {pg | clock | signal}]
[-freeze_push_down_nets]
[-allow_feedthroughs]
[-layers layer_object_list]
[-objects object_list]
[-to_object_type blockage | original_type]
[-partial_overlap]
[-remove_routing]
[-row_offset_x int]
[-row_offset_y int]
[-include_shapes all | with_net_id | without_net_id]
[soft_macros]
```

Data Types

<i>layer_object_list</i>	collection
<i>object_list</i>	collection
<i>soft_macros</i>	collection

ARGUMENTS

-object_type {routing | route_guides | blockages | cells | rows | shapes}
Specifies the types of object to be pushed down. You can specify more than one object type. If you do not specify this option, the default object type is routing.
The descriptions for the supported object types are as follows:

- **routing**
This object type includes wires, paths, vias and pins. If a net is connected to these objects at the top level, but does not exist in the soft macro, child net will be created in the soft macro. If a top level terminal is overlapped with soft macro and has a top level net connecting to it, the terminal will be copied into child cell as a soft macro pin with the same shape of terminal. If you are pushing down power and ground nets in a design with multiply-instantiated modules (MIMs), this command leaves gaps between the top-level power and ground strap and the PG pins for MIM instances that are not aligned with the MIM master instance and do not have power and ground rings around them. These gaps enable later power and ground routing to connect the PG pins. The tool also handles partial routes pushing down for the limited situations where partial routes are physically completely inside the soft macro boundary and no new child level ports need to be created.

- `route_guides`

This object type includes rectangular route guides and their corresponding attributes.

- `blockages`

This object type includes all rectangular blockages on metal blockage layers and soft or hard placement blockages.

- `cells`

This object type includes all standard cells within the boundary of the specified soft macros. If a net is connected to these cells at the top level, but does not exist in the soft macro, the net is created in the soft macro.

- `rows`

This object type includes all cell rows that touch the specified soft macros.

- `shapes`

This object type includes all rectangular and rectilinear shapes that touch the specified soft macros on routing layers. Shape means metal segment here, and the rectangular placement blockage is handled as **blockages** object type. Net shapes, wires, paths, vias and pins are handled as **routing** object type. If you specify shapes in **-object_type**, all rectangles and polygons with or without net id will be included.

-cut_type `push_down` | `cut_down` | `copy_down`

Specifies how the command should process the pushed-down objects. You can specify only one type. If you do not specify this option, the **push_down** method is used.

If you specify **push_down**, the tool creates a copy of the object in the soft macro and deletes the object from the top level. The tool also creates pins for routing objects where the routing object touches the soft macro boundary. This method applies to the following object types: routing, route guides, blockages, shapes, and cells. For rows, **push_down** behaves the same as **copy_down**. The wires or paths passing through the soft macro area are removed from the parent cell and moved into the child cell. However, when the wires or paths are partially overlapped with the cell boundaries and the wire type is PG net, they are not removed from the parent cell. Instead, a copy is created within the child cell and pins are created at the edge of the child cell boundary where the copied wires and paths end. If wires or paths are partially overlapped with the cell boundaries and the wire type is not PG net, **push_down** skips this net. If you push down a PG net and the net does not have a connection to a soft macro, the push down method creates a new input PG port on the soft macro and creates pins for this new port. The push down method processes tie-high and tie-low nets the same way as PG nets, except it skips those nets that have colinear overlaps with the child cell boundary. If you specify **cut_down**, the tool creates pins for routing objects where the routing object touches the soft macro boundary and deletes the object from the top level. For routing objects, route guides, blockages, shapes and cells, **cut_down** behaves the same as **push_down**. For rows, **cut_down** behaves the same as **copy_down**. For power and ground net cut down, the tool also generates a Tcl file named `childCellName_PGStrap.tcl` which create power straps for the design.

If you specify **copy_down**, the tool creates a copy of the object in the soft macro, but keeps the object in the top level. Unlike **push_down** and **cut_down**, the tool doesn't create pins for power and ground nets during **copy_down**.

-no_overlap_checking

Pushes down the objects without checking for routing overlaps. This means that routing overlaps are ignored, even if they cause shorts. This option is only valid for routing object type.

-connect_copy_down_wires

Connects wires that are pushed down to their nets. This option should be used with the **copy_down** cut type. By default, the **copy_down** cut type does not connect the routings to the child net newly created inside the soft macro. For other cut types, push down connects routings to their nets automatically. In addition, this option is valid only for the routing object type and does not impact other object types.

-wire_net_type {pg | clock | signal}

Specifies what types of nets to process. You can specify more than one type of net. If you do not specify this option, pg (power, ground, tie-high, and tie-low) nets and signal nets are processed.

Push down routing does not push down routing objects of signal and clock nets that are colinear with the cell boundary. If you see the warning message of skipping the routes colinear with cell boundary, please re-route and possibly apply route guides to avoid routes going colinear with the boundaries.

This option is ignored if **routing** is not included in **-object_type**.

-freeze_push_down_nets

Sets the frozen attribute on the pushed down nets that have routing at the top level before pushing down. With this option, the router will not touch the routing objects that being pushed down. In addition, this option also sets the **dont_touch** attribute on the pushed down nets to prevent modification or replacement of the pushed down nets during optimization.

This option is valid only for routing object type.

-allow_feedthroughs

Specifies how the command should process physical routings. This option is used only for signal nets and clock nets and is ignored for power nets, ground nets, tie-high nets, and tie-low nets.

If you specify this option, the command creates new ports for feedthrough routings and creates a new net at top level for the split net.

If you do not specify this option, the command does not create new ports for feedthrough routings if the top-level net has a connection with the soft macro. If you do not specify this option and the soft macro has a port connected to the top-level net, pins are created at each crossing point between the soft macro boundary and the physical routings. Multiple pins can be created for one port on the soft macro level. It does not create a new net on the top level for the routing connections between soft macros.

This option affects only routing objects.

-layers *layer_object_list*

Objects that are pushed down are limited to specific layers.

This option is valid if **routing** or **fshapes** is included in **-object_type**.

However, this option is ignored if the **-allow_feedthroughs** option is specified.

-objects *object_list*

Specifies the objects which are to be pushed down. The objects include nets, routing objects (wires, paths, vias, via arrays, and pins), route guides,

blockages, cells, and shapes. For example, if *net_object_list* is included in **-objects**, the routing objects associated with the specified nets are pushed down.

If you do not specify this argument, the command processes all objects whose object types are listed in the **-object_type** option.

If the object type for an object listed in the *object_list* argument is not included in the **-object_type** option, the object is ignored.

-to_object_type blockage | original_type

Specifies the type for objects after they are pushed down to the child level. This option is valid only when **routing** or **cells** are specified in the **-object_type** option and the **-cut_type copy_down** option is used.

The supported to_object types are listed below:

/(bu **blockage**

Creates a placement blockages or route guide in the child level to mirror the top-level cell or routing object.

/(bu **original_type** (the default)

Creates an object in the child level with the same object type as the one at the top level.

-partial_overlap

Pushes down cells that partially overlap the soft macro boundary.

By default, only those cells that are completely within the soft macro boundary are pushed down. In a design, if this option is not specified and the cells overlap with a soft macro, routing blockages are created for the overlapped pins.

This option is ignored if **cells** is not included in **-object_type**.

-remove_routing

Removes the routing of the nets that are connected between pushed down cells. This option only works when the **push_down_type** is **push_down** and the **object_type** is **cells**.

The default behavior is to push down the cells and the routing of the nets between the pushed down cells. If you specify the **-remove_routing** option, the tool does not push the routing down to the child cell. Instead, these routes are removed only from the top level.

This option is ignored if **cells** is not included in the **-object_type**.

-row_offset_x int

Specifies the horizontal offset in microns between the soft macro core and rows.

The default is 0. This option is valid only for the rows object type.

-row_offset_y int

Specifies the vertical offset in microns between the soft macro core and rows.

The default is 0. This option is valid only for the rows object type.

-include_shapes all | with_net_id | without_net_id

Specifies what type of shapes this command should process. If you specify **-include_shapes with_net_id**, only shapes connected to a net are processed; If you specify **-include_shapes without_net_id**, only shapes not on any net are processed. The default value of this option is **all**, which means both shapes with and without net ids are processed.

This option is valid only when **shapes** is included in **-object_type**.

soft_macros

Specifies the soft macros into which to push down the specified objects.
If you do not specify this argument, the command processes all soft macros in the current design.

DESCRIPTION

This command transfers physical objects from the top level to the soft macro level.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example pushes down routing and cell objects for all soft macros in the current design. Wires that are pushed down are connected to their nets.

```
prompt> push_down_fp_objects -object_type {routing cells}
```

The following example pushes down the VDD routing for all soft macros in the current design. Wires that are pushed down are connected to the VDD nets in the soft macros.

```
prompt> push_down_fp_objects -object_type {routing} \  
-wire_net_type pg -objects [get_nets VDD]
```

The following example pushes down the HW#12089 horizontal wire for all soft macros in the current design.

```
prompt> push_down_fp_objects -object_type {routing} \  
-objects [get_net_shape {HW#12089}]
```

SEE ALSO

push_up_fp_objects(2)

push_flip_chip_route

Pushes routed flip-chip wires either in a specified direction or away from the specified nets.

SYNTAX

```
status push_flip_chip_route
-nets collection_of_nets
    | -nets_in_file nets_file
[-layer mX | MX | tech_layer_number]
[-direction up | down | left | right]
[-mode 1 | 2 | 3]
[-sweep_range sweep_range]
[-all]
```

Data Types

<i>collection_of_nets</i>	collection
<i>nets_file</i>	string
<i>sweep_range</i>	integer

DATA TYPES

<i>collection_of_nets</i>	collection
<i>nets_file</i>	string
<i>tech_layer_number</i>	integer
<i>number</i>	integer
<i>sweep_range</i>	integer
<i>mX</i>	string
<i>MX</i>	string

ARGUMENTS

-nets *collection_of_nets*
Specifies the flip-chip nets to be pushed when using modes 1 or 2. For mode 3, this option specifies the nets to push routed wires away from.
The **-net**, **-nets_in_file** options are mutually exclusive. You must specify one of these options.

-nets_in_file *nets_file*
Specifies the name of a file that contains the flip-chip nets to be pushed when using modes 1 or 2. For mode 3, this option specifies the nets to push routed wires away from.
The **-net**, **-nets_in_file** options are mutually exclusive. You must specify one of these options.

-layer mX | MX | tech_layer_number
Specifies the metal layer on which to push the specified flip-chip wires. You can specify only a single metal layer. (m1 to m15) You specify the metal layer by using its technology file layer number or m1 to m15.

By default, the tool pushes the wires on the top metal layer.

`-direction up | down | left | right`

Specifies the direction in which to push the flip-chip nets.

If you do not specify the direction, the tool does not push any wires.

`-mode 1 | 2 | 3`

Specifies the mode in which to push the flip-chip nets. The meanings for each mode are

* Mode 1 (default mode)

Pushes the flip-chip nets in the specified direction.

A wire is pushed until it is totally blocked by a routing obstruction (a metal blockage or wire) or it meets the bounding box of the target routed net. When a wire is blocked, the tool splits the wire and tries to push parts of the wire. For example, assume that a vertical wire A is being pushed to the right, as shown in Figure 1, but blockage B partially blocks the way. When the tool finishes pushing the wire, wire A is split into 3 parts: vertical wire A1, horizontal wire C, and vertical wire A2, as shown in Figure 2.

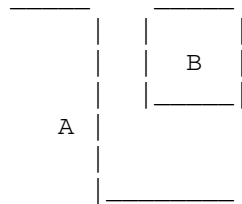


Figure 1

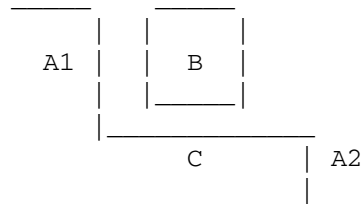


Figure 2

When pushing wires, the tool respects the spacing requirements defined by **set_route_flip_chip_options -layer_spacing**. The

Mode 1 does not push the segment inside of I/O driver cells.

* Mode 2

Pushes the nets as described in mode 1, but allows pushing the segment inside of I/O driver cells.

* Mode 3

Pushes the nearest 2xN nets away from a specific net, where N is specified in the **-sweep_range** option.

Use mode 3 to push the neighboring wires away from an open net so that the net can be routed.

`-sweep_range sweep_range`

Specifies the sweep range value used with mode 3. The default is 10.

DESCRIPTION

This command provides several modes to manually push routed flip-chip wires.

A flip-chip net is a net that connects a flip-chip driver I/O cell to a bump cell (which is also called a flip-chip pad). You can write the flip-chip nets to a file

by using the **write_flip_chip_nets** command.

When the tool pushes a wire, it moves the wire in a certain direction until it is blocked by other obstacles, respecting the spacing requirements defined by **set_route_flip_chip_options -layer_spacing**. The wires remain on their original metal layer and the connectivity is not changed.

EXAMPLES

The following command pushes flip-chip nets NET1 and NET2 on metal layer 57 up using mode 2.

```
prompt> push_flip_chip_route -layer 57 -mode 2 \  
-nets {NET1, NET2} -direction up
```

The following command pushes 40 neighboring nets sway from flip-chip net NET3.

```
prompt> push_flip_chip_route -layer m9 -mode 3 \  
-nets NET3 -sweep_range 20
```

SEE ALSO

```
set_route_flip_chip_options(2)  
route_flip_chip(2)  
create_stack_via_on_pad_pin(2)  
write_flip_chip_nets(2)  
remove_flip_chip_route(2)  
display_flip_chip_route_flylines(2)  
optimize_flip_chip_route(2)
```

push_up_fp_objects

Transfers physical objects from the soft macro level to the top level.

SYNTAX

```
status push_up_fp_objects
[-object_type {routing | pins | route_guides | blockages | cells | rows | shapes}]
[-layers layer_object_list]
[-to_object_type routing | pins]
[-top_level_interface_nets net_object_list]
[-preserve_child_preroutes]
[-remove_routing]
[-include pushed_down_objects_only | all]
[-include_shapes all | with_net_id | without_net_id]
[-child_object_names object_list]
[soft_macros]
[-ignore_not_push_down_nets]
```

Data Types

<i>layer_object_list</i>	collection
<i>net_object_list</i>	collection
<i>object_list</i>	collection
<i>soft_macros</i>	collection

ARGUMENTS

`-object_type {routing | pins | route_guides | blockages | cells | rows | shapes} | rows | shapes}` Specifies the types of object to be pushed up. You can specify more than one object type. If you do not specify this option, the default object type is routing.

The descriptions for the supported object types are as follows:

routing	This object type includes wires, paths, and vias.
pins	This object includes soft and hard macro pins.
route_guides	This object type includes rectangular route guides and their corresponding attributes.
blockages	This object type includes all rectangular blockages on metal blockage layers and soft or hard placement blockages.
cells	This object type includes all standard cells within the boundary of the specified soft macros.
rows	This object type includes all cell rows that touch the specified soft macros. Rows are not pushed up to the top level (the top level already contains the row objects), they are just removed from the soft macros.

shapes This object type includes all rectangular and rectilinear shapes on routing layers. Shape means metal segment here, and the rectangular placement blockage is handled as **blockages** object type. Wires, paths, vias and pins are handled as **routing** object type. If you specify shapes in **-object_type**, all rectangles and polygons with or without net id will be included.

-layers *layer_object_list*

Objects that are pushed up are limited to specific layers.
Use this option with routing or pin objects only.

-to_object_type routing | pins

Specifies the type for objects after they are pushed up to the top level from soft macros. You should use this option when pushing up soft or hard macro pins or routings only.

For soft macro pins, you can push up soft macro pins from the soft macro to the top level as wire routing or as top-level terminals. However, for hard macro pins, this command only supports pushing up hard macro pins to top level terminals.

You can push up child-level routing to top-level routing or to top-level terminals if there are top-level ports on the connected top-level nets. If the child-level routing is pushed up to top-level terminals and the routing is rectilinear, the pushed-up routing is broken into rectangular electrically equivalent (EEQ) pins. If the child-level pin is pushed up to top-level routing and the pin is rectilinear shape, the pushed-up routing will be in rectangular shape.

Here is a description of the supported object types.

routing This object type includes wires, paths, and vias.

pins This object includes top-level terminals.

-top_level_interface_nets *net_object_list*

Specifies a collection of top-level interface nets for which routing or pin objects are pushed up. Only child objects with logical connections to those top-level nets are pushed up.

-preserve_child_preroutes

Do not delete routing objects from the soft macro after creating them at the top level. You should use this option used when pushing up routings

.

-remove_routing

Removes the routings of the nets that are connected between pushed up cells
This option only works when the **object_type** is **cells**.

The default behavior is to push up the cells and the

routing of the nets between the pushed up cells. If you specify the **-remove_routing** option, the tool does not push the routing up to the top cell. Instead, the routes are removed only from the child level.

-include *pushed_down_objects_only* | *all*

Specifies the objects to be pushed up. This option applies only to cells, nets, and routing objects. By default (**pushed_down_objects_only**), only the object types that were created from the **push_down_fp_objects** command are pushed up to the top level. If you specify **all**, the object types that are created solely inside soft macros are also pushed up.

This option is ignored for route guides, blockages, shapes and row object types.
For these object types, the command always pushes up both previously pushed down objects and objects that are created solely inside soft macros.

-include_shapes *all* | *with_net_id* | *without_net_id*

Specifies what type of shapes this command should process. If you specify **-include_shapes with_net_id**, only shapes connected to a net will be processed. If you specify **-include_shapes without_net_id**, only shapes not on any net will be processed. The default value of this option is **all**, which means both shapes with and without net id are processed.

This option is valid only when **shapes** is included in **-object_type**.

-child_object_names *object_list*

Specifies the objects inside soft macros to push up.
The object names must be full hierarchical names. They can be cell instance names or net names.

soft_macros

Specifies the soft macros from which to push up objects.

If you do not specify this argument, objects are pushed up from all soft macros in the current design.

DESCRIPTION

This command transfers physical objects from the soft macro level to the top level.

Any netlist connections made by the **push_down_fp_objects** command are deleted by this command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example pushes up routing and cell objects for all soft macros in the current design.

```
prompt> push_up_fp_objects -object_type {routing cells}
```

SEE ALSO

```
push_down_fp_objects(2)
```

pwd

Displays the pathname of the present working directory (pwd), also called the current directory.

SYNTAX

string **pwd**

ARGUMENTS

None

DESCRIPTION

Displays the pathname of the current directory. The directory from which you invoke the Design Analyzer or dc_shell is the initial current directory.

A file specification of dot (.) is shorthand for the current directory. Dot is commonly included in the **search_path** variable.

Use the **cd** command to change the current directory. Note that changing the current directory in the Design Analyzer does *not* affect the current directory of the UNIX shell from which you invoked the Design Analyzer.

EXAMPLES

```
prompt> pwd  
/usr/designer/joe
```

SEE ALSO

cd(2)
search_path(3)

query_objects

Searches for and displays objects in the database.

SYNTAX

```
string query_objects  
[-verbose]  
[-class class_name]  
[-truncate elem_count]  
object_spec
```

Data Types

<i>class_name</i>	string
<i>elem_count</i>	int
<i>object_spec</i>	list

ARGUMENTS

-verbose

Displays the class of each object found. By default, only the name of each object is listed. With this option, each object name is preceded by its class, as in "cell:U1/U3".

-class *class_name*

Establishes the class for a named element in the *object_spec*. Valid classes are design, cell, net, and so on.

-truncate *elem_count*

Truncates display to *elem_count* elements. By default, up to 100 elements display. To see more or less elements, use this option. To see all elements, set *elem_count* to 0.

object_spec

Provides a list of objects to find and display. Each element in the list is either a collection or an object name. Object names are explicitly searched for in the database with class *class_name*.

DESCRIPTION

The **query_objects** command (a simple interface) finds and displays objects in the runtime database of Design Compiler. The command does not have a meaningful return value; it simply displays the objects found and returns the empty string.

The *object_spec* is a list containing collection handles or object names. For elements of the *object_spec* that are collection handles, **query_objects** simply displays the contents of the collection.

For elements of the *object_spec* that are names (or wildcarded patterns), **query_objects** searches for the objects in the class specified by *class_name*. Note that **query_objects** does not have a predefined, implicit order of classes for which searches are initiated. If you do not specify *class_name*, only those elements that are collection handles display. Messages display for the other elements (see

EXAMPLES).

To control the number of elements displayed, use the `-truncate` option. If the display is truncated, you see the elipsis (...) as the last element. Note that if the default truncation occurs, a message displays showing the total number of elements that would have displayed (see EXAMPLES).

NOTE: For those of you familiar with the `dc_shell` **find** command, the **query_objects** command covers part of the functionality of **find**. The output from **query_objects** looks similar to the output of **find**, but remember that the result of **query_objects** is always the empty string.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

These following examples show the basic usage of **query_objects**.

```
prompt> query_objects [get_cells o*]
{"or1", "or2", "or3"}
prompt> query_objects -class cell U*
{"U1", "U2"}
prompt> query_objects -verbose -class cell \
    [list U* [get_nets n1]]
{"cell:U1", "cell:U2", "net:n1"}
```

When you omit the **-class** option, only those elements of the *object_spec* that are collections generate output. The other elements generate error messages.

```
prompt> query_objects [list U* [get_nets n1] n*]
Error: No such collection 'U*' (SEL-001)
Error: No such collection 'n*' (SEL-001)
{"n1"}
```

When the output is truncated, you get the elipsis at the end of the display. For the following example, assume the default truncation is 5 (it is actually 100).

```
prompt> query_objects [get_cells o*] -truncate 2
{"or1", "or2", ...}
prompt> query_objects [get_cells *]
{"or1", "or2", "or3", "U1", "U2", ...}
Output truncated (total objects 126)
```

SEE ALSO

`collections(2)`
`get_cells(2)`
`get_clocks(2)`
`get_designs(2)`

```
get_lib_cells(2)
get_lib_pins(2)
get_libs(2)
get_nets(2)
get_path_groups(2)
get_pins(2)
get_ports(2)
get_timing_paths(2)
```

quit

Exits the shell.

SYNTAX

string **quit**

ARGUMENTS

None.

DESCRIPTION

This command exits from the application. It is basically a synonym for **exit** with no arguments.

EXAMPLES

The following example exits the current session.

```
prompt> quit
```

SEE ALSO

`exit(2)`

read_antenna_violation

Reads an antenna violation file into the tool.

SYNTAX

```
status read_antenna_violation  
[-format hercules]  
[-file file_name]
```

ARGUMENTS

```
-format hercules  
    Specifies the antenna violation file format.  
  
-file file_name  
    Specifies the antenna violation file name.
```

DESCRIPTION

This command is used to read an antenna violation file. The **insert_diode** and **connect_spare_diode** commands use this information.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following is an example of the **read_antenna_violation** command.

```
prompt> read_antenna_violation \  
-format hercules -file antenna.rpt
```

The following is an example of the input file.

```
C 272.799988 128.000000 METAL METAL 77.773973 0.350400 4
```

SEE ALSO

```
insert_diode(2)  
connect_spare_diode(2)
```

read_aocvm

Reads advanced on-chip variation (OCV) derate factor tables.

SYNTAX

```
status read_aocvm  
[-distance_row row_number]  
aocvm_file
```

Data Types

<i>row_number</i>	integer
<i>aocvm_file</i>	string

ARGUMENTS

-distance_row *row_number*
Specifies which row to use from a 2-D advanced OCV derate table containing both depth and distance dependency data. PrimeTime supports both depth and distance data, whereas this tool supports only depth data. This option lets you use a 2-D advanced OCV table intended for PrimeTime and specify which row to use in the table. By default, the command uses the most pessimistic derate values found among the rows of the table.

DESCRIPTION

The **read_aocvm** command reads advanced on-chip variation (OCV) derate factor tables from a disk file. The tables are annotated onto one or more design objects. Allowed design object types are hierarchical cells, library cells, and designs. The advanced OCV data is used to reduce pessimism and improve the accuracy of results by applying different derate factors based on the depth of each path (number of successive gates in the path).

The **read_aocvm** command can read binary and compressed binary files created by PrimeTime's **write_binary_aocvm** command. No additional arguments are required to read binary or compressed binary files.

To use the files read by **read_aocvm** and perform advanced OCV analysis, the **timing_aocvm_enable_analysis** variable must be set to true.

The behavior of advanced on-chip variation analysis in this tool is the same as graph-based advanced OCV analysis in PrimeTime with PrimeTime's **timing_aocvm_analysis_mode** variable set to **clock_network_only**. The tool applies different derate factors to clock paths based on the depth of each clock path. It applies constant derate factors to data paths, irrespective of depth.

EXAMPLES

In the following example, the first command shows an advanced OCV derate file *test.aocvm*, a single table to be annotated on all library cells. The second command

```
read_aocvm  
1346
```


reads in and applies this table.

```
prompt> sh cat test.aocvm

version:          1.0

object_type:      lib_cell
rf_type:          rise
delay_type:       cell
derate_type:      late
object_spec:      my_lib/*
depth:            1 2 3
distance:         100 1000
table:            1.20 1.10 1.08 \
                  1.22 1.15 1.11

prompt> read_aocvm test.aocvm
```

The tool uses the data from the second row in the table because that row contains the more extreme late derate values. To use the data in the first row instead, you would use the **-distance_row 1** option in the **read_aocvm** command.

SEE ALSO

```
remove_aocvm(2)
report_aocvm(2)
report_timing(2)
set_aocvm_coefficient(2)
timing_aocvm_enable_analysis(3)
```

read_ddc

Reads in one or more design files in .ddc (Synopsys logical database) format.

SYNTAX

```
status read_ddc  
file_names  
[-scenarios scenario_list]  
[-active_scenarios active_scenario_list]
```

Data Types

<i>file_names</i>	list
<i>scenario_list</i>	list
<i>active_scenario_list</i>	list

ARGUMENTS

file_names
Specifies the names of one or more files to be read. If specifying more than one file name, enclose the names in braces ({}).

-scenarios *scenario_list*
Specifies a list of scenarios whose constraints should be read from the .ddc file. For complete information on this option see the man page for the **read_file** command.

-active_scenarios *active_scenario_list*
Specifies a list of scenarios that should be made active as the file is read. For complete information on this option see the man page for the **read_file** command.

DESCRIPTION

The **read_ddc** command reads design information from the Synopsys logical database (.ddc) files into **dc_shell**.

This command is derived from **read_file -format ddc**. For more information, see the man page for the **read_file** command.

Multicorner-Multimode Support

This command uses information from both active and inactive scenarios.

SEE ALSO

read_file(2)

read_def

Annotates the design with the data from a file in Design Exchange Format (DEF).

SYNTAX

```
status read_def
[-check_only]
[-enforce_scaling]
[-no_incremental]
[-verbose]
[-turn_via_to_inst]
[-inexactly_matched_via_to_inst]
[-lef lef_file_names]
[-snet_no_shape_as_user_enter]
[-snet_no_shape_as_detail_route]
[-preserve_wire_ends]
def_file_names
```

Data Types

```
lef_file_names      list
def_file_names     list
```

ARGUMENTS

- check_only**
Turn on read_def check_only mode. check_only mode is used to analyze the input DEF file before doing real object annotation, and to give diagnostic information on the correctness and integrity of DEF file. check_only mode does not annotate any DEF information into the Milkyway.
- enforce_scaling**
Instructs the tool to continue to annotating information given in the DEF input regarding of possible roundoff errors due to unit conversion.
- no_incremental**
Specifies that all pre-existing physical annotations will be removed, and the physical information in the DEF file will be read in.
- verbose**
Enables the printing of additional debugging messages of two types: those that contain information and those that contain warnings. The **-quiet** and **-verbose** options are mutually exclusive; you should use only one. If you specify both options, the tool honors **-verbose**.
- turn_via_to_inst**
Specifies that turn vias are created as via cell instances. Turn vias are represented in DEF as a single rectangle (usually a piece of dangling metal). By default, turn vias are created as path objects.
- inexactly_matched_via_to_inst**
Specifies that inexactly matched vias are created as via cell instances.

Inexactly matched vias are vias which match a Milkyway technology contact code's cut dimensions, but the enclosure dimensions are larger than the contact code's enclosure dimensions. By default, inexactly matched vias are created as contact arrays and extra paths on the enclosure layers.

-lef lef_file_names

Specifies to import the rotated vias and design-specific nondefault rules from the LEF files into the design. The incremental LEF files are generated by the **write_def** command.

-snet_no_shape_as_user_enter

Marks the wire segments without "+ SHAPE" statement in special nets with the route type "user enter".

-snet_no_shape_as_detail_route

Marks the wire segments without "+ SHAPE" statement in special nets with the route type "detail route".

-preserve_wire_ends

Indicates that special power/ground wiring should be created as paths without extensions (that is, square ends), special fill wiring should be created as rectangles, and all other special and regular wiring should be created as wires with half-width extensions (that is, square ends by half-width).

def_file_names

Specifies the name of the file from which to draw physical data to annotate onto the design. This file must be in Design Exchange Format (DEF). For multiple files, the def files are read in as incremental mode, and you must specify the def files in the right order. It strongly recommended that you remove overlapping floorplanning data, such as ROWS, and duplicate routing geometry data in the DEF files. These conditions might be error prone in the subsequent flow.

DESCRIPTION

This command annotates the design with physical design data taken from a file in Design Exchange Format (DEF). The command can read either an uncompressed or compressed (in gzip format) DEF file.

The command reads physical data associated with the current design from a DEF file. If cell, port, or net specified in the DEF file do not exist in the database, they will be created as the corresponding object in Milkyway.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

In the following example, the design is annotated with the physical data defined in the file named top.def.

```
prompt> read_def top.def
```

SEE ALSO

```
write_def(2)
```

read_drc_error_file

Creates an error cell from the given drc error file.

SYNTAX

```
status read_drc_error_file  
[-drc_type type]  
[-error_cell cell_name]  
drc_file
```

Data Types

<i>type</i>	string
<i>cell_name</i>	string
<i>drc_file</i>	string

ARGUMENTS

-drc_type *type*
Type of the drc_file. Valid values are: calibre. If this option is not specified, the default type is "calibre".

-error_cell *cell_name*
Name to assign the error cell.
For calibre drc report, if this option is not specified, <cell_name>.err is used by default, where <cell_name> is found inside the drc report file.
If this option is not specified and <cell_name> is not found, this command will abort.

drc_file
The DRC error report to read in.

DESCRIPTION

read_drc_error_file takes a drc error report as input and creates an error cell. The error cell can then be loaded with the error browser.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following is an example of how to run **read_drc_error_file** command.

```
prompt> read_drc_error_file drc_report
```

SEE ALSO

`report_error_coordinates(2)`

read_file

Reads designs or libraries into memory, or reads libraries into the shell.

SYNTAX

```
list read_file  
file_list  
[-format format_name]  
[-ilm]  
[-single_file single_file_name]  
[-scenarios scenario_list]  
[-active_scenarios active_scenario_list]
```

Data Types

<i>file_list</i>	list
<i>format_name</i>	string
<i>single_file_name</i>	string
<i>scenario_list</i>	list
<i>active_scenario_list</i>	list

SYNTAX for lc_shell

ARGUMENTS

file_list
Specifies a list of files to read. When specifying more than one *input_file*, enclose the list of names in braces ({}).

-format *format_name*
Specifies the format in which a design is read. The *format_name* can be one of the following formats:

Format	Description
ddc	Synopsys internal database format (default)
db	Synopsys internal database format
equation	Synopsys equation format
pla	Berkeley (Espresso) PLA format
st	Synopsys State Table format

-ilm
Reads from the Milkyway ILM (interface logic model) view.

-single_file *single_file_name*
Puts the designs in the *file_list* option into the single internal design file specified by *single_file_name*, regardless of the input format.

-scenarios *scenario_list*
Specifies a list of scenarios whose constraints should be read from the .ddc file. Constraints for any scenarios that are not listed are not read, which may save memory and time if the file is large and certain scenarios are not required in the current session. If this option is not used then all scenarios

will be read from the file. This option may only be used with the `.ddc` format.

`-active_scenarios active_scenario_list`

Specifies a list of scenarios that should be made active as the file is read. Any scenarios that are not listed will be restored in the inactive state. This option must be used in conjunction with the **-scenarios** option and may contain only scenarios that are listed in **-scenarios scenario_list**. If the `-active_scenarios` option is not used then the active or inactive state of the scenarios will be restored according to the state that was recorded when the file was written (unless the scenarios already exist). This option may only be used with the `.ddc` format.

DESCRIPTION

This command reads the designs in *file_list* into memory. The current design is automatically set to the first design that is read. The **read_file** command also can read the libraries in *file_list* into the shell.

While it is good practice to always specify the **-format** option, if **-format** is not specified then the tool will attempt to infer the format based on the file name extensions, if any. The recognized extensions are `.ddc` (ddc); `.db`, `.pdb`, `.sdb`, `.sldb` (db); `.v` or `.verilog` (verilog); `.sv` or `.sverilog` (sverilog); `.vhd` or `.vhdl` (vhdl). Extensions are not case-sensitive. If the file name ends in `.gz` then the preceding extension, if any, is used -- e.g., an extension of `.v.gz` is recognized as verilog format. (The `.gz` extension is not supported for ddc files.) If multiple files are specified on the same command line then their formats must all be the same. If the format cannot be inferred from the file names then the default format, ddc, is used.

Internally, designs are stored in design files. An internal design file is a "container" of designs and has a complete, unique, UNIX-type filename associated with it. The designs in a design file also have unique names, but designs with the same name can exist in different design files. For information about displaying design files and their contents, see the **list_designs** man page.

By default, when the tool reads files in a format other than `.ddc` or `.db`, it places each design in its own internal design file named in the *input_file_path/design_name*. **ddc** format. The names of files read in `.db` or `.ddc` format remain the same.

When the tool reads an HDL parameterized design file, the design is not converted to a `.ddc` file, but instead is parsed and stored in an intermediate format. You then can build the design by using the **elaborate** command or by instantiating the design in another HDL design.

Because internal database files at lower levels of hierarchy are automatically read when they are linked, you do not need to read them separately. Designs are not automatically linked at the time they are read. You can use the **link** command after the **read_file** command to ensure that referenced designs are read correctly. For information on automatic loading, see the **link** man page.

Internally, libraries are stored in library files. An internal library file is a "container" of libraries and has a complete, unique, UNIX-type filename associated with it. The libraries in a library library file also have unique names, but libraries with the same name can exist in different library files. For information

about displaying library files and their contents, see the **list_libs** man page.

By default, when the tool reads non-database format files, each library is placed in its own internal library file named in the *input_file_path/file_name.db*:library_name format. The names of files read in .db format remain the same.

An input file can have either a simple or a complex filename. A simple filename has no directory specification, which means (in UNIX) that it does not contain a / (slash). Simple filenames such as adder.ddc or library.db must be found in a directory listed in the **search_path** variable. A complex filename has a directory specification in it, which means that it does contain one or more slashes. Complex filenames such as ./test.ddc, ../test.ddc, or ~john/test/test.ddc are not searched for in the **search_path**.

Multicorner-Multimode Support

This command supports an option to specify which active scenarios it should work with.

EXAMPLES

The following example uses simple and complex filenames to read in a file of equations. This is a complex path specification, so **search_path** is not used.

```
prompt> read_file -format equation ~bill/dc/test/test.fnc
Loading file '/usr/bill/dc/test/test.fnc'
test
```

The following example uses a simple file name to specify the same file as shown in the example above. Thus, its directory must appear in the **search_path**.

```
prompt> search_path = {~bill/dc/test}
~bill/dc/test
prompt> read_file -format equation test.fnc
Loading file '/usr/bill/dc/test/test.fnc'
test
```

The following example uses simple and complex filenames to read the library. This is a complex path specification, so the **search_path** is not used.

```
prompt> read_file ~bill/dc/test/test.db
Loading db file '/usr/bill/dc/test/test.db'
{test}
```

The following example shows that in order to specify the same file as shown in the example above as a simple filename, its directory must appear in the **search_path**.

```
prompt> search_path = {~bill/lc/test}
{~bill/lc/test}
```

```
prompt> read_file test.db  
Loading db file '/usr/bill/lc/test/test.db'  
{test}
```

SEE ALSO

```
all_scenarios(2)  
all_active_scenarios(2)  
change_names(2)  
create_scenario(2)  
link(2)  
list_designs(2)  
list_files(2)  
list_libs(2)  
set_active_scenarios(2)  
which(2)  
write(2)  
search_path(3)  
view_read_file_suffix(3)
```

read_flip_chip_bumps

Reads bump locations and connected nets from a file with the advanced input format (AIF).

SYNTAX

```
status read_flip_chip_bumps
-physical_lib_cell {phy_lib_cels}
[-orientation N | W | S | E | FN | FE | FS | FW]
file_name
```

Data Types

<i>phy_lib_cels</i>	collection
<i>file_name</i>	string

ARGUMENTS

```
-physical_lib_cell {phy_lib_cels}
    Specify a collection of physical library cells. The collection must contain
    one cell whose location serves as the reference point from which the flip-
    chip bump cells are placed in the die area.

-orientation N | W | S | E | FN | FE | FS | FW
    Specifies the orientation of the bump cells. The default is N.

file_name
    Specifies the name of the input file. The format of the file must be AIF.
```

DESCRIPTION

This command reads bump locations and net connections from a text file in the AIF format. The file can be specified with a relative or absolute path. The tool reads the bump locations and connected nets from the [NETLIST] section of the file even though the file might contain other unrelated data. The coordinates of the bump cells are referencing to the center of the die. The die size in the file is taken into account when calculating its coordinates referencing the lower-left corner. So $ICC\_Pad\_x = AIF\_Pad\_x + die\_width/2$, $ICC\_Pad\_y = AIF\_pad\_y + ide\_hight/2$.

The following is an example of AIF file.

```
;This is a comment
;File stars from here
```

```
[DATABASE]
TYPE=AIF
VERSION=2.0
UNITS=um
```

```
read_flip_chip_bumps
1358
```

[DIE]
WIDTH=8785
HEIGHT=8785
NAME=DIE1G

[PADS]
PAD1=SQ 100 10
PAD2=RECT 200 100 10

[NETLIST]
;Netname Pad# Type Pad_X Pad_Y Ball#
GND BUMP_1-PAD BUMPCELL 3795.0 -3795.0
GND BUMP_2-PAD BUMPCELL 3565.0 -3795.0
GND BUMP_3-PAD BUMPCELL 3335.0 -3795.0
GND BUMP_4-PAD BUMPCELL 3105.0 -3795.0
GND BUMP_5-PAD BUMPCELL 2875.0 -3795.0
GND BUMP_6-PAD BUMPCELL 2645.0 -3795.0
GND BUMP_7-PAD BUMPCELL 2415.0 -3795.0
GND BUMP_8-PAD BUMPCELL 2185.0 -3795.0
GND BUMP_9-PAD BUMPCELL 1955.0 -3795.0
GND BUMP_10-PAD BUMPCELL 1725.0 -3795.0
GND BUMP_11-PAD BUMPCELL 1495.0 -3795.0
GND BUMP_12-PAD BUMPCELL 1265.0 -3795.0
GND BUMP_13-PAD BUMPCELL 1035.0 -3795.0
GND BUMP_14-PAD BUMPCELL 805.0 -3795.0
GND BUMP_15-PAD BUMPCELL 575.0 -3795.0
GND BUMP_16-PAD BUMPCELL 345.0 -3795.0
GND BUMP_17-PAD BUMPCELL 115.0 -3795.0
GND BUMP_18-PAD BUMPCELL -115.0 -3795.0
GND BUMP_19-PAD BUMPCELL -345.0 -3795.0
GND BUMP_20-PAD BUMPCELL -575.0 -3795.0
GND BUMP_21-PAD BUMPCELL -805.0 -3795.0

Descriptions:

[DATABASE]
Identifies the file as an AIF file and contains the version and units of the coordinates.

[DIE]
Describes the width and height of the die and a name for the die. Width is measured along the horizontal X axis and height is measured along the vertical Y axis. The width and height are defined per die nominal described dimensions.

[PADS]
The PADS section defines all pad types needed for the die.

[NETLIST]
Defines each net and the coordinates of each die pad. Each column must have an entry. If a particular data item is not present, you must use a hyphen as a placeholder.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reads bump locations and net connections from file bump_loc.

```
prompt> read_flip_chip_bumps -  
physical_lib_cel [get_physical_lib_cells "BUMP"] ../scripts/bump_loc
```

SEE ALSO

```
write_flip_chip_bumps(2)  
place_flip_chip_array(2)  
expand_flip_chip_cell_locations(2)
```

read_floorplan

Reads in a script that describes a floorplan into current Milkyway cel.

SYNTAX

```
int read_floorplan  
file_name [-echo]
```

Data Types

file_name string

ARGUMENTS

file_name
 Specifies the name of the file that contains the floorplan script to be read.

-echo
 Indicates that each command is to be echoed as it is executed.

DESCRIPTION

The **read_floorplan** command reads in a script file, which contains commands describing floorplan information.

Like **source**, the **read_floorplan** command is sensitive to variables that control script execution (for example, **sh_continue_on_error** and **sh_script_stop_severity**). The **read_floorplan** command uses **sh_source_uses_search_path** to determine if **search_path** is used to find the script.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reads the floorplan script top.fp.

```
prompt> read_floorplan top.fp
```

```
Reading floorplan ...
```

```
1
```

SEE ALSO

`source(2)`
`write_floorplan(2)`
`search_path(3)`
`sh_continue_on_error(3)`
`sh_script_stop_severity(3)`
`sh_source_uses_search_path(3)`

read_io_constraints

Reads I/O constraints into the specified design.

SYNTAX

```
status read_io_constraints  
[-append]  
[-cell name]  
io_constraints_file
```

Data Types

<i>name</i>	string
<i>io_constraints_file</i>	string

ARGUMENTS

-append

Appends the I/O constraints to the current Milkyway cell. By default, the command overwrites the existing I/O constraints.

-cell *name*

Specifies the name of the Milkyway cell or plan group into which the I/O constraints are read. If you do not specify this option, the constraints are read into the current Milkyway cell.

io_constraints_file

Specifies the name of the file that contains the Tcl commands of the **set_pin_physical_constraints**, **set_pad_physical_constraints** and **set_choplevel_pad_physical_constraints** commands.

In this file, the **set_pin_physical_constraints** and

set_pad_physical_constraints commands must be in the following formats:

set_pin_physical_constraints -pin_name ... **set_pad_physical_constraints** -pad_name ...

Neither **-cell name** nor **object** should appear in these two commands.

DESCRIPTION

Reads the I/O constraints defined in the *io_constraints_file* into the current Milkyway cell.

I/O constraints are used for pad and terminal placement during the floorplan stage.

By default, a Milkyway cell is required to be open before reading the I/O constraints. If no Milkyway cell is open, you can use the **open_mw_cel** command to open one. The **current_mw_cel** command can be used to set an open Milkyway cell as the current Milkyway cell.

If you want to append any constraints, you must use the **-append** option. For example, if there are physical constraints for pin A in the current cell, but there is no pin A constraints entry in *io_constraints_file*, the physical constraints of pin A will

be erased if the `read_io_constraints io_constraints_file` is used.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

In the following example, an I/O constraint file is read into the current Milkyway cell and the existing I/O constraint is overwritten.

```
prompt> read_io_constraints top.tcl  
1
```

To load I/O constraints in incremental mode, the `-append` option is specified.

```
prompt> read_io_constraints -append top_1.tcl  
1
```

SEE ALSO

```
open_mw_lib(2)  
open_mw_cel(2)  
current_mw_cel(2)  
initialize_floorplan(2)  
set_pin_physical_constraints(2)  
set_pad_physical_constraints(2)  
set_chiplevel_pad_physical_constraints(2)
```

read_lib

Reads a technology library, physical library, or symbol library into the shell.
Creates a physical library for GDSII.

SYNTAX

```
int read_lib
[-format format_name]
[-symbol intermediate_symbol_library_file]
[-plibrary physical_library_output_file]
[-pplibrary pseudo_physical_file_name]
file_name
[-no_warnings]
[-names_file file_list]
[-test_model CTL_file_list]
[-html]
[-lib_messages {predefined_lib_message_variables}]
```

Data Types

<i>format_name</i>	string
<i>intermediate_symbol_library_file</i>	string
<i>physical_library_output_file</i>	string
<i>pseudo_physical_file_name</i>	string
<i>file_name</i>	string
<i>file_list</i>	list
<i>CTL_file_list</i>	list
<i>predefined_lib_message_variables</i>	list

ARGUMENTS

- format** *format_name*
Specifies the name of the external format of the specified symbol library. Use this option if you want to specify EDIF/GDSII format; the default is Synopsys Library Compiler format. If the format is GDSII, the **read_lib** command creates a physical library file.
- symbol** *intermediate_symbol_library_file*
Specifies the name of a symbol library file to be created in Synopsys library format. This option is used with EDIF symbol library description files and in conjunction with the **-format EDIF** option.
- plibrary** *physical_library_output_file*
Specifies the name of the intermediate pseudo-physical library file, which is generated from GDSII. This file is also read into the tool for generating a memory database. It is used with the GDSII physical library file and in conjunction with the **-format GDSII** option of GDSII.
- pplibrary** *pseudo_physical_file_name*
Specifies the name of the technology file, as given out by the vendor. This file should be in the pseudo-library format for the tool to read technology-specific information. It is used with the GDSII physical library file and in

conjunction with the **-format GDSII** option of GDSII.

file_name

Specifies the name of the library file to be read. A technology or physical library must be in Synopsys Library Compiler format. A symbol library might be in either Synopsys Library Compiler format or EDIF format. If it is in GDSII format, the file name should be *gdsii* file.

-no_warnings

Specifies that all messages of severity 'warning' are to be suppressed. By default, all messages are issued. Warning messages can identify serious library problems; use this flag sparingly.

-names_file file_list

Specifies a set of one or more name-mapping files when this option is used in conjunction with the **-format edif** option. Design Compiler's EDIF reader uses this set of files to change the names of symbols or symbol pins in the incoming EDIF files (*file_name*) to resolve name-based consistency issues. If these files are required, you must create them before executing the **read_lib** command. When specifying more than one name-mapping file, enclose them in braces ({}).

-test_model CTL_file_list

Specifies a set of one or more CTL files as test model of cells in the library. The format of the *CTL_file_list* is <cell_name:>file_name where cell_name is optional and is to specify the particular cell a CTL file models. When the cell_name is missing, the environment name in the CTL file is used as the cell name it is modelling. When specifying more than one CTL file, enclose them in braces ({}).

-html

Specifies that library screener results are reported in html.

-lib_messages {predefined_lib_message_variables}

Specifies one or more predefined or user defined tcl variables that contain a list of message IDs to be listed on the front page of the html report. The predefined variables include \$read_lib_ccs_noise_msg.

DESCRIPTION

Reads a library file into the shell (or GUI). The library file is automatically compiled by the Synopsys Library Compiler.

This command is also used to extract the geometries from GDSII files.

Technology specific information to extract GDSII files should be in Synopsys pseudo-library format.

The **names_file** option enables you to change the names of symbol and symbol pins. The general format of the name-mapping files is the same for the **read_lib** command as it is for the **change_names** command. The object types for the **read_lib** command are **reference or component** (to change symbol names), and **pin** (to change symbol pin names).

The library specified in *file_name* can have either a simple or complex filename. A simple filename has no directory specification, which means (in UNIX) that it does not contain a "/" (slash). Simple filenames such as `test.lib` or `library.db` must be found in a directory listed in the **search_path** variable. A complex filename has a directory specification in it, which means that it does contain a slash. Complex filenames such as `./test.lib` or `~synopsys/dc/test.lib` are not searched for in **search_path**.

You can read any number of libraries into the shell. Two libraries can have the same name in the system as long as they are read from different sources. The command **list_libs** shows all the libraries present in the system.

For details on library file syntax, see *Library Compiler Reference Manual*. Function statements and state information are ignored when reading technology library files if you do not have a Library Compiler license. You do not need a license to read symbol library files.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

In the following example, the library file `"tech_lib.lib"`, which contains technology information in Synopsys technology library entry format, is read. The **-no_warnings** flag causes all Library Compiler warning messages to be suppressed.

```
prompt> read_lib ~synopsys/libraries/tech_lib.lib -no_warnings
```

The following example reads a symbol library in EDIF format. It also specifies that a library file named `sym_lib.sdb` be created in Synopsys symbol library entry format.

```
prompt> read_lib sym_lib.edif -format EDIF -symbol sym_lib.slib
```

The following example shows a name-mapping file called `lib.names` and its usage:

Design	Type	Old Name	New Name
lib	reference	NR2	nr2
NR2	pin A	a	
NR2	pin B	b	
NR2	pin Z	z	
lib	reference	HO22	aoi2
HO22	pin I1	a	
HO22	pin I2	b	
HO22	pin I3	c	
HO22	pin O1	z	

```
prompt> read_lib foo.edif -format EDIF -names_file lib.names \
-symbol foo.slib
```

The following example reads a list of GDSII files names `foo1` and `foo2`, and a technology file named `bar.plib` and generates an intermediate pseudo-plib file named `foo.plib`. The `foo.plib`, which contains technology information in Synopsys technology

library entry format, is read.

```
prompt> read_lib -format gdsii -plib library foo -pplibrary bar.plib \  
{foo1.gdsii foo2.gdsii}
```

The following example reads a library file as well as a CTL file when `-test_model` option is used.

```
prompt> read_lib bar.lib -test_model {foo1.ctl foo2:foo2.ctl}
```

The following example reads a library file and report screener messages in html.

```
prompt> read_lib bar.lib -html
```

SEE ALSO

- `change_names(2)`
- `compare_lib(2)`
- `list_libs(2)`
- `write_lib(2)`
- `search_path(3)`

read_mw_eco_list

Reads an ECO change file and performs ECO operations on Milkyway designs.

SYNTAX

```
status read_mw_eco_list
[designcell_name]
[-library lib_name]
[-change_file change_filename]
[-explicit_uniq]
[-loose_name_match]
[-softmacro]
```

Data Types

<i>designcell_name</i>	string
<i>lib_name</i>	string
<i>change_filename</i>	string

ARGUMENTS

designcell_name

The name of Milkyway design cell that will be changed. If not specified, the current design will be used. Make sure the design cell should be opened first before running this command.

NOTE: The hierarchy preservation data and the flat design data of the cell need to be consistent.

-library *lib_name*

The name of Milkyway Library in which the design cell stores. If not specified, the current edited library will be used, or TCL variable *mw_design_library* will be used.

-change_file *change_filename*

The file name of Milkyway ECO change file. The file shall be hierarchy ECO change file for hierarchy ECO. **NOTE:** Syntax error in the change file will make Milkyway ECO fail.

-explicit_uniq

ECO will automatically uniquify the parent cell instances by default. If the flag is set, all of the parent cell instances will be required to be uniquified before making ECO change into the design.

-loose_name_match

ECO will match hierarchy nets with exact same name as change files by default. If this flag is set, hierarchical net names will be matched with loose rules. ECO will match hierarchy nets with exact same name at first, and then try the match again with escaped or unescaped name. **NOTE:** This flag is only applicable to nets.

-softmacro

ECO will make changes only to logical hierarchy design by default. All logical

hierarchy data exist in single Milkyway cell. If this flag is set, ECO will make changes in physical hierarchy designs, which include multiple Milkyway cells and are stored as softmacro instances in top level Milkyway design.

DESCRIPTION

This command updates both flat and hierarchy information of the design cell according to the hierarchy ECO change file. The hierarchy information data will keep consistency with the flat cell.

Hierarchy ECO (HECO) Syntax

read_mw_eco_list will read the hierarchy ECO change file and modify both hierarchical and flat data step by step. The operations supported by **read_mw_eco_list** include:

Uniquification

Uniquify hierarchical cell instance
+HU [hierPath/]hierCIName newHierCIMName

Port Operations

Add new port / hierarchical port instance
+HP [hierPath/]hierCIName hierPoIMName direction

Remove port / hierarchical port instance
-HP [hierPath/]hierCIName hierPoIMName

Rename port / hierarchical port instance
*HP [hierPath/]hierCIName hierPoIMName newHierPoIMName

Net Operations

Add new flat / hierarchical net
+HN hierPath/NetName

Remove flat / hierarchical net
-HN hierPath/NetName

Rename flat / hierarchical net
*HN hierPath/NetName newNetName

Connection Operations

Connect flat / hierarchical net with port / hierarchical port instance
+HC hierPath/NetName [childCIName/]PoIMName
Disconnect port / hierarchical port instance from flat / hierarchical net
-HC hierPath/NetName [childCIName/]PoIMName

Instance Operations

Add new flat / hierarchical cell instance
+HI [hierPath/]CIName CIMasterName [viewName]

Remove flat / hierarchical cell instance
-HI [hierPath/]CIName [CIMName] [CITYPE]


```
Rename flat / hierarchical cell instance
*HI [hierPath/]CIName newCIName

Resize flat cell instance
+HR [hierPath/]CIName newCIMName [oldCIMName]
Note: only flat cell instance resizing is supported

Move flat / hierarchical cell instance into another
      hierarchical branch
*HT [hierPath/]CIName [newHierPath/]newCIName
Note: all connections to the cell instance will be cut down
```

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

Following are examples of changes made on Milkyway design cells.

The following example is to make changes from the hierarchy ECO to a change file

```
prompt> read_mw_eco_list \  
-change_file eco_changefile.txt orig_design
```

SEE ALSO

read_parasitics

Reads parasitics information from a disk file for the delay calculation tools.

SYNTAX

```
int read_parasitics  
[-format SPEF | SBPF]  
[-syntax_only]  
[-max_file max_file_name  
[-min_file min_file_name]]  
[-keep_capacitive_coupling]  
[-triplet_type min | typ | max]  
  | | [-ilm_context]  
[file_list]
```

Data Types

<i>max_file_name</i>	string
<i>min_file_name</i>	string
<i>file_list</i>	list

ARGUMENTS

-format SPEF | SBPF
Specifies that parasitics files are to be read in either Standard Parasitic Exchange Format (SPEF) or Synopsys Binary Parasitic Format (SBPF).

-syntax_only
Specifies only to parse the files.

-max_file *max_file_name*
Specifies to read the file that contains the maximum parasitic data for the current design. This option is can be used alone or in conjunction with the **-min_file** option. The **-max_file** and *file_list* options are mutually exclusive; you can use only one.

-min_file *min_file_name*
Specifies to read the file that contains the minimum parasitic data for the current design. If you use this option, you must also use the **-max_file** option. The **-min_file** and *file_list* options are mutually exclusive; you can use only one.

-keep_capacitive_coupling
Specifies to retain coupling capacitances.

-triplet_type min | typ | max
Specifies the type of RC values stored in the SPEF file. This option is only valid for SPEF files.

-ilm_context
Reads parasitics with the interface logic model (ILM).

`file_list`

Specifies the files to read. If not specified, the tool reads from the `design_name_parasitics.reduced` file, where you substitute the name of the current design for `design_name`. The `file_list` option is mutually exclusive with the **-max_file** and **-min_file** options.

DESCRIPTION

This command reads parasitics for the current design from a disk file.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example reads parasitics for all nets in the current design. The file named `top.sbpf` is in reduced Synopsys Binary Parasitic Format (SBPF).

```
prompt> read_parasitics -format sbpf top.sbpf
```

SEE ALSO

`remove_annotated_delay(2)`
`report_annotated_delay(2)`
`reset_design(2)`
`write_parasitics(2)`

read_partition

Loads a Verilog netlist into a partition editor

SYNTAX

```
status read_partition
[-input_files files]
[-verbose]
```

Data Types

```
files          string list
```

ARGUMENTS

```
-input_files files
    Specifies that a partition editor is to be created by using one or more
    Verilog files. The structure of the Verilog files does not matter. Each module
    must be defined only once. All unused modules will remain as independent top
    modules.

-verbose
    The read_partition command outputs more detailed messages.
```

DESCRIPTION

This command allows you to load a Verilog netlist into a partition editor. After loading the netlist, you can use the **set_partition_data** and **create_partition** commands to determine a partitioning for the design and to create a Milkyway database for the design.

EXAMPLES

This example creates a partition editor.

```
prompt> read_partition -input_files {myDesign.v}
Partition Name: pfe0
Loading verilog file myDesign.v
MYDESIGN: 73958 instances of 134 modules.
1
```

SEE ALSO

```
create_partition(2)
set_partition_data(2)
list_partition_data(2)
remove_partition(2)
```

read_rail_maps

Reads rail map data from a map file.

SYNTAX

```
status read_rail_maps  
[-file file_name]  
[-nets net_names]
```

Data Types

```
file_name      string  
net_names     list
```

ARGUMENTS

-file *file_name*
Reads the specified map data file into IC Compiler. This map data file must have been generated by a compatible version of PrimeRail.

-nets *net_names*
Reads the map data for the specified nets into IC Compiler. A design should be open when using this option. The corresponding map files are automatically located and their names are determined by using the rail options defined by the **set_rail_options** command. The map data files must have been generated by a compatible version of PrimeRail.

DESCRIPTION

This command reads map data files generated by PrimeRail and loads them into IC Compiler. Reading map data files into IC Compiler enables the display of PrimeRail maps generated by the **analyze_rail** command. You do not need to have a layout window open at the time this command is executed, but viewing the loaded maps requires an open layout window. A map data file can contain data for one or many maps, depending upon the specific analyses performed using the **analyze_rail** command or the map file exported from the **pgExportICCDData** command in the PrimeRail tool. For more information, see the *PrimeRail User Guide*.

EXAMPLES

The following example shows how to read a map file.

```
prompt> read_rail_maps -file map_file
```

The following example shows how to read net map data.

```
prompt> read_rail_maps -nets {net1 net2}
```

SEE ALSO

`set_rail_options(2)`
`analyze_rail(2)`
`remove_rail_maps(2)`

read_saif

Reads a SAIF file and annotates switching activity information on nets, pins, ports, and cells in the current design.

SYNTAX

```
status read_saif
-input file_name
-instance_name name
[-target_instance instance]
[-ignore ignore_name]
[-ignore_absolute ig_absolute_name]
[-exclude exclude_file_name]
[-exclude_absolute ex_absolute_file_name]
[-names_file name_changes_log_file]
[-scale scale_value]
[-unit_base unit_value]
[-khrate khrate_value]
[-map_names]
[-auto_map_names]
[-rtl_direct]
[-verbose]
```

Data Types

<i>file_name</i>	string
<i>name</i>	string
<i>ignore_name</i>	string
<i>ig_absolute_name</i>	string
<i>exclude_file_name</i>	string
<i>ex_absolute_file_name</i>	string
<i>name_changes_log_file</i>	string
<i>scale_value</i>	integer
<i>unit_value</i>	string
<i>khrate_value</i>	float

ARGUMENTS

-input *file_name*
Specifies the name of the SAIF file to be read.

-instance_name *name*
Specifies the name of the instance of the current design as it appears in the SAIF file. The **read_saif** command assumes that the current design is instantiated as an instance in the testbench used to generate the SAIF file. The current design appears as an instance in the SAIF file. The **read_saif** command annotates all sub-instances in the hierarchy of the specified instance, and annotates the instance itself. Please enter each instance name completely, without any trailing hierarchy separator (/).

-target_instance *instance*
Specifies the target instance on which the switching activity in the SAIF

file is to be annotated. If not specified, the target instance is assumed to be the current instance.

-ignore *ignore_name*

Specifies the name of an instance in the SAIF file for which switching activity is to be ignored. The **read_saif** command ignores switching activity within the hierarchy of that instance in the SAIF file. The *ignore_name* option can be a hierarchical name separated by a slash (/). The **-instance_name** is applied first. The **read_saif** command then strips off the prefix of a net name that matches *name* before deciding whether or not this net name is under the hierarchy specified by the **-ignore** option.

-ignore_absolute *ig_absolute_name*

Specifies the name of an instance in SAIF file for which switching it ignored, but this option is not affected by the **-instance_name** argument. The **read_saif** command determines whether a net name is under the **-ignore** hierarchy before applying the **-instance_name** argument.

-exclude *exclude_file_name*

Specifies the name of a file that contains a list of names to be ignored. The *exclude_file_name* option is used when you want to ignore switching activity for several instances. The file must contain each *ignore_name* on a separate line, without the **-exclude** option. The ignored names are recognized after the **-instance_name** argument.

-exclude_absolute *ex_absolute_file_name*

Specifies the name of a file that contains a list of absolute names to be ignored. The absolute names are not affected by the **-instance_name** argument.

-names_file *name_changes_log_file*

Specifies a previously-created log file, which is usually the output of a **change_names -log** command. The log file is used as the input to **read_saif**. Whereas the design file can contain multiple name changes, the SAIF file contains only the original object names. So the object search by name might fail to find objects with names that were changed. The *name_changes_log_file* contains a list of **change_names** commands for cells, ports, and nets in the current design. The file provides mapping information so that **read_saif** can identify the correct objects, even if the names have changed. Note that *name_changes_log_file* is a text file, and it can be edited manually to include other name changes that are not captured by the **change_names** command.

-scale *scale_value*

Specifies the value of the scaling factor for the base synthesis time unit. Allowed values for *scale_value* are **1** (the default), **10**, or **100**. The synthesis time unit is usually obtained automatically from the target library. In some cases the target library is unknown and **read_saif** issues a warning that the intended synthesis time unit is assumed to be 1 ns (nanosecond). For example, this can occur when **read_saif** is issued before the **compile** command. If this warning message appears and the time unit of the intended target library is not 1 ns, then use the **-scale** and **-unit_base** options to specify the time unit of the intended target library. For instance, if the intended time units are 100 ps, the required options are **-scale 100 -unit_base ps**.

-unit_base *unit_value*

Specifies the base synthesis time unit. Allowed values for *unit_value* are **ns**

(the default), **us**, or **ps**. See the **-scale** argument above for information about using it with the **-unit_base** option.

-khrate *khrate_value*

Specifies a default derating factor value to use for inertial glitches in the SAIF file. If **-khrate** is not specified, the tool uses the default derating factor of 0.5.

-map_names

Specifies that the SAIF name mapping mechanism is used to match the objects in the SAIF file with the design objects.

-auto_map_names

Specifies that a name mapping will be created automatically using SAIF file, and the created name mapping will be used to read the SAIF file. The **-auto_map_names** flag basically has the effect of creating the name mapping information using **saif_map -create_map** and reading in the SAIF file information using **read_saif -map_names**.

-rtl_direct

Indicates that the SAIF file was generated from RTL simulation without reading in an RTL forward SAIF file. When using this option, make sure that the SAIF file obtained from simulation was done on an RTL design, and that no RTL forward SAIF files were used.

Examples of RTL simulation flows that do not use RTL forward SAIF files include:

- When using the vpower PLI/FLI, the **read_rtl_saif** command was not performed, and the net monitoring was set to *rtl_on* using the **set_gate_level_monitoring** command in the PLI, and the **set_net_monitoring_policy** command in FLI.

- When using the vcd2saif utility, the **-rtl** argument was not used.

For more information about using the the vpower PLI for Verilog, the power FLI interface for the MTI VHDL simulator, and the vcd2saif application, see the Power Compiler manuals. manuals and the appropriate usage files.

Please note that flows involving forward RTL SAIF files, and flows involving the **read_saif -rtl_direct** will be made obsolete in future releases. Use flows involving the **saif_map** and **read_saif -map_names** and **-auto_map_names** instead.

-verbose

Specifies to show information in verbose message mode. Verbose messages include showing the warnings on design objects in the SAIF file that cannot be annotated on the design.

DESCRIPTION

The **read_saif** command reads a SAIF file, and annotates the switching activity attributes **toggle_rate** and **static_probability** for nets, ports, and pins of the current design. The **report_power** command uses this information to calculate dynamic power values. If a particular object in the SAIF file is not be found in the current design, the tool ignores the object and issues a warning message. The **read_saif** command returns 1 if at least one of the objects in the file is successfully annotated. Otherwise, it returns 0.

To identify the instance (and corresponding subinstances) you want to annotate, use the **-instance_name** option. To set the synthesis timing unit, use **-unit_base** and **-scale**, unless you are certain that the synthesis timing unit is 1 ns. To specify a default derating factor other than 0.5, use the **-khrate** option.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example annotates the switching activity on the current design, which has been instantiated as Test/U1 in the simulation testbench:

```
prompt> read_saif -input file -instance_name Test/U1
```

The following commands annotate the current design, which has been instantiated as Test/U1 in the simulation testbench, assuming that the man library timing unit is 10 ns:

```
prompt> read_saif -input file -instance Test/U1 -scale 10 \
        -unit ns
```

```
prompt> read_saif -input file -instance Test/U1 -scale 10
```

The commands below demonstrate annotation of multiple designs. The *foo1* design is instantiated as Test1/U1, and the *foo2* design is instantiated as Test1/U2.

```
prompt> current_design foo1
```

```
prompt> read_saif -input file_1 -instance Test1/U1 -scale 10
```

```
prompt> current_design foo2
```

```
prompt> read_saif -input file_2 -instance Test1/U2 -scale 10
```

The following examples illustrate the use of the **-names_file** option. In the first three commands, the *name_changes.log* file is created. In the fourth command, the *name_changes.log* file is used as input to the **read_saif** command. The name changes recorded in *name_changes.log* are used in name matching during execution of **read_saif**.

```
prompt> read_ddc design.ddc
prompt> define_name_rules my_rules -restricted "A-Z"
prompt> change_names -rule my_rules -log name_changes.log
prompt> read_saif -input file -instance Test/U1 \
        -names name_changes.log
```

SEE ALSO

report_saif(2)
report_power(2)
set_switching_activity(2)

read_sdc

Reads in a script in Synopsys Design Constraints (SDC) format.

SYNTAX

```
status read_sdc  
file_name  
[-echo]  
[-syntax_only]  
[-version sdc_version]
```

Data Types

<i>file_name</i>	string
<i>sdc_version</i>	string

ARGUMENTS

file_name
Specifies the name of the file that contains the SDC script to be read.

-echo
Indicates that each constraint is to be echoed as it is executed.

-syntax_only
Indicates that SDC script is processed only to check the syntax and semantics of the script.

-version *sdc_version*
Specifies the version of SDC to which the file conforms. Allowed values are 1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, and latest (the default).

DESCRIPTION

This command reads in a script file in Synopsys Design Constraints (SDC) format, which contains commands for use by PrimeTime or by Design Compiler in its Tcl mode. SDC is also licensed by external vendors through the Tap-in program. SDC formatted script files are Tcl scripts that use a subset of the commands supported by PrimeTime and Design Compiler.

By default, **read_sdc** executes the constraints as they are read. If there are errors part way through the script, it is possible that some constraints are applied, and some are not. You can use the **-syntax_only** option to simply check the script without applying any constraints. This also verifies conformance to the specified SDC version. If there are any errors during the checking phase, then the result of **read_sdc** is 0.

Like **source**, the **read_sdc** command is sensitive to variables that control script execution (for example, **sh_continue_on_error** and **sh_script_stop_severity**). The **read_sdc** command uses **sh_source_uses_search_path** to determine if **search_path** is used to find the script.

read_sdc supports several file formats. The file can be a simple ascii script file, an ascii script file compressed with gzip, or a Tcl bytecode script, created by TclPro Compiler.

Note that SDC does not allow command abbreviation. Also note that **exit** and **quit** do not cause the application to exit, but they do cause the reading of the SDC file to stop.

The latest SDC Version supports the following commands. Those added for versions 1.3 and later are noted. Some constraints which PrimeTime ignores are not listed.

General Purpose Commands

- list
- expr
- set

Object Access Functions

- all_clocks
- all_inputs
- all_outputs
- current_design
- current_instance
- get_cells
- get_clocks
- get_libs
- get_lib_cells
- get_lib_pins
- get_nets
- get_pins
- get_ports
- set_hierarchy_separator

Basic Timing Assertions

- create_clock
- create_generated_clock (1.3)
- set_clock_gating_check
- set_clock_latency
- set_clock_transition
- set_clock_uncertainty
- set_data_check (1.4)
- set_false_path
- set_input_delay
- set_max_delay
- set_min_delay
- set_multicycle_path
- set_output_delay
- set_propagated_clock

Secondary Assertions

- set_disable_timing
- set_max_time_borrow
- set_timing_derate (1.5)

Environment Assertions

- set_case_analysis
- set_drive
- set_driving_cell
- set_fanout_load
- set_input_transition
- set_load
- set_logic_zero
- set_logic_one
- set_logic_dc
- set_max_area
- set_max_capacitance
- set_max_fanout
- set_max_transition
- set_min_capacitance
- set_min_fanout
- set_operating_conditions
- set_port_fanout_number
- set_resistance
- set_wire_load_min_block_size
- set_wire_load_mode
- set_wire_load_model
- set_wire_load_selection_group

For a complete guide to using SDC with Synopsys applications, see the *Using the Synopsys Design Constraints Format Application Note* which is available through SolvNET at <http://solvnet.synopsys.com>.

The usage of some of these commands is restricted when reading SDC. In some cases, some command options are not allowed. In some applications, some of the commands are not supported. For example, PrimeTime does not support the **set_logic*** commands. These commands are ignored when loaded in with **read_sdc**, with a message (for an example, see the EXAMPLES section). If a command not supported by SDC is found by **read_sdc**, it appears as an unknown command, with a message. The same condition exists for command options. If an option is not supported by SDC, a message is issued indicating that condition. However, if the option is unknown, a different message is issued.

Note that although **create_generated_clock** is supported (beginning with version 1.3), there is no provision for the **get_generated_clocks** shortcut which is available in PrimeTime and Design Compiler. Generated clocks are simply clocks with some additional attributes and in SDC, they are retrieved through the **get_clocks** command.

The following features are added for SDC Version 1.5 or later:

set_clock_latency -clock option, **set_timing_derate** command with all options, **set_max_transition** -clock_path -data_path -rise -fall options, **set_clock_uncertainty** -rise/fall_from/to options, **set_operating_conditions** -object_list option, synonyms for all **get_object** commands, **get_objects** -nocase support independently with **-regexp**, **get_objects** -regexp support, **get_objects** -of_objects support for **get_cells/get_nets/get_pins**.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example reads the SDC script top.sdc.

```
prompt> read_sdc top.sdc -version 1.2
```

```
Reading SDC version 1.2...
1
```

The following example reads the SDC script top2.sdc, The script contains commands not supported by SDC, and CMD-005 messages are generated for those commands.

```
prompt> read_sdc -syntax_only top2.pt -version 1.2
Checking syntax/semantics using SDC version 1.2...
Error: Unknown command 'link_design' (CMD-005)
** Completed syntax/semantic check with 1 error(s) **
0
```

The following example shows the result when an unsupported command is in an SDC file. One SDC message is generated per instance of an unsupported command. At the end of the read, a summary of all unsupported commands in the file is generated. This SDC file was read into PrimeTime.

```
prompt> read_sdc t2.sdc -version 1.2
Reading SDC version 1.2...
Warning: Constraint 'set_logic_zero' is not supported by pt_shell. (SDC-3)
Warning: Constraint 'set_logic_zero' is not supported by pt_shell. (SDC-3)
Warning: Constraint 'set_logic_one' is not supported by pt_shell. (SDC-3)

Summary of unsupported constraints:
Information: Ignored 1 unsupported 'set_logic_one' constraint. (SDC-4)
Information: Ignored 2 unsupported 'set_logic_zero' constraints. (SDC-4)
1
```

The following example is a script that is not SDC-compliant, because it contains unsupported commands or command options.

```
current_design TOP
set_resistance -value 12.2 [get_nets i1t2]
```

The following example shows the output generated by **read_sdc** when reading the previous script. In SDC, **current_design** can be used only to get the current design; no arguments are allowed. Also, there is no **-value** option for **set_resistance**, so an appropriate message is generated.

```
prompt> read_sdc t3.tcl -echo -version 1.2
Reading SDC version 1.2...
current_design TOP
Error: extra positional option 'TOP' (CMD-012)
```

```
read_sdc
1384
```

```
set_resistance -value 12.2 [get_nets i1t2]
Error: unknown option '-value' (CMD-010)
0
```

The following example shows the message generated when an option is supported by the command but not by SDC.

Information: The '-value' option for set_resistance is unsupported. (CMD-038)

SEE ALSO

```
source(2)
write_sdc(2)
search_path(3)
sh_continue_on_error(3)
sh_script_stop_severity(3)
sh_source_uses_search_path(3)
```

read_sdf

Reads leaf cell and net timing information from a file in Standard Delay Format (SDF) and uses that information to annotate the current design.

SYNTAX

```
string read_sdf  
[-load_delay net | cell]  
[-path path_name]  
[-min_type sdf_min | sdf_typ | sdf_max]  
[-max_type sdf_min | sdf_typ | sdf_max]  
[-worst]  
[-min_file min_sdf_file_name]  
[-max_file max_sdf_file_name]  
sdf_file_name
```

Data Types

<i>path_name</i>	string
<i>min_sdf_file_name</i>	string
<i>max_sdf_file_name</i>	string
<i>sdf_file_name</i>	string

ARGUMENTS

-load_delay net | cell

Indicates whether load delays are included in net delays or in cell delays in the timing file being read. The load delay is the portion of cell delay arising from the capacitive load of the net driven by the cell. The default is **cell**.

-path *path_name*

Specifies the path from the current design to the subdesign for which the timing file has been created.

-min_type sdf_min | sdf_typ | sdf_max

Specifies which of the SDF triplet delay values are to be read from the SDF file for minimum delay. Delays in SDF are represented in the form of triplets (sdf_min:sdf_typ:sdf_max). If this is not specified, the command uses **sdf_min**.

-max_type sdf_min | sdf_typ | sdf_max

Specifies which of the SDF triplet delay values are to be read from the SDF file for maximum delay. Delays in SDF are represented in the form of triplets (sdf_min:sdf_typ:sdf_max). If this is not specified, the command uses **sdf_max**.

-worst

Indicates that **read_sdf** is to annotate the current design only with delays worse than the current annotated delays; this applies to annotated net and cell delays and annotated timing checks. For the hold timing check, the minimum numerical value is annotated after comparing the current annotated

value and the new value. For all other timing checks and delays, the maximum numerical value is annotated after comparing the current annotated value and the new value. Use this option when the timing file contains conditional timing.

To annotate the worst timing delay and timing checks of all timing conditions for each pin-to-pin annotation, use the **remove_annotated_delay** command with the **-all** option and the **remove_annotated_check** command before using the **read_sdf** command with the **-worst** option. Note that this behavior does not depend on the type of timing arc being read (that is, whether the arc is a setup or a hold arc), even though it might appear that minimum values are read for hold timing arcs. The correct behavior is one where all the values read from SDF files are either minimum, typical, or maximum, regardless of the type (setup or hold).

-min_file *min_sdf_file_name*

Specifies the file from which minimum delay timing information is to be read. The timing file must be in SDF format version v1.0, v2.0, or v2.1. Use this option only if the minimum and maximum delays are in two separate SDF files. The default is for minimum and maximum delays to be in a single SDF file.

-max_file *max_sdf_file_name*

Specifies the file from which maximum delay timing information is to be read. The timing file must be in SDF format version v1.0, v2.0, or v2.1. Use this option only if the minimum and maximum delays are in two separate SDF files. The default is for minimum and maximum delays to be in a single SDF file.

sdf_file_name

Specifies the name of the file from which the timing information is to be read. The timing file must be in SDF format version v1.0, or v2.1.

DESCRIPTION

This command reads from a disk file SDF leaf cell and net timing information and uses it to annotate the current design. The timing file must be in SDF format v1.0 or v2.1. Instance-specific pin-to-pin cell and net delays are read from the SDF file and annotated on the current design. When you specify the **-path** option, the command annotates the current design with information from a timing file created for an instance of a subdesign of the current design. When you specify a subdesign, you cannot annotate the current design with the net delays to the ports of the subdesign.

The load delay, also known as extra source gate delay, is that portion of the cell delay caused by the capacitive load of the driven net. Some delay calculators consider the load delay part of the net delay; others consider the load delay part of the cell delay. By default, the **read_sdf** command treats the load delay as part of the cell delay in the timing file being read. Use the **-load_delay net** option to indicate that your timing file includes the load delay in the net delay instead of in the cell delay.

Setup and hold, recovery, and removal timing checks, when present in the timing file, are used to annotate the current design. The SDF constructs for setup and hold are "SETUP" and "HOLD". The SDF construct for recovery and removal are "RECOVERY", "REMOVAL" and "RECREM."

Instance names in the design must match instance names in the timing file. For example, if the timing file is created from a design using VHDL naming conventions, the design must use VHDL naming conventions. To modify design names, use the **change_names** command.

To remove delays read and annotated by using the **read_sdf** command, use the **reset_design** or **remove_annotated_delay** command. To remove timing checks annotated with **read_sdf**, use the **remove_annotated_check** command.

Multicorner-Multimode Support

This command is not supported in a multi-scenario design flow.

EXAMPLES

The following example reads the adder.sdf timing file from disk and uses it to annotate the timing on the current design. The timing file contains load delays included in the cell delays.

```
prompt> read_sdf -load_delay cell adder.sdf
```

The following example reads the timing information of instance *u1* of design *MULT16* from the mult16_u1.sdf disk file, which contains the timing for instance *u1* of design *MULT16*. The timing is annotated on the design *MY_DESIGN*. In this case, load delay is included in the net delays.

```
prompt> current_design MY_DESIGN
prompt> read_sdf -load_delay net -path u1 mult16_u1.sdf
```

The following example reads minimum and maximum timing information from two separate SDF files and annotates the current design with delays corresponding to minimum and maximum operating conditions, respectively.

```
prompt> read_sdf -min_file min.sdf -max_file max.sdf
```

SEE ALSO

change_names(2)
remove_annotated_check(2)
remove_annotated_delay(2)
report_annotated_check(2)
report_annotated_delay(2)
reset_design(2)
set_annotated_check(2)
set_annotated_delay(2)
write_sdf(2)

read_tdf_ports

Reads in a Top Design Format (TDF) file and translates the "pin" and "pad" to mpc constraints for ports in the floorplan in the mpc flows.

SYNTAX

```
int read_tdf_ports  
[TDF file]  
-output string  
-apply  
-verbose
```

ARGUMENTS

TDF file

Specifies TDF file to be read. This argument is mandatory.

-output *string*

Specifies the output file created by the command containing the mpc constraints for the input data which can be applied to the design. This command will just write the output file if you want to apply at the same time are writing the output file then provide the -apply switch.

-apply

Specifies the mpc constraints generated should be applied directly to the design.

-verbose

Instructs the command to write out verbose messages while parsing and processing the data from TDF file.

DESCRIPTION

This command reads in the pin and pad functions from the Timing Definition Format (TDF) file and creates port constraints for MPC as specified in the **set_mpc_port_options** command.

If you are having trouble creating a TDF file you can create same constraints using the **set_mpc_port_options** command.

MPC flow supports pad placement by getting its location from the port it is connected to.

The following functions will be read in from a tdf file pad padName padSide [padorder] [padOffset] pin pinName layer width height pinSide [pinOrder] [pinOffset]

For details on tdf Syntax see the TDF syntax documentation.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following examples shows how read in a tdf file.

```
prompt> read_tdf_ports -output "port_cons.tcl" input_tdf.tdf prompt>
read_tdf_ports -output "port_cons.tcl" -apply input_tdf.tdf prompt>
read_tdf_ports -verbose input_tdf.tdf
```

SEE ALSO

```
create_placement(2)
set_mpc_options(2)
set_mpc_macro_options(2)
report_mpc_options(2)
report_mpc_port_options(2)
report_mpc_macro_options(2)
```

read_verilog

Reads in one or more design or library files in Verilog format.

SYNTAX

```
status read_verilog
[-dirty_netlist]
[-allow_black_box]
[-verbose]
[-bus_direction_for_undefined_cell connection | msb | lsb]
[-keep_module keep_module_list]
[-top top_module_name]
[-cell cell_name]
| verilog_files
```

Data Types

<i>keep_module_list</i>	list
<i>top_module_name</i>	string
<i>cell_name</i>	string
<i>verilog_files</i>	list

ARGUMENTS

-dirty_netlist

Controls whether the Verilog reader handles a dirty netlist. A dirty netlist is a Verilog netlist that contains objects that are inconsistent with the design libraries, such as floating pins, nets, and pins or ports. When the Verilog reader reads in a dirty netlist, it checks for the following missing or incomplete information issues warning messages, and then creates objects that allow the tool to continue:

- Missing reference cell in the reference library

There is no reference cell available in the reference libraries and the definition is not in the input Verilog file.

- Port definition is inconsistent with the reference library

There are additional ports present that are not available in the reference library.

- Port definition is inconsistent with the reference cell

There are additional ports present that are not available in the reference cell definition.

-allow_black_box

Allows the tool to continue if a module cannot be found in the reference libraries or in the input Verilog file.

If you do not specify this option, **read_verilog** issues an error message and quits when it finds an undefined module.

-verbose

Displays verbose information including possible top module names, black box

names, and module names with mismatched ports.

-bus_direction_for_undefined_cell connection | msb | lsb

Specifies the bus direction for an undefined cell. If an undefined module is instantiated, **read_verilog** determines the bus direction according to one of following three factors: connection, msb, or lsb. By default, it determines the bus direction from the connection.

-keep_module *keep_module_list*

Specifies the list of keep modules. For each module in the list, the Verilog reader does not expand it into the top cell. Instead, it expands it into a soft macro cell and instantiates it in the top cell. This is for the design planning top-down flow.

-top *top_module_name*

Specifies the top module name. The top module is the module that is not instantiated by any other modules. Generally there is only one top module in a design and **read_verilog** can scan and identify the top module automatically. However, if the design has multiple top modules, you must use this option to specify one of the modules.

-cell *cell_name*

Specifies the cell name. The tool creates the design in the Milkyway database using the specified cell name. If not specified, the tool uses the name of the top module as the cell name.

verilog_files

Specifies the name of one or more Verilog files to be read.

DESCRIPTION

The **read_verilog** command loads in netlist information from one or more Verilog files.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

SEE ALSO

read_zrt_route_guidance

Reads in a route guidance file.

SYNTAX

```
status read_zrt_route_guidance  
rgf_file_name
```

Data Types

rgf_file_name string

ARGUMENTS

rgf_file_name
Specifies the name of the route guidance file that captures the information that the router needs to modify the layout to fix DRC errors or lithography hotspots that were detected by a sign-off quality DRC or lithography checking tool. This file must follow the syntax described in the "Route Guidance File Format" section.
This is a required argument.

DESCRIPTION

This command reads a route guidance file that provides the router with information to modify the layout to fix DRC errors or lithography hotspots. The information is stored in a Milkyway error view called *<current_cell_name>_signoff_fix.err*. If the error view already exists, the command overwrites the existing data in the error view. You must open the CEL view of the associated design before running the **read_zrt_route_guidance** command.

Route Guidance File Format

The route guidance file must have the following syntax:

```
<?xml version="1.0"?>  
<RouteModificationInterface>  
  <Version>x.y</Version>  
  <Date>mm/dd/yyyy</Date>  
  <Unit>unit_value</Unit> # the units are in nm  
  <ErrorsAndGuidance>  
    <Category>  
      <Type>error_type</Type>  
      <Severity>error_severity</Severity>  
    <Error>  
      <Location>(llx, lly, urx, ury, layer)</Location>  
      <FixGuidance>  
        <MoveVia>  
          <Layer>layer_name</Layer>  
          <OriginalLocation>(x, y)</OriginalLocation>
```

```

        <NewLocation>(x, y)</NewLocation>
    </MoveVia>...
    <RemoveWires>
        (llx, lly, urx, ury, layer)
        ...
    </RemoveWires>
    <InsertWires>
        (llx, lly, urx, ury, layer)
        ...
    </InsertWires>
</FixGuidance>...
<AvoidMetalArea>
    (llx, lly, urx, ury, layer, block_weight)
    ...
</AvoidMetalArea>...
</Error>...
</Category>...
</ErrorsAndGuidance>
</RouteModificationInterface>

```

For more information about the route guidance file, see the *<i>IC Compiler Zroute User Guide</i>*.

EXAMPLES

In the following example, the **read_zrt_route_guidance** command reads the hotspot data defined in the file named `route_guidance.xml`.

```
prompt> read_zrt_route_guidance route_guidance.xml
```

SEE ALSO

`route_zrt_with_route_guidance(2)`

rebuild_mw_lib

Rebuilds the Milkyway library.

SYNTAX

```
status_value rebuild_mw_lib  
libName
```

Data Types

```
libName      string
```

ARGUMENTS

```
libName  
    Specifies the Milkyway library to be rebuilt. The value of mw_lib should be  
    a valid library name. The library must be closed for this command to work.
```

DESCRIPTION

This command rebuilds a Milkyway library by scanning all designs in the associated library directory.

The command returns a status indicating success or failure.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example rebuilds the current Milkyway library:

```
prompt> rebuild_mw_lib  
Scanning library...  
    place.CEL;5  
    place.CEL;4  
    place.CEL;3  
    place.CEL;2  
    place.CEL;1  
    place.EXP;1  
    place.NETL;1  
    place.PARA;1  
Rebuilding library...  
A total of 8 items have been rebuilt. (0 removed, 0 new added)  
1
```

SEE ALSO

`close_mw_lib(2)`
`copy_mw_lib(2)`
`create_mw_lib(2)`
`current_mw_lib(2)`
`open_mw_lib(2)`
`rename_mw_lib(2)`
`report_mw_lib(2)`

recover_tie_connection

Recover tie connection from direct PG implementation.

SYNTAX

```
status recover_tie_connection  
[-net pg_net_list]  
[-cell cell_list]  
[-hierarchy_based]
```

Data Types

pg_net_list list

ARGUMENTS

-net *pg_net_list*

If this option is specified, the recovery will be performed on these flat/hierarchical PG nets and the PG nets connecting to them. If anything in the list is not found or is not a PG net, the command will issue an error message and stop. If this option is not specified, the operation will be performed on all PG nets.

-cell *cell_list*

If a leaf cell is specified, the recovery will be performed on the tie pin/port which drives the tie on the leaf cell. If a hier cell is specified, the recovery will be performed on all the leaf cells within this hier cell as if all these leaf cells are specified.

If anything in the list is not found or is not a cell, the command will issue an error message and stop. If this option is not specified, the operation will be performed on the entire design.

-hierarchy_based

This option needs to be specified with "-cell" option. When both "-hierarchy_based" and "-cell" options are specified, "-cell" option has to specify hierarchical cells. Under this circumstance, the recovery will be performed on all ties within those specified hierarchical cells.

DESCRIPTION

This command disconnects PG net connection to a tie pin/port, and then connects the tie pin/port to a tie net.

EXAMPLES

The following example uses the **recover_tie_connection** to recover tie connection from direct PG implementation. The **report_tie_net** command is used to verify the Milkyway net connection before and after the command.

```
prompt> recover_tie_connection
```

```
1
prompt> report_tie_net
Design has 1 tie high net(s):

    net SNPS_LOGIC1 connects with following port(s) or pin(s):
        U222/A
1
```

SEE ALSO

`derive_pg_connection(2)`

redirect

Redirects the output of a command to a file.

SYNTAX

```
string redirect  
[-append] [-tee] [-file  
    | -variable  
    | -channel] [-compress]  
target  
{command_string}
```

Data Types

<i>target</i>	string
<i>command_string</i>	string

ARGUMENTS

-append

Appends the output to *target*.

-tee

Like the unix command of the same name, sends output to the current output channel as well as to the *target*.

-file

Indicates that *target* is a file name, and redirection is to that file. This is the default. It is exclusive of *-variable* and *-channel*.

-variable

Indicates that *target* is a variable name, and redirection is to that Tcl variable. It is exclusive of *-file* and *-channel*.

-channel

Indicates tha *target* is a Tcl channel, and redirection is to that channel. It is exclusive of *-variable* and *-file*.

-compress

Compress when writing to file. If redirecting to a file, then this option specifies that the output should be compressed as it is written. The output file is in a format recognizable by "gzip -d". You cannot specify this option with *-append*.

target

Indicates the target of the output redirection. This is either a filename, Tcl variable, or Tcl channel depending on the command arguments.

command_string

The command to execute. Intermediate output from this command, as well as the result of the command, will be redirected to *target*. The *command_string* should be rigidly quoted with curly braces.

DESCRIPTION

The **redirect** command performs the same function as the traditional unix-style redirection operators `>` and `>>`. The *command_string* must be rigidly quoted (that is, enclosed in curly braces) in order for the operation to succeed.

Output is redirected to a file by default. Output can be redirected to a Tcl variable by using the *-variable* option, or to a Tcl channel by using the *-channel* option.

Output can be sent to the current output device as well as the redirect target by using the *-tee* option. See the examples section for an example.

The result of a **redirect** command which does not generate a Tcl error is the empty string. Screen output occurs only if errors occurred during execution of the *command_string* (other than opening the redirect file). When errors occur, a summary message is printed. See the examples.

Although the result of a successful **redirect** command is the empty string, it is still possible to get and use the result of the command that you redirected. Construct a **set** command in which you set a variable to the result of your command. Then, redirect the **set** command. The variable holds the result of your command. See the examples.

The **redirect** command is much more flexible than traditional unix redirection operators. With **redirect**, you can redirect multiple commands or an entire script. See the examples for an example of how to construct such a command.

Note that the builtin Tcl command **puts** does not respond to output redirection of any kind. Use the builtin **echo** command instead.

EXAMPLES

In the following example, the output of the plus procedure is redirected. The echoed string and the result of the plus operation is in the output file. Notice that the result was not echoed to the screen.

```
prompt> proc plus {a b} {echo "In plus" ; return [expr $a + $b]}
prompt> redirect p.out {plus 12 13}
prompt> exec cat p.out
In plus
25
```

In this example, a typo in the command created an error condition. The error message indicates that you can use **error_info** to trace the error, but you should first check the output file.

```
prompt> redirect p.out {plus2 12 13}
Error: Errors detected during redirect
      Use error_info for more info. (CMD-013)
prompt> exec cat p.out
Error: unknown command 'plus2' (CMD-005)
```

In this example, we explore the usage of results from redirected commands. Since the result of **redirect** for a command which does not generate a Tcl error is the empty string, use the **set** command to trap the result of the command. For example, assume that there is a command to read a file which has a result of "1" if it succeeds, and "0" if it fails. If you redirect only the command, there is no way to know if it succeeded.

```
redirect p.out {read_a_file "a.txt"}
# Now what?  How can I redirect and use the result?
```

But if you set a variable to the result, then it is possible to use that result in a conditional expression, etc.

```
redirect p.out {set rres [read_a_file "a.txt"]}
if {$rres == 1} { echo "Read ok!"
}
```

The **redirect** command is not limited to redirection of a single command. You can redirect entire blocks of a script with a single **redirect** command. This simple example with **echo** demonstrates this feature:

```
prompt> redirect p.out {
?      echo -n "Hello "
?      echo "world"
?      }
prompt> exec cat p.out
Hello world
prompt>
```

The **redirect** command allows you to tee output to the previous output device and also to redirect output to a variable. This simple example with **echo** demonstrates these features:

```
prompt> set y "This is "
This is
prompt> redirect -tee x.out {
      echo XXX
      redirect -variable y -append {
      echo YYY
      redirect -tee -variable z {
      echo ZZZ
      }
      }
      }
XXX
prompt> exec cat x.out
XXX
prompt> echo $y
This is YYY
ZZZ
```

```
prompt> echo $z  
ZZZ
```

SEE ALSO

```
echo(2)  
error_info(2)
```


redo

Redoes last undo operation

SYNTAX

```
status redo
[-all]
[-mark string]
```

ARGUMENTS

-all
Redo all operations in undo stack

-mark *string*
Redo all operations in undo stack to the named mark

DESCRIPTION

This command redoes the last undone operation. The redone command can be undone using the **undo** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example undoes the result of **move_objects** and then redoes it.

```
prompt> move_objects -to {1400 1600} [get_route_guides *]
prompt> undo
prompt> redo
```

SEE ALSO

`undo(2)`
`undo_mark(2)`
`undo_config(2)`

reduce_fp_rail_stacked_via

Given a complete or partial power network, you can select a set of stacked vias in the power network for removal to reduce congestion. This command honors the specified voltage (IR) drop constraint and maintains power network connectivity. The layer-based global route congestion maps are required for this command.

route_zrt_global, **route_global** or **route_fp_proto** can be used to create the congestion maps. The **reduce_fp_rail_stacked_via** command does not physically remove the selected stacked vias. Signal vias are not removed. Power network stacked vias in hard-macros or soft-macros are not removed. Power network stacked via reduction does not support running more than two nets at a time.

SYNTAX

```
status reduce_fp_rail_stacked_via
-nets nets
[-power_budget power]
[-analyze_power]
[-lowest_voltage_drop]
[-target_voltage_drop target_voltage]
[-voltage_supply voltage]
[-effort low | medium | high]
[-pad_lib_cells pad_lib_cells]
[-read_pad_instance_file file_name]
[-read_pad_lib_cell_file file_name]
[-extract_stacked_via_only]
[-use_pins_as_pads]
[-top_level_only]
[-ignore_blockages]
[-ignore_conn_view_layers layer]
[-read_power_compiler_file file_name]
[-read_prime_power_file file_name]
[-read_default_power_file file_name]
[-output_directory directory_name]
```

Data Types

<i>nets</i>	collection or list
<i>power</i>	float
<i>target_voltage</i>	float
<i>voltage</i>	float
<i>pad_lib_cells</i>	string
<i>file_name</i>	string
<i>layer</i>	collection of list
<i>directory_name</i>	string

ARGUMENTS

```
-nets nets
    Specifies the names of the power or ground nets on which to perform power
    network stacked via reduction.
```

-power_budget *power*

Specifies a total power budget for the design. The power budget is divided among the instances according to their sizes. For hard macros or standard cells, the power budget is computed based on the percentage of the total area. For a hierarchical block, it is computed based on the sum of all cells and blocks inside it. The power unit is in milliwatts. The default is **1000**.

-analyze_power

Specifies power analysis for each cell instance from IC Compiler by considering specified switching probabilities. Before using this option for power analysis, the default toggle rate (default is 0.1) and default static probability (default is 0.5) for primary inputs and black box outputs can be set by using the **set power_default_toggle_rate** and **set power_default_static_probability** commands. The toggle rate and/or static probability for a specified object, for example, a net, pin or port can be set by using the **set switching_activity** command. The switching activity interchange format (SAIF) file may also be read by using the **read_saif** command. The **set_cell_internal_power** command can be used to set the power value of a pin per toggle. Refer to the **report_power** command for more detailed information. It is recommended that you run the **report_power -cell -flat -nosplit** command before you use **-analyze_power** option.

The **-analyze_power** option can be used concurrently with the **-power_budget** option. If a power budget is specified using **-power_budget** together with **-analyze_power** and the power budget is larger than the sum of the instance power calculated by **-analyze_power**, the difference between the power budget and the total power by **-analyze_power** will be assigned to those instances that are not assigned in **-analyze_power** if there are any such instances. Otherwise, the power budget is ignored. By default, the power budget is set to 0 with **-analyze_power** option.

-analyze_power can also be used concurrently with **-read_default_power_file**. The power specified in the default power file has a higher priority than that calculated by the **-analyze_power** option, that is, if the power value is calculated and read from the default power file for one instance, the instance is assigned with the power from the default power file.

"**-analyze_power**" is mutually exclusive with "**-read_power_compiler_file**" and "**-read_prime_power_file**".

By default, power is NOT calculated from IC Compiler unless this option is explicitly used.

-lowest_voltage_drop

Specifies power network stacked via reduction to maintain the voltage (IR) drop of the existing power network when removing stacked vias. By default, this option is off.

-target_voltage_drop *target_voltage*

Specifies target voltage (IR) drop constraint for power network stacked via reduction when stacked vias are removed from the existing network. The unit is millivolt.

-voltage_supply *voltage*

Specifies the supply voltage for the power network. The voltage unit is volts. The default is **1.5**.

-effort low | medium | high

Specifies the CPU effort for power network stacked via reduction. The default

is medium effort.

-pad_lib_cells *pad_lib_cells*

Specifies the pad library cells and the associated net as follows

net_name:pad_lib_cell_name

If net name is not specified, the pad library cell is used for both power and ground nets. By default, all the pads logically connected to signal **1** or **0** are assumed as power pads.

-read_pad_instance_file *file_name*

Specifies the pad instance file name and optionally, the net name. If you do not specify the net name, the command assumes that the specified pad instance is used for both power and ground nets.

The *instance_and_net* format is as follows:

instance_name [*net_name*]

For example, a valid *instance_and_net* value might be **VDD1 VDD**. By default, this option is off.

-read_pad_lib_cell_file *file_name*

Specifies the pad library cell file name and optionally, the net name. If you do not specify the net name, the command assumes that the specified pad library cell is used for both power and ground nets. The

lib_cell_name_and_net format is as follows:

lib_cell_name [*net_name*]

For example, a valid *lib_cell_name_and_net* value might be **VDD.FRAME VDD**. By default, this option is off.

-extract_stacked_via_only

Extracts all the removable stacked vias from the existing power network. The removable stacked vias can be extracted without affecting the power network connectivity, that is, each instance can still connect to a power pad through the power network, even when these removable stacked vias are removed.

This option also calculates the total congestion overflow and estimates the total congestion overflow due to all stacked vias in the power network, and the maximum potential overflow reduction if all removable stacked vias were removed. Voltage (IR) drop analysis is NOT performed with this option. In other words, this option estimates the most optimistic congestion reduction without considering the IR drop impact. When this option is on, options such as **-power_budget**, **-analyze_power**, **-lowest_voltage_drop**, **-target_voltage_drop**, **-voltage_supply**, **-effort** are ignored because the IR drop impact is not considered. By default, this option is off.

-use_pins_as_pads

Allows power network synthesis to use PG pins on the design boundary as PG pads. This option is useful in block-level simulation. The default is off.

-top_level_only

Ignores cells inside soft macros, but considers all the power and ground net wires and vias in the design. By default, all cell instances in each soft macro are flattened to the top-level design. Note that this option is only used for power network analysis purpose. The stacked vias in soft-macros are not to be removed. The default is off.

-ignore_blockages
 Ignores blockages due to hard macro blocks for virtual pad pseudo connection. By default, this option is off.

-ignore_conn_view_layers layer
 Ignores all metal layers (select the All option) or ignores specified metal layers (select the Specified option) in connectivity (.CONN) views. By default, this option is off and none of the metal layers is ignored.

-read_power_compiler_file file_name
 Obtains the power calculation source from a Power Compiler report file. By default, this option is off.

-read_prime_power_file file_name
 Obtains the power calculation source from a Prime Power/Prime Time PX report file. By default, this option is off.

-read_default_power_file file_name
 Uses the default format to obtain a power calculation. Refer to the **synthesize_fp_rail** or **analyze_fp_rail** commands for more detailed information regarding the default power file format.
 By default, the command also reads a power information file generated from AstroRail by using the **poDumpPowerInfo** command.

-output_directory directory_name
 Specifies the name of the directory in which to store the IR drop and electromigration results with the file name *design_name.net_name.pw_hl.pna* and the power allocation results with the file name *design_name.net_name.power*.
 The default is *pna_output*.

DESCRIPTION

This command allows you to perform power network stacked via reduction in a complete or partially built PG network. The layer-based global route congestion maps are required for this command. **route_zrt_global**, **route_global** or **route_fp_proto** can be used to create the congestion maps. A set of stacked vias in a congested area is selected for removal such that the voltage (IR) drop constraint is satisfied and the connectivity of the existing power network is maintained. After performing stacked via reduction, you can preview the IR drop map and electromigration (EM) if the selected stacked vias were removed. This command also estimates the congestion overflow reduction based on the existing congestion map if the selected stacked vias were to be removed. You can also find out where stacked vias are to be removed in the updated IR drop map. (The stacked via is not in the IR drop map if it is to be removed.) If the results and constraints are satisfied, you can commit the stacked via removal and remove these objects from the database by using the **remove_fp_rail_stacked_via** command.

If you use the **-extract_stacked_via_only** option, this command only estimates the congestion due to all stacked vias and the maximum potential reduction in congestion if all removable stacked vias were removed without affecting power network connectivity. IR drop impact is NOT considered with this option. With this option, you are able to decide if it is worthwhile to perform the actual stacked via reduction and reduce the turn-around time.

EXAMPLES

The following example shows how to perform stacked via reduction with medium effort on net *VDD* and *VSS* in the power network. The total power budget in the design is 1000mW, the voltage supply is 1.2V, the pad library cell is *pvdi.FRAME* for net *VDD* and *pvdo.FRAME* for net *VSS*, the lowest target IR drop is used for each net.

```
prompt> reduce_fp_rail_stacked_via\  
        -nets {VDD VSS}\  
        -lowest_voltage_drop\  
        -effort medium\  
        -power_budget 1000\  
        -voltage_supply 1.2\  
        -pad_lib_cells "VDD:pvdi.FRAME VSS:pvdo.FRAME"
```

The following example shows how to estimate the congestion due to all stacked vias in net *VDD* and the maximum potential congestion overflow reduction if all removable stacked vias were removed without considering IR drop impact but maintaining the network connectivity.

```
prompt> reduce_fp_rail_stacked_via\  
        -nets VDD\  
        -extract_stacked_via_only\  

```

SEE ALSO

```
create_fp_virtual_pad(2)  
analyze_fp_rail(2)  
load_fp_rail_map(2)  
remove_fp_rail_stacked_via(2)  
route_global(2)  
route_zrt_global(2)  
route_fp_proto(2)
```

refine_fp_macro_channels

Refines channels between macros to avoid congestion.

SYNTAX

```
status refine_fp_macro_channels
```

ARGUMENTS

The **refine_fp_macro_channels** command has no arguments.

DESCRIPTION

This command moves macros to adjust channels between them to the width as close to the optimal width based on the congestion map as possible. The command needs a congestion map to run. It will attempt to use the global route congestion map; if unavailable it will attempt to use the placement congestion map; if that one is unavailable as well, it will produce an error. The command will not touch fixed hard macros. Non-fixed hard macros must be inside core and not overlap each other or blockages. For a corresponding command to adjust channels between plan groups, see `shape_fp_blocks -adjust_channels`.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLE

```
prompt> refine_fp_macro_channels
```

SEE ALSO

`shape_fp_blocks(2)`

refine_placement

Performs incremental placement with congestion optimization.

SYNTAX

```
int refine_placement  
[-congestion_effort low | medium | high]  
[-perturbation_level min | medium | high | max]  
[-ignore_scan]  
[-num_cpus number_of_cpus]  
[-coordinate {X1 Y1 X2 Y2}]
```

Data Types

<i>number_of_cpus</i>	integer
<i>X1</i>	float
<i>Y1</i>	float
<i>X2</i>	float
<i>Y2</i>	float

ARGUMENTS

-congestion_effort low | medium | high
Specifies the effort level of the incremental congestion optimization. The default effort level is **medium**.

-perturbation_level min | medium | high | max
Specifies the perturbation level for the placement in order to optimize congestion. Specifying **min** produces the least amount of movement from the placement. The higher the level specified, the more movement you may expect from the placement, so specifying **max** produces the most movement from the placement. The default effort level is **medium**.

-ignore_scan
Ignores scan chain connections during placement.

-num_cpus *number_of_cpus*
Specifies the number of CPUs used in parallel during coarse placement. The *number_of_cpus* is an integer value that is less than or equal to the number of free CPUs on your machine and greater than or equal to 1. If unspecified, the default is to use 1 CPU.

-coordinate {X1 Y1 X2 Y2}
Specifies the coordinates of the area from which to run incremental congestion removal. The default is to run incremental congestion removal for the entire design.

DESCRIPTION

The **refine_placement** command performs incremental congestion optimization for the current design. This command does not touch netlist and it provides a legalized

placement at the end.

Multicorner-Multimode Support

This command uses information from active scenarios only.

EXAMPLES

The following example runs the **refine_placement** command in low effort congestion mode:

```
prompt> refine_placement -congestion_effort low
```

SEE ALSO

```
create_placement(2)  
remove_congestion_options(2)  
report_congestion(2)  
report_congestion_options(2)  
set_congestion_options(2)
```

remove_all_spacing_rules

Removes all inter-cell spacing rules from the current library.

SYNTAX

```
status remove_all_spacing_rules
```

ARGUMENTS

There are no arguments for this command

DESCRIPTION

This command removes permanently inter-cell spacing rules that have been set with commands `set_lib_cell_spacing_label` and `set_spacing_label_rule` from the currently opened library. In order to re-instate inter-cell spacing constraints, the user will have to rerun `set_lib_cell_spacing_label` and `set_spacing_label_rule` commands. If there are rules defined in multiple libraries, this command must be run for each library in order to delete all inter-cell spacing rules.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

In the following example all rules are removed.

```
prompt> remove_all_spacing_rules
```

```
Info: Removing rules from library tsmc18_6lm
```

SEE ALSO

```
set_lib_cell_spacing_label(2)  
set_spacing_label_rule(2)  
report_spacing_rules(2)
```

remove_annotated_check

Removes annotated timing check information.

SYNTAX

```
int remove_annotated_check  
-all | -from from_list | -to to_list  
[-rise | -fall]  
[-clock rise | fall]  
[-setup] [-hold]  
[-recovery] [-removal]  
[-nochange_low] [-nochange_high]
```

Data Types

```
from_list      list  
to_list        list
```

ARGUMENTS

-all

Specifies to remove all annotated checks in the current design. This includes rise and fall values for setup, hold, recovery, removal, and nochange pin-to-pin timing checks. You must specify either **-all**, **-from**, or **-to**; if you specify **-all**, you cannot specify any other arguments or options.

-from *from_list*

Specifies a list of leaf cell clock pins that are the startpoints of the timing arcs from which annotated checks are to be removed. You must specify either **-all**, **-from**, or **-to**; if you specify **-all**, you cannot specify any other arguments or options.

-to *to_list*

Specifies a list of leaf cell data pins that are the endpoints of the timing arcs from which annotated checks are to be removed. You must specify either **-all**, **-from**, or **-to**; if you specify **-all**, you cannot specify any other arguments or options.

-rise | -fall

Specifies whether the check to be removed is for data rise or fall transition. If you don't specify either **-rise** or **-fall**, both values are removed.

-clock rise | fall

Specifies whether the check to remove is for clock rising or falling. By default, checks for both clock rise and fall are removed.

-setup

Indicates to remove only setup timing check information.

-hold

Indicates to remove only hold timing check information.

-recovery
Indicates to remove only recovery timing check information.

-removal
Indicates to remove only removal timing check information.

-nochange_low
Indicates to remove only the nochange timing check against data low information.

-nochange_high
Indicates to remove only the nochange timing check against data high information.

DESCRIPTION

Removes annotated timing checks between the specified pins. Data rise and fall and clock rise and fall annotated checks are removed by default.

You can use the **remove_annotated_check** command for pins at lower levels of the design hierarchy. These pins are specified as "INSTANCE1/INSTANCE2/PIN_NAME."

If the design is not already linked, **remove_annotated_check** links it automatically.

The **remove_annotated_check** command removes annotated timing checks set by **set_annotated_check**, and also removes setup, hold, recovery, removal, or nochange annotated timing checks set by **read_sdf**. The **reset_design** command removes annotated timing checks in addition to other information.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example removes an annotated setup check between pins "u1/u2/CP" and "u1/u2/D".

```
prompt> remove_annotated_check -setup -from u1/u2/CP -to u1/u2/D
```

The following example removes all annotated timing checks on the current design.

```
prompt> remove_annotated_check -all
```

SEE ALSO

current_design(2)
link(2)
read_sdf(2)
report_annotated_check(2)
reset_design(2)
set_annotated_check(2)

remove_annotated_delay

Removes the annotated delay between two pins.

SYNTAX

```
int remove_annotated_delay  
-all  
  | -cell_all  
  | -net_all  
  | -non_clock_cell_all  
  | -non_clock_net_all  
  | -from from_list  
  | -to to_list
```

Data Types

```
from_list    list  
to_list      list
```

ARGUMENTS

- all**
Specifies to remove all net and cell annotated delays in the current design. You must specify either **-all**, **-from**, or **-to**; if you specify **-all**, you cannot specify any other arguments.
- cell_all**
Specifies to remove all cell annotated delays in the current design.
- net_all**
Specifies to remove all net annotated delays in the current design. The user can specify **-cell_all** and **-net_all** together to remove all cell and net annotated delays.
- non_clock_cell_all**
Specifies to remove all cell annotated delays in the current design except those on the clock network.
- non_clock_net_all**
Specifies to remove all net annotated delays in the current design except those on the clock network. The user can specify **-non_clock_cell_all** and **-non_clock_net_all** together to remove all cell and net annotated delays not on the clock network.
- from *from_list***
Specifies a list of leaf cell pins or top-level ports that are the startpoints of the timing arcs from which to remove annotated delays. You must specify either **-all**, **-from**, or **-to**; if you specify **-all**, you cannot specify any other arguments.
- to *to_list***
Specifies a list of leaf cell pins or top-level ports that are the endpoints

of the timing arcs from which to remove annotated delays. You must specify either **-all**, **-from**, or **-to**; if you specify **-all**, you cannot specify any other arguments.

DESCRIPTION

Removes annotated delays between the specified pins, which must be on the same net or on the same cell. Both rise and fall annotated delays are removed. There must be a timing arc between the pins in *from_list* and the pins in *to_list* or no annotated delay is removed.

You can use **remove_annotated_delay** for pins at lower levels of the design hierarchy. These pins are specified as "INSTANCE1/INSTANCE2/PIN_NAME."

If the current design is hierarchical, you must link the design with **link -all** before using **remove_annotated_delay**.

To annotate net delay values between pins in a design, use **set_annotated_delay -net**. To annotate cell delay values between pins in a design, use **set_annotated_delay -cell**. You must use **-from** and **-to** in the same manner you used them to set annotated values. For example, use **remove_annotated_delay -from** to remove an annotated delay set with **set_annotated_delay -from**. To remove an annotated delay set with **set_annotated_delay -to**, use **remove_annotated_delay -to**. To remove an annotated delay set with **set_annotated_delay -from -to**, use **remove_annotated_delay -from -to**.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example removes an annotated net delay between output pin "Z" of cell instance "U1/U2/U3" and input pin "A" of cell instance "U6". The delay to remove was annotated with the **set_annotated_delay -net <value> -from U1/U2/U3/Z -to U6/A**.

```
prompt> remove_annotated_delay -from U1/U2/U3/Z -to U6/A
```

The following example removes the annotated lumped net delay between output pin "Z" of cell instance "U1/U2/U3" and all fanout pins on that net. The delay to be removed was annotated with **"set_annotated_delay -net <value> -from U1/U2/U3/Z"**.

```
prompt> remove_annotated_delay -from U1/U2/U3/Z
```

The following example removes all annotated net and cell delays from the current design.

```
prompt> remove_annotated_delay -all
```

SEE ALSO

current_design(2)
link(2)

remove_annotated_delay
1416

```
report_timing(2)
reset_design(2)
set_annotated_delay(2)
```

remove_annotated_transition

Removes the annotated transition at a pin.

SYNTAX

```
int remove_annotated_transition  
-all
```

ARGUMENTS

-all

Specifies to remove annotated transition time from every pin in the current design. You must specify either **-all** or **-at**; if you specify **-all**, you cannot specify any other arguments.

DESCRIPTION

Removes annotated transitions at specified pins. Both rise and fall annotated transitions will be removed.

To annotate net transition values at pins in a design, use **set_annotated_transition**.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example removes the annotated transition at every pin in the design.

```
prompt> remove_annotated_transition -all
```

The following example removes the annotated transition at the output pin "Z" of cell instance "U1/U2/U3". The transition to be removed was annotated with "set_annotated_transition value U1/U2/U3/Z".

```
prompt> remove_annotated_transition -at U1/U2/U3/Z
```

SEE ALSO

```
current_design(2)  
link(2)  
report_timing(2)  
reset_design(2)  
set_annotated_transition(2)
```


remove_annotations

Removes all annotated information on the design.

SYNTAX

```
status remove_annotations
```

ARGUMENTS

The `fbremove_annotations` command has no arguments.

DESCRIPTION

The **remove_annotations** command removes all annotations on the design, including delay, transition, resistance, capacitance, and check. These include the data set by the **set_annotated_check**, **set_annotated_delay**, **set_annotated_transition**, **set_resistance**, and **set_load** commands.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example removes all annotated information set by previous commands:

```
prompt> set_annotated_check 0.25 -from m2/st_reg[0]/CP -to m2/st_reg[0]/D -setup
Warning: There is no 'setup_falling' timing arc between pins
        'm2/st_reg[0]/CP' and 'm2/st_reg[0]/D'. (OPT-815)
1

prompt> set_annotated_delay -cell 0.03 -from m1/U18/A -to m1/U18/Z
Information: Annotated 'cell' delays are assumed to include load delay. (UID-282)
1

prompt> set_annotated_transition 0.07 m2/st_reg[0]/D
1

prompt> remove_annotations
1
```

SEE ALSO

```
current_design(2)
link(2)
remove_annotated_check(2)
remove_annotated_delay(2)
remove_annotated_transition(2)
```

```
remove_sdc(2)
report_timing(2)
reset_design(2)
set_load(2)
set_resistance(2)
```

remove_antenna_rules

Deletes all of the antenna rules stored in the library.

SYNTAX

```
status_value remove_antenna_rules  
[mw_lib]
```

Data Types

mw_lib list

ARGUMENTS

mw_lib
Specifies the Milkyway library to be updated. The value of *mw_lib* can be a library name or a one-element collection of a library. The *mw_lib* option is optional. The default is to use the current Milkyway library.

DESCRIPTION

This command deletes all of the antenna rules stored in the library. The command returns a status indicating success or failure.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

```
prompt> remove_antenna_rules
```

SEE ALSO

```
define_antenna_rule(2)  
define_antenna_layer_ratio_scale(2)  
define_antenna_layer_rule(2)  
report_antenna_rules(2)
```

remove_aocvm

Removes advanced on-chip variation (OCV) derate factor information.

SYNTAX

```
status remove_aocvm  
[-coefficient]  
[-derate]  
[object_list]
```

Data Types

object_list list

ARGUMENTS

-coefficient

Causes only advanced OCV coefficients to be removed. The **-coefficient** and **-derate** options are mutually exclusive.

-derate

Causes only advanced OCV derate factors to be removed. The **-coefficient** and **-derate** options are mutually exclusive.

object_list

If **-coefficient** has been specified, causes the advanced OCV coefficients set on the cells or library cells in the *object_list* to be removed. If **-derate** has been specified, causes the advanced OCV derate factors set on the design, hierarchical cells, or library cells in the *object_list* to be removed.

DESCRIPTION

Removes user-specified advanced on-chip variation (OCV) coefficients and derate factors that were previously set by the **set_aocvm_coefficient** and **read_aocvm** commands. If the command is specified with no options, both the coefficients and derate factors are removed.

To display advanced OCV derate factors and coefficients, use the **report_aocvm** command.

EXAMPLES

The following example removes the random advanced OCV coefficients from the library cell named **lib/bufA**.

```
prompt> remove_aocvm -coefficient lib/bufA
```

The following example removes all advanced OCV derate factors that were set using the **read_aocvm** command.

```
prompt> remove_aocvm -derate
```

The following example removes all advanced OCV derate factors from the current design.

```
prompt> remove_aocvm -derate [current_design]
```

SEE ALSO

```
read_aocvm(2)  
report_aocvm(2)  
set_aocvm_coefficient(2)  
timing_aocvm_enable_analysis(3)
```

remove_attribute

Removes an attribute from the specified objects.

SYNTAX

```
collection remove_attribute
[-class class_name]
[-quiet]
object_list
attribute_name
```

Data Types

<i>class_name</i>	string
<i>object_list</i>	list
<i>attribute_name</i>	string

ARGUMENTS

-class *class_name*
Specifies the class name for the object specified in *object_list*, if the element of *object_list* is a name. Valid classes are design, port, cell, net, etc.

-quiet
Turns off the warning messages that would otherwise be issued if the attribute or objects are not found.

object_list
Specifies a list of objects from which the attribute is to be removed. Each element in the list is either a collection or a pattern that is combined with the *class_name* to find the objects.

attribute_name
Specifies the name of the attribute to be removed.

DESCRIPTION

This command removes the specified attribute from the specified objects. For a complete listing of attributes, refer to the **attributes** man page.

This command creates a collection of objects that have the specified attribute removed. A returned empty string indicates that no object has been removed.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows the first command defining a new attribute named "X" on the net. The second command sets the "X" attribute to 30 on all the nets in the current hierarchy. The third command removes the "X" attribute from the nets named VSS and VDD. The fourth command retrieves the "X" attribute from the net named VDD. The fifth command retrieves the "X" attribute from a net named in.

```
prompt> define_user_attribute -type int -class net X
Info:User-defined attribute 'X' on class 'net'.
1
prompt> set_attribute [get_nets * -all] X 30
{"out", "in", "VDD", "VSS"}
prompt> remove_attribute [get_nets V* -all] X
{"VDD", "VSS"}
prompt> get_attribute [get_nets VDD -all] X
ERROR : Failed to get attr(X)'s value.
prompt> get_attribute [get_nets in] X
30
```

SEE ALSO

```
collections(2)
define_user_attribute(2)
get_attribute(2)
list_attributes(2)
reset_design(2)
set_attribute(2)
```

remove_base_arrays

Removes base arrays from the current design.

SYNTAX

```
status_value remove_base_arrays  
[-all | pattern]
```

Data Types

pattern string

ARGUMENTS

-all

Specifies to remove all base arrays from the current design. The **-all** option is mutually exclusive with *patterns*.

pattern

Specifies to remove the base arrays that match the specified *pattern*. The *patterns* value is mutually exclusive with the **-all** option.

DESCRIPTION

The **remove_base_arrays** command removes base arrays from the current design. Specify the **-all** option to remove all base array records from the current design. Specify a pattern of base arrays to remove only the base arrays that match the specified pattern.

The command returns a value of **1** if it succeeds, or **0** if the command fails.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes all base arrays in the design:

```
prompt> remove_base_arrays -all  
1
```

SEE ALSO

create_base_array(2)

remove_bounds

Removes bounds from the current design.

SYNTAX

```
int remove_bounds  
[-verbose]  
[-all]  
[-name bound_name_list]  
objects
```

Data Types

<i>bound_name_list</i>	list
<i>objects</i>	string

ARGUMENTS

-verbose
Specifies to print information about what are being deleting.

-all
Specifies to remove all bounds from the current design. This command removes only bounds that have been created by using the **create_bounds** command. Bounds created in an input PDEF file cannot be removed. By default, this option is off.

-name *bound_name_list*
Specifies to remove bounds with the given names. By default, this option is off.

objects
Specifies bound objects to be removed. You can use the *get_bounds* command to specify the objects.

DESCRIPTION

This command removes bounds constraints from the current design. If you specify the **-all** option, all bounds are be removed from the design, with the exception of those specified in an input PDEF file. If you give a list of bound names, the bounds with the specified names are removed. The ID is displayed every time a bound is created. You can use the **report_bounds** command to view the ID.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes bounds named *foo_0* and *foo_1*.

```
prompt> remove_bounds -name {foo_0 foo_1}  
prompt> remove_bounds [get_bounds *]
```

SEE ALSO

```
create_bounds(2)  
create_placement(2)  
get_bounds(2)  
report_bounds(2)  
set_cell_location(2)  
update_bounds(2)
```

remove_buffer

Removes the buffer cells at a specified driver pin or net on a mapped design.

SYNTAX

```
status remove_buffer  
-from start_point  
-net net_list  
[-to end_point_list]  
[-level integer]  
cell_list
```

Data Types

<i>start_point</i>	list
<i>net_list</i>	list
<i>end_point_list</i>	list
<i>integer</i>	integer
<i>cell_list</i>	list

ARGUMENTS

-from *start_point*

Specifies the starting pin or port of the fanout tree from which the buffer(s) or inverter pair(s) need to be removed. Only driver pins or driver ports can be specified for this option.

This option **-from** is mutually exclusive with **-net** and **cell_list** options.

-net *net_list*

Specifies the starting net of the fanout tree from which the buffer(s) or inverter pair(s) need to be removed. Only one net can be specified for this option and the driver pin of this net would be considered as the start for fanout traversal. i.e. all the load cells connected to the net is considered in buffer tree.

This option **-net** is mutually exclusive with **-from** and **cell_list** options.

-to *end_point_list*

Specifies the ending pin(s) or port(s) where to stop the fanout traversal. Only those buffer(s) or inverter pair(s) that are in the fanin path of **-to** and the fanout path of **-from** or **-net** will be removed.

This option **-to** is mutually exclusive with **-level** option.

-level *integer*

Specifies the number of levels of buffers or inverter pairs to remove. If you specify an *integer* value that is less than the number of buffers or inverters between those specified by the *start_point_list* and *end_point_list* arguments, clean up occurs only on the *integer* number of levels from the value of *end_point_list*. Valid **-level** values are from 1 to 1,000,000. By default, value 1 is used for this option.

This option **-level** is mutually exclusive with **-to** option.

cell_list

Specifies the buffer or inverter cell(s) to remove. Each buffer is a cell name or a collection of netlist cells. When inverter cells are specified, it should be specified in pairs.

This option **cell_list** is mutually exclusive with the **-to** and **-from** options.

DESCRIPTION

This command removes buffer cells when you specify a *cell_list* and the buffer tree at the specified pins, or net on a mapped design. When a driver pin or a net is given, the buffer tree is removed with this driver as the root. The buffer tree removal process starts at the source and removes all buffer and inverter cells in its transitive fanout until it finds nonbuffer cells or when the **-level** number of transitive fanouts is reached.

This command does not remove the buffer tree across the hierarchy.

This command can remove inverters only as pairs. You must specify both of the pairing inverters if you use the *cell_list* argument. The *end_point_list* specified in the **-to** option can contain only load pins of the second inverter in the inverter pair. An inverter pair is considered as one level when decrementing the **-level**.

This command accepts the final implementation of buffer tree removal, even if it negatively impacts the timing or Design Rule Checking (DRC) violations in the design.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes a buffer cell.

```
prompt> remove_buffer -from cell11/O
Disconnecting net 'net1' from pin 'cell12/I'.
Disconnecting net 'net2' from pin 'cell15/I'.
Disconnecting net 'net2' from pin 'cell12/O'.
Removing net 'net2' in design 'Top'.
Connecting net 'net1' to pin 'cell15/I'.
Removing cell 'cell12' in design 'Top'.
1
```

```
prompt> remove_buffer -net net1
Disconnecting net 'net1' from pin 'cell12/I'.
Disconnecting net 'net2' from pin 'cell15/I'.
Disconnecting net 'net2' from pin 'cell12/O'.
Removing net 'net2' in design 'Top'.
Connecting net 'net1' to pin 'cell15/I'.
Removing cell 'cell12' in design 'Top'.
1
```

```
prompt> remove_buffer cell2  
Disconnecting net 'net1' from pin 'cell2/I'.  
Disconnecting net 'net2' from pin 'cell5/I'.  
Disconnecting net 'net2' from pin 'cell2/O'.  
Removing net 'net2' in design 'Top'.  
Connecting net 'net1' to pin 'cell5/I'.  
Removing cell 'cell2' in design 'Top'.  
1
```

SEE ALSO

```
all_fanout(2)  
insert_buffer(2)  
report_buffer_tree(2)  
report_cell(2)  
report_constraint(2)
```

remove_buffer_tree

Removes the buffer tree at a given driver pin.

SYNTAX

```
int remove_buffer_tree  
[-from pin_or_net_list]  
[-verbose]  
[-all]
```

Data Types

pin_or_net_list list

ARGUMENTS

-from *pin_or_net_list*
Specifies a list of driver pins or nets from which buffer trees are to be removed. This option is mutually exclusive with the **-all** option.

-verbose
Indicates that additional information is to be displayed during buffer removal.

-all
Removes all buffer trees from all nets. This option is mutually exclusive with the **-from** option. By default, this option is off.

DESCRIPTION

This command removes the buffer tree for each specified driver pin and for the driver pin of each specified net.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes a buffer tree driven by pin a/0.

```
prompt> remove_buffer_tree -from [get_pins a/0]
```

The following example forces all high fanout buffer trees in the design to be removed.

```
prompt> remove_buffer_tree -all
```

SEE ALSO

`create_buffer_tree(2)`
`legalize_placement(2)`
`report_buffer_tree(2)`
`report_buffer_tree_qor(2)`
`report_net_fanout(2)`
`set_cost_priority(2)`

remove_bus

Removes a port bus or net bus.

SYNTAX

```
status remove_bus  
object_list
```

Data Types

object_list list

ARGUMENTS

object_list
Specifies a list of port or net buses to remove.

DESCRIPTION

Removes port or net buses from a design. If port and net buses have the same name, both port buses and net buses are removed. Individual members of buses remain. The bus can be from the current design or one of its subdesigns. To delete a bus on a subdesign, it has to be unique.

Multicorner-Multimode Support

This command is not supported in a multi-scenario design flow.

EXAMPLES

The following example removes the specified port buses.

```
prompt> remove_bus {AB AC}
```

The following example removes the specified net buses.

```
prompt> remove_bus {BC BD}
```

In the following example, all buses with names that end in "s" are removed. If port and net buses have the same name, both buses are removed.

```
prompt> remove_bus *s
```

In the following example, all buses with names that start with "s" in the instance U1/U2 are removed. U1/U2 is a unique instantiation of design bot.

```
prompt> remove_bus U1/U2/s*
```


SEE ALSO

`create_net(2)`
`remove_net(2)`

remove_case_analysis

Removes the case analysis value from the specified input ports or pins.

SYNTAX

```
string remove_case_analysis  
port_or_pin_list | -all
```

Data Types

```
port_or_pin_list      list
```

ARGUMENTS

```
port_or_pin_list  
    Specifies ports or pins from which to remove the case analysis value.
```

```
-all  
    Specifies to remove case analysis for all ports or pins in the design where  
    the set_case_analysis command has previously been set.
```

DESCRIPTION

The **remove_case_analysis** command removes the case analysis value previously specified on ports or pins. Case analysis is specified using the **set_case_analysis** command. You must specify either **-all** or *port_or_pin_list*. The tool issues an error message if neither argument is specified.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example specifies that ports *in0* and *in2* are set to the constant logic value 0:

```
prompt> set_case_analysis 0 {in0 in2}  
prompt> report_case_analysis
```

```
*****  
Report : case_analysis  
Design : middle  
Version: 1997.01-development  
Date   : Mon Jul 22 08:13:50 1996  
*****
```

Pin name	Case analysis value

in2	0
in0	0

The following example removes the case analysis entry on port in2.

```
prompt> remove_case_analysis {in2}
prompt> report_case_analysis
```

```
*****
Report : case_analysis
Design : middle
Version: 1997.01-development
Date   : Mon Jul 22 08:14:16 1996
*****
```

Pin name	Case analysis value
-----	-----
in0	0

SEE ALSO

```
report_case_analysis(2)
set_case_analysis(2)
```

remove_cell

Removes cells from the current design.

SYNTAX

```
status remove_cell  
cell_list | -all  
[-freeze_silicon]
```

Data Types

cell_list list

ARGUMENTS

cell_list
 Specifies a list of cells to be removed from the current design. Each cell name must exist in the current design. You must specify either *cell_list* or **-all**.

-all
 Removes all cells in the current design.

-freeze_silicon
 Supports the ECO freeze silicon flow. The cell instances are not actually removed from the design. They are marked as spare cells.

DESCRIPTION

The **remove_cell** command removes cells or cell instances from the current design. The command also removes pins owned by the specified cells. The design or library cell to which the cell refers is not removed. If a cell instance is used, the parent design must be unique.

To create cells, use the **create_cell** command.

EXAMPLES

The following example uses **remove_cell** to remove the specified cells in the current design:

```
prompt> get_cells ""  
{U1 U2 U3 U4 U5 U6 U7 U8}  
  
prompt> remove_cell {U1 U2}  
Removing cell 'U1' in design 'example'.  
Removing cell 'U2' in design 'example'.  
  
prompt> get_cells ""
```

```
{U3 U4 U5 U6 U7 U8}
```

The following example removes all cells remaining in the current design:

```
prompt> remove_cell -all  
Removing cell 'U3' in design 'example'.  
Removing cell 'U4' in design 'example'.  
Removing cell 'U5' in design 'example'.  
Removing cell 'U6' in design 'example'.  
Removing cell 'U7' in design 'example'.  
Removing cell 'U8' in design 'example'.
```

```
prompt> get_cells ""  
{}
```

In the following example, the cell instances U1/foo1 and U1/foo2 are removed. The design MID must be unique.

```
prompt> remove_cell {U1/foo1 U1/foo2}  
Removing cell 'U1/foo1' in design 'MID'.  
Removing cell 'U1/foo2' in design 'MID'.
```

SEE ALSO

```
create_cell(2)  
current_design(2)  
map_freeze_silicon(2)  
place_freeze_silicon(2)
```

remove_cell_degradation

Removes the **cell_degradation** attribute on specified ports or designs.

SYNTAX

```
int remove_cell_degradation  
object_list
```

Data Types

object_list list

ARGUMENTS

object_list
Specifies a list of names of input ports for removing the **cell_degradation** attribute.

DESCRIPTION

Removes the **cell_degradation** attribute on the specified ports or designs.

To get information about optimization and design rule constraints, use **report_constraint**. To set the **cell_degradation** attribute, use **set_cell_degradationfp**.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes the cell degradation value on the port named input_1:

```
prompt> remove_cell_degradation input_1
```

SEE ALSO

```
set_cell_degradation(2)  
all_inputs(2)  
all_outputs(2)  
characterize(2)  
remove_attribute(2)  
report_constraint(2)  
set_max_capacitance(2)
```

remove_cell_sites

Deletes the specified cell sites.

SYNTAX

```
status remove_cell_sites  
-all | cell_sites
```

Data Types

cell_sites collection

ARGUMENTS

-all
Deletes all the cell sites in the current design. Mutually exclusive with the *cell_sites* argument.

cell_sites
Specifies the cell sites to be deleted. Mutually exclusive with the **-all** option.

DESCRIPTION

This command removes the specified cell sites.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes all cell sites from the current design.

```
prompt> remove_cell_sites -all  
1
```

SEE ALSO

`get_cell_sites(2)`

remove_cell_vt_type

removes voltage threshold type of a library cell or all cells of a library. Voltage threshold type is used for mixed voltage threshold filler cell insertion

SYNTAX

```
integer remove_cell_vt_type  
-library library_name | -lib_cell cell_name
```

Data Types

<i>library_name</i>	string
<i>cell_name</i>	string

ARGUMENTS

```
-library library_name | -lib_cell cell_name  
    Use -library to specify a library name or -lib_cell to specify a cell name.
```

DESCRIPTION

Use this command to remove vt type string of a library cell or cells of a library.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example will remove vt type setting from all cells of library RefLib1

```
prompt> remove_cell_vt_type -library "RefLib1" fP
```

SEE ALSO

```
insert_stdcell_filler(2)  
set_vt_filler_rule(2)  
set_cell_vt_type(2)  
report_cell_vt_type(2)
```


remove_checkpoint_designs

Remove checkpoint designs created by optimization commands

SYNTAX

```
int remove_checkpoint_designs  
[-command command]
```

Data Types

command string

ARGUMENTS

-command *command*
This option is either "place_opt", "clock_opt" or "route_opt" and causes the command to remove all checkpoint designs created by this command. By default, all checkpoint designs are removed.

DESCRIPTION

By enabling checkpointing using the set_checkpoint_strategy, optimization will save the design (snapshot) at periodic intervals during optimization. This allows the user to examine the intermediate results while optimization is still proceeding.

After all analysis has been performed, the user can use this command to remove the checkpoint designs created by optimization command so that the disk space is freed up. If the user has created designs that have names that clash with the checkpoint design names, these designs WILL NOT be removed. Only checkpoint designs created by optimization will be removed.

If a design is still being accessed, then this command will not remove the design.

EXAMPLES

```
prompt> remove_checkpoint_designs -command place_opt
```

will remove all checkpoint designs created by the place_opt command.

```
prompt> remove_checkpoint_designs
```

will remove all checkpoint designs.

SEE ALSO

```
place_opt(2)  
clock_opt(2)  
route_opt(2)  
set_checkpoint_strategy(2)
```

remove_clock

Removes clocks from the current design.

SYNTAX

```
int remove_clock  
clock_list | -all
```

Data Types

clock_list list

ARGUMENTS

```
clock_list | -all  
             Specifies which clocks to remove. If you specify -all, all clocks are removed.  
             Otherwise, the clocks in clock_list are removed.
```

DESCRIPTION

Removes the specified clock objects from the current design. You must specify either *clock_list* or **-all**.

If a clock to be removed is the only member of a path group, that path group is also removed. For information about path groups, refer to the **group_path** man page.

To list all clock sources in the design, use **report_clock**.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example removes clock "CLK1".

```
prompt> remove_clock CLK1
```

The following example removes all clocks from the current design.

```
prompt> remove_clock -all
```

SEE ALSO

```
create_clock(2)  
current_design(2)  
group_path(2)  
report_clock(2)  
reset_design(2)
```

remove_clock
1444

remove_clock_gates

Identifies and removes clock-gating cells according to the specified option.

SYNTAX

```
status remove_clock_gates  
[-gated_registers gated_register_list]  
[-min_bitwidth minsize_value]  
[-gating_cells clock_gating_cells_list]
```

Data Types

<i>gated_register_list</i>	object_list
<i>minsize_value</i>	int
<i>clock_gating_cells_list</i>	object_list

ARGUMENTS

-gated_registers *gated_register_list*
Specifies that all registers listed in *gated_register_list* will be nongated.

-min_bitwidth *minsize_value*
Specifies that all registers with size smaller than the *minsize_value* will be nongated.

-gating_cells *clock_gating_cells_list*
Specifies that all objects gated by clock-gating cells that are listed in *clock_gating_cells_list* will be nongated, and removes clock-gating cells from the design.

DESCRIPTION

The command identifies and removes clock-gating cells according to the specified option. It actually removes clock gating from objects (registers and clock-gating cells) performed by Power Compiler. The command provides a mechanism to remove clock gating selectively from various parts of the design.

This command also removes redundant clock-gating cells that do not gate any clock-gating cells. Any associated test observation logic is removed during optimization.

All registers that are nongated are remapped to new sequential cells, which might result in new pin names for the registers. A register can be gated by multiple clock gates in a multiple-stage clock-gating design. If the register is nongated, all clock gating on the clock path for this register is removed. However, if you specify to remove only one clock-gating cell, the remaining clock-gating cells will still exist on the clock path of the register.

Pin-based timing exceptions set on the pins of the old register might have been set by the **set_max_delay**, **set_min_delay**, **set_multicycle_path**, and **set_false_path** commands. These exceptions might not be properly transferred during the process if the new and old pin names do not match. The **remove_clock_gates** command issues a

warning message if there are pin-based timing exceptions on the register to be nongated.

EXAMPLES

The following example removes clock gating from all registers gated by the cell named `clk_gate_pipeline_reg_A` and removes the cell.

```
prompt> remove_clock_gates -gating_cell \  
clk_gate_pipeline_reg_A
```

The following example removes clock gating from the gated registers named `pipeline_reg_A[0]` and `pipeline_reg_A[2]`. If the clock-gating cell that gates these registers does not drive any other gating cells, it is also removed.

```
prompt> remove_clock_gates \  
-gated_registers {pipeline_reg_A[0] pipeline_reg_A[2]}
```

SEE ALSO

remove_clock_gating_check

Removes setup and hold checks from the specified clock gating cells.

SYNTAX

```
int remove_clock_gating_check  
[-setup]  
[-hold]  
[-rise]  
[-fall]  
object_list
```

Data Types

object_list list

ARGUMENTS

-setup
Removes setup checks from the specified gating cells.

-hold
Removes hold checks from the specified gating cells.

-rise
Removes the clock gating constraint on the rising delays. If you do not specify either **-rise** or **-fall**, the default is to remove constraints on both rising and falling delays.

-fall
Removes the clock gating constraint on the falling delays. If you do not specify either **-rise** or **-fall**, the default is to remove constraints on both rising and falling delays.
If you omit the setup and hold arguments, by default both types of checks are removed.

object_list
Specifies a list of designs, cells, pins, or clock objects from which to remove the clock gating checks.

DESCRIPTION

The **remove_clock_gating_check** command removes clock gating checks that were added using the **set_clock_gating_check** command.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example removes setup and hold checks from the *CHIP1* design:

```
prompt> remove_clock_gating_check CHIP1
```

The following example removes only the setup checks on the clock gating inputs of the *CLK_GATE1* cell in the current design:

```
prompt> remove_clock_gating_check -setup CLK_GATE1
```

The following example removes only the fall hold checks on the clock gating inputs of the *CLK_GATE2* cell in the current design:

```
prompt> remove_clock_gating_check -fall -hold CLK_GATE2
```

SEE ALSO

```
create_clock(2)  
current_design(2)  
report_clock(2)  
set_clock_gating_check(2)
```

remove_clock_groups

Removes specific exclusive or asynchronous clock groups from the current design.

SYNTAX

```
status remove_clock_groups  
-logically_exclusive  
    | -asynchronous  
    | -physically_exclusive  
name_list | -all
```

Data Types

name_list list

ARGUMENTS

- logically_exclusive**
Specifies that groups set for logically exclusive clocks are to be removed. The **-logically_exclusive**, **-physically_exclusive** and **-asynchronous** options are mutually exclusive.
- asynchronous**
Specifies that groups set for asynchronous clocks are to be removed. The **-exclusive** and **-asynchronous** options are mutually exclusive.
- physically_exclusive**
Specifies that groups set for physically exclusive clocks are to be removed. The **-logically_exclusive**, **-physically_exclusive** and **-asynchronous** options are mutually exclusive.
- name_list*
Specifies a list of clock groups to be removed, which matches the groups in the given names. You should use the **set_clock_groups** command to predefine these names. Substitute the list you want for *name_list*. The *name_list* argument and **-all** option are mutually exclusive.
- all**
Specifies to remove all groups set for exclusive or asynchronous clocks in the current design. The *name_list* argument and **-all** option are mutually exclusive.

DESCRIPTION

Removes specific exclusive or asynchronous clock groups from the current design. These clock groups are specified by the *name_list* from the current design. You must specify either **-logically_exclusive**, **-physically_exclusive** or **-asynchronous**. You must specify either *name_list* or **-all**.

To find the names for existing groups, use the **report_clock** command with the **-groups** option.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example removes logically exclusive clock groups named mux.

```
prompt> remove_clock_groups -logically_exclusive mux
```

The following example removes all asynchronous clock groups from the current design.

```
prompt> remove_clock_groups -asynchronous -all
```

SEE ALSO

```
report_clock(2)  
set_clock_groups(2)
```


remove_clock_latency

Removes clock latency information from the specified objects.

SYNTAX

```
string remove_clock_latency  
[-fall]  
[-min]  
[-max]  
[-source]  
[-early]  
[-late]  
object_list  
[-rise]
```

Data Types

object_list list

ARGUMENTS

- fall
Specifies that the fall clock latency should be removed. By default, this is removed for both minimum and maximum clock latency.
- min
Specifies that the minimum clock latency should be removed. By default, this is removed for both rise and fall clock latency.
- max
Specifies that the maximum clock latency should be removed. By default, this is removed for both rise and fall clock latency.
- source
Specifies that clock source latency should be removed.
- early
Specifies that the late clock source latency should be removed. If this option is not specified, both early and late clock latencies are removed.
- late
Specifies that the late clock source latency should be removed. If this option is not specified, both early and late clock latencies are removed.
- object_list*
Provides a list of clocks, ports, or pins.
- rise
Specifies that the rise clock latency should be removed. By default, this is removed for both minimum and maximum clock latency.

DESCRIPTION

The **remove_clock_latency** command removes user-specified clock network or source latency information from specified objects. Clock network latency is the time it takes for a clock signal on the design to propagate from the clock definition point to a register clock pin. Clock source latency (also called insertion delay) is the time it takes for a clock signal to propagate from its actual ideal waveform origin point to the clock definition point in the design. Clock network and source latency information is set on objects using the **set_clock_latency** command. See the **set_clock_latency** man page for more information.

You can remove selected portions of the clock latency by specifying various such as **-rise**, **-fall**, **-min**, **-max**, **-late**, and **-early**.

To report clock network and source latency information, use the **report_clock** command with the **-skew** option.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example removes clock latency information from clock CLK1:

```
prompt> remove_clock_latency [get_clocks CLK1]
```

The following example removes clock source latency information from clock CLK1:

```
prompt> remove_clock_latency -source [get_clocks CLK1]
```

The following example removes only rise clock source latency information from clock CLK1:

```
prompt> remove_clock_latency -rise -source [get_clocks CLK1]
```

SEE ALSO

```
create_clock(2)
remove_attribute(2)
remove_clock(2)
remove_clock_uncertainty(2)
report_clock(2)
reset_design(2)
set_clock_latency(2)
set_clock_uncertainty(2)
set_input_delay(2)
```

remove_clock_mesh

Removes the premesh tree, clock mesh or the postmesh tree routes and/or logical connections based on the options specified.

SYNTAX

```
status remove_clock_mesh  
[-premesh]  
[-clockmesh]  
[-postmesh]  
[-route_only]  
[-clock_tree ]
```

ARGUMENTS

-premesh
Specifies the command to delete only the premesh part of the mesh tree.

-clockmesh
Specifies that the clockmesh needs to be removed.

-postmesh
Specifies the command to remove the post mesh tree.

-route_only
Specifies the command to remove only the routes and not touch the logical connectivity.

DESCRIPTION

The operation for various options specified are : For premesh : By default the pre-mesh tree including the last level mesh drivers will be removed along with pre-mesh routes. However, if the premesh option is given alongwith the other option - **route_only**, the logical pre-mesh tree will not be deleted but the pre-mesh routes till the input pins of the last level mesh drivers will be deleted. This **-route_only** may be useful when user wants to experiment with customized pre-mesh routes (like H/T/I) and with general routes We may have 2 conditions : 1) If there is no isolation buffer in the clock-mesh network, premesh tree means the tree starting from the clock root. The command **remove_clock_mesh** will remove the premesh tree starting from the root. 2) If there is an isolation buffer that implies that **adjust_premesh_connection** could separate out the mesh part from the macro pins. Also we need to be aware that **remove_clock_mesh** could be called after doing the whole flow which means optimization could have been called and which might have added buffers between the root and the isolation buffer and in the paths of those premesh macros. It is reasonable for **remove_clock_mesh** to do the following :- a) Remove premesh tree starting from the output of isolation buffer to the mesh b) Remove the buffers between the root and the isolation buffer's input c) Don't remove any logical objects on the macro paths and don't remove isolation buffer d) However, the routes, if any, will be deleted from the whole pre-mesh tree. (i.e., the routes of pre-isolation-buffer part, post isolation buffer part and the routes on the premesh macro paths). For clockmesh: If specified then all the routes (mesh straps and comb-

routes, if any) of the mesh net will be deleted. This implies the following things :- a) After clock-mesh deletion, the logical mesh net will remain multi-driven net unless otherwise the pre-mesh was already deleted. b) When user will form another set of straps (perhaps with different pitches etc.) after doing the clock-mesh deletion, **route_mesh_net** command must be run to make it completely routed

For postmesh: If this command is invoked with **-postmesh**, the command will expect that the pre-mesh tree and the mesh have already been deleted. In any case, this command will be just deleting the routes, if any, between the ICG's and the flops. If the premesh and mesh have not been deleted already then the command will error out with a suitable message.

If this command is issued with no options, it will delete: (a) clock-mesh routes, (b) pre-mesh with routes, (c) post-mesh parts with routes in order.

In case the **-route_only** option is given then the command will give proper message saying that the intended usage is only for custom mesh structures.

EXAMPLE

The following command removes the clock mesh tree :

```
prompt> remove_clock_mesh -clock_tree clk1
```

SEE ALSO

```
create_clock_mesh(2)  
adjust_premesh_connection(2)  
add_clock_drivers(2)  
compile_premesh_tree(2)
```

remove_clock_sense

Removes clock sense information from the specified pins.

SYNTAX

```
integer remove_clock_sense  
[-all]  
[-clocks clock_list]  
pins
```

Data Types

<i>clock_list</i>	list
<i>pins</i>	list

ARGUMENTS

-all

Removes all clock sense information in the current design. By default, this option is off.

-clocks *clock_list*

Specifies the list of clocks for which the clock sense information is removed. This option can remove only clock sense information that has been set by using the **set_clock_sense** command with the **-clocks** option. It does not remove clock sense settings from pins. By default, the clock sense information is removed for all clocks passing through the specified pins.

pins

Specifies the pins from which to remove clock sense information.

DESCRIPTION

This command removes user-specified clock sense (unateness) information from specified pins and clocks. To set clock sense information, use the **set_clock_sense** command.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example removes positive unateness defined on a pin named *XOR/Z* with respect to clock *CLK1*, but does not remove the negative unateness.

```
prompt> set_clock_sense -positive -clocks [get_clock CLK1] XOR/Z  
prompt> set_clock_sense -negative XOR/Z
```

```
prompt> remove_clock_sense -clocks [get_clocks CLK1] XOR/Z
```

The following example removes all unateness information in the current design.

```
prompt> remove_clock_sense -all
```

SEE ALSO

```
set_clock_sense(2)
```

remove_clock_transition

Removes clock transition attributes on the specified clock objects.

SYNTAX

```
int remove_clock_transition  
clock_list
```

ARGUMENTS

```
clock_list  
    Specifies on which clocks the transition attributes must be removed.
```

DESCRIPTION

Removes the clock transition attributes on the specified clocks.

To list all clock transition values which have been set, use **report_clock -skew**.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example removes transition attributes on clock CLK.

```
prompt> remove_clock_transition CLK
```

The following example removes transition attributes on all clocks in the current design.

```
prompt> remove_clock_transition all_clocks()
```

SEE ALSO

```
create_clock(2)  
current_design(2)  
report_clock(2)  
reset_design(2)  
set_clock_transition(2)
```

remove_clock_tree

Removes buffers and inverters from the clock tree.

SYNTAX

```
status remove_clock_tree  
[-clock_trees name_or_source_pin_list]  
[-honor_dont_touch]  
[-synopsys_only]  
[-high_fanout_net net_or_pin_list]
```

Data Types

<i>name_or_source_pin_list</i>	list
<i>net_or_pin_list</i>	list

ARGUMENTS

-clock_trees *name_or_source_pin_list*
Removes only those clock trees whose names or root pins are listed in *name_or_source_pin_list*. Each element of *name_or_source_pin_list* must be either a clock tree name or a clock root (port or pin) shown in **report_clock**. By default, the command is applied to all currently defined clock trees.

-honor_dont_touch
Keeps buffers and inverters that have the **dont_touch** attribute. By default, this option is off.

-synopsys_only
Removes buffering for only those clock trees inserted by using the **compile_clock_tree** command. By default, this option is off.

-high_fanout_net *net_or_pin_list*
Removes a buffer tree for high fanout nets that were synthesized using **compile_clock_tree -high_fanout_net**. The nets or nets' driving pin must be specified in the *net_or_pin_list*.

DESCRIPTION

This command removes clock tree buffering and updates the design database with the results of the updated clock trees. The command removes only the clock tree buffering; the clock tree definition is maintained. You cannot use this command on a design with clock mesh nets.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example reports an already inserted clock tree named CLK1, then removes all preexisting buffering.

```
prompt> report_clock_tree -clock_trees CLK1  
prompt> remove_clock_tree -clock_trees CLK1
```

SEE ALSO

```
compile_clock_tree(2)  
report_clock_tree(2)
```

remove_clock_tree_exceptions

Removes the specified clock tree exceptions, which were previously set with the **set_clock_tree_exceptions** command.

SYNTAX

```
status remove_clock_tree_exceptions
[-all]
[-float_pins float_pin_collection]
[-stop_pins stop_pin_collection]
[-non_stop_pins non_stop_pin_collection]
[-exclude_pins exclude_pin_collection]
[-dont_touch_subtrees dots_pin_collection]
[-dont_buffer_nets collection_or_list_of_nets]
[-dont_size_cells collection_or_list_of_cells]
[-size_only_cells collection_or_list_of_cells]
[-float_pin_logic_level]
[-preserve_hierarchy collection_of_pin_or_cells]
[-clocks object_list]
```

Data Types

<i>float_pin_collection</i>	collection
<i>stop_pin_collection</i>	collection
<i>non_stop_pin_collection</i>	collection
<i>exclude_pin_collection</i>	collection
<i>dots_pin_collection</i>	collection
<i>collection_or_list_of_nets</i>	collection
<i>collection_or_list_of_cells</i>	collection
<i>collection_of_pin_or_cells</i>	collection

ARGUMENTS

-all

Remove all the clock exceptions in the design.

-float_pins *float_pin_collection*

Specifies a list or collection of predefined float pins. When used with the **-float_pin_logic_level** option, only the logic-level value for the specified pins is reset to 0. Otherwise, both the phase delay and logic-level values for the specified pins are removed.

-stop_pins *stop_pin_collection*

Specifies the pins from which to remove the stop pin exception.

-non_stop_pins *non_stop_pin_collection*

Specifies the pins from which to remove the nonstop pin exception.

-exclude_pins *exclude_pin_collection*

Specifies the pins from which to remove the exclude exception.

remove_clock_tree_exceptions

`-dont_touch_subtrees dts_pin_collection`
 Specifies the pins from which to remove the dont_touch_subtree exception.

`-dont_buffer_nets collection_or_list_of_nets`
 Specifies the nets from which to remove the dont_buffer attribute.

`-dont_size_cells collection_or_list_of_cells`
 Specifies the cells from which to remove the dont_size attribute.

`-size_only_cells collection_or_list_of_cells`
 Specifies the cells from with to remove the size_only attribute.

`-float_pin_logic_level`
 Removes the logic-level value on the specified float pins. When used with the **-float_pins** option, the logic level on the specified pins is removed. Otherwise, the logic-level value is removed from all the pins in the design.

`-preserve_hierarchy collection_of_pin_or_cells`
 Specifies the pins or cells from with to remove the hierarchy preservation exception.

DESCRIPTION

This command removes the specified clock tree exceptions, which were previously set with the **set_clock_tree_exceptions**. You can remove the following exceptions: float pins (phase delay, logic level, or both), stop pins, exclude pins, dont_touch_subtree pins, dont_buffer nets, dont_size cells, and size_only cells.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example defines a float pin and then removes the definition.

```
prompt> set_clock_tree_exceptions -float_pins [get_pins RAM1/WE]
prompt> remove_clock_tree_exceptions -float_pins [get_pins RAM1/WE]
```

The following example defines a stop pin and then removes the definition.

```
prompt> set_clock_tree_exceptions -stop_pins [get_pins AND/B]
prompt> remove_clock_tree_exceptions -stop_pins [get_pins AND/B]
```

The following example removes all float pin logic-level exceptions:

```
prompt> remove_clock_tree_exceptions -float_pin_logic_level
```

The following example removes the float pin logic-level exception on the specified float pins:

```
prompt> remove_clock_tree_exceptions -float_pin_logic_level \
```

-float_pins {BUF1/A BUF2/A BUF3/A}

The following example defines an exclude pin and then removes the definition.

```
prompt> set_clock_tree_exceptions -exclude_pins [get_pins Reg1/CLK]  
prompt> remove_clock_tree_exceptions -exclude_pins [get_pins Reg1/CLK]
```

The following example defines an cell for which boundary needs to be preserved and then removes the definition.

```
prompt> set_clock_tree_exceptions -preserve_hierarchy [get_cells Reg1]  
prompt> remove_clock_tree_exceptions -preserve_hierarchy [get_cells Reg1]
```

SEE ALSO

```
set_clock_tree_exceptions(2)  
set_clock_tree_options(2)  
compile_clock_tree(2)
```

remove_clock_tree_options

Removes clock tree objects from database.

IMPORTANT NOTICE for IC Compiler

Note: This command is currently NOT SUPPORTED; Use the `reset_clock_tree_options` command instead. Ignore the rest of the document until this notice is removed.

SYNTAX

```
void remove_clock_tree_options  
[-clock_trees name_or_source_pin_list]
```

Data Types

name_or_source_pin_list list

ARGUMENTS

`-clock_trees name_or_source_pin_list`

Removes only those clock trees whose names or root pins are listed in *name_or_source_pin_list*. Each element of *name_or_source_pin_list* can be either a user-assigned symbolic name or the name of the source (port or pin) of each clock tree. It must match one of the following:

- The *clock_tree_name* value set in the **-clock_tree_name** option of the **set_clock_tree_options** command used when creating the clock tree.
 - The default clock tree name that is the full name of the clock tree root pin or port passed to the **set_clock_tree_options** command by specifying the *one_pin_col_spec* value of the **-root** option.
- If you do not use the **remove_clock_tree_options** command with the **-clock_trees** option, the command is applied to all currently-defined clock-trees.

DESCRIPTION

This command deletes the database entry for the clock tree object predefined by the **set_clock_tree_options** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

This example creates and modifies a new clock tree structure named CLK1. The library units for the technology are in pico-seconds and femto-farads. It creates a clock tree with a source pin named U1/Z and then removes it.

```
prompt> set_clock_tree_options -clock_tree_name CLK1\  
-root [get_pins U1/Z]  
prompt> remove_clock_tree_options -clock_trees CLK1
```

SEE ALSO

set_clock_tree_options(2)

remove_clock_uncertainty

Removes clock uncertainty information previously set by the **set_clock_uncertainty** command.

SYNTAX

```
string remove_clock_uncertainty
[object_list
 | -from from_clock
 | -rise_from rise_from_clock
 | -fall_from fall_from_clock
-to to_clock
 | -rise_to rise_to_clock
 | -fall_to fall_to_clock]
[-rise]
[-fall]
[-setup]
[-hold]
```

Data Types

<i>object_list</i>	list
<i>from_clock</i>	list
<i>rise_from_clock</i>	list
<i>fall_from_clock</i>	list
<i>to_clock</i>	list
<i>rise_to_clock</i>	list
<i>fall_to_clock</i>	list

ARGUMENTS

object_list
Specifies a list of clocks, ports, or pins from which uncertainty information is to be removed. By default, uncertainty information is removed from all objects in the current design. Use either the pair of **-from/-rise_from/-fall_from** and **-to/-rise_to/-fall_to** options or the *object_list* option. Do not specify both, because they are mutually exclusive.

-from *from_clock*
Specifies the source and destination clocks for interclock uncertainty. Specify either the pair of **-from/-rise_from/-fall_from** and **-to/-rise_to/-fall_to**, or *object_list*. Do not specify both.

-rise_from *rise_from_clock*
Specifies the source and destination clocks, but indicates that *uncertainty* applies only to the rising edge of the source clock. Use only one of the **from**, **-rise_from**, or **-fall_from** options.

-fall_from *fall_from_clock*
Specifies the source and destination clocks, but indicates that *uncertainty* applies only to the falling edge of the source clock. Use only one of the **from**, **-rise_from**, or **-fall_from** options.

-to to_clock
 Specifies the destination clocks for interclock uncertainty. Specify either the pair of **-from/-rise_from/-fall_from** and **-to/-rise_to/-fall_to**, or *object_list*. Do not specify both.

-rise_to rise_to_clock
 Specifies the destination clocks for interclock uncertainty, but indicates that *uncertainty* applies only to the rising edge of the destination clock. Use only one of the **-to**, **-rise_to**, or **-fall_to** options.

-fall_to fall_to_clock
 Specifies the destination clocks for interclock uncertainty, but indicates that *uncertainty* applies only to the falling edge of the destination clock. Use only one of the **-to**, **-rise_to**, or **-fall_to** options.

-rise
 Specifies that uncertainty is to be removed for only the rising clock edge. By default, uncertainty is removed for both rising and falling clock edges. This option is valid only for interclock uncertainty, and is now obsolete. Unless you need this option for backward-compatibility, use **-rise_to** instead.

-fall
 Specifies that uncertainty is to be removed for only the falling clock edge. By default, uncertainty is removed for both rising and falling clock edges. This option is valid only for interclock uncertainty, and is now obsolete. Unless you need this option for backward-compatibility, use **-fall_to** instead.

-setup
 Specifies that only setup check uncertainty is to be removed. By default, both setup and hold check uncertainties are removed.

-hold
 Specifies that only hold check uncertainty is to be removed. By default, both setup and hold check uncertainties are removed.

DESCRIPTION

Removes clock uncertainty information previously set by the **set_clock_uncertainty** command from clocks, ports, or pin or between specified clocks. If the command is issued without options, all clock uncertainty information is removed from the current design. To display clock uncertainty information, use the **report_clock** command with the **-skew** option.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example removes uncertainty information from a clock named *CLK* and a pin named *clk_buf/Z*.


```
prompt> remove_clock_uncertainty [get_clocks CLK]
prompt> remove_clock_uncertainty [get_pins clk_buf/Z]
```

The following example removes interclock uncertainties between the *PHI1* and *PHI2* clock domains:

```
prompt> remove_clock_uncertainty -from PHI1 -to PHI1
prompt> remove_clock_uncertainty -from PHI2 -to PHI2
prompt> remove_clock_uncertainty -from PHI1 -to PHI2
prompt> remove_clock_uncertainty -from PHI2 -to PHI1
```

SEE ALSO

```
all_clocks(2)
create_clock(2)
report_clock(2)
set_clock_latency(2)
set_clock_transition(2)
set_clock_uncertainty(2)
```

remove_congestion_options

Removes congestion options from the current design.

SYNTAX

```
int remove_congestion_options  
[-all] id_list
```

Data Types

id_list list

ARGUMENTS

-all
 Specifies to remove all congestion options from the current design.

id_list
 Indicates to remove congestion options with the specified ids.

DESCRIPTION

The **remove_congestion_options** command removes the congestion options from the current design. If the **-all** option is specified, all congestion options are removed from the design. If a list of ids is provided, the congestion options with the specified ids are removed. The id is displayed every time a regional congestion option is created. Or, it can be viewed by using the **report_congestion_options** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes all congestion options.

```
prompt> remove_congestion_options -all
```

SEE ALSO

```
report_congestion(2)  
report_congestion_options(2)  
set_congestion_options(2)
```

remove_cts_scenario

Removes the current CTS scenario setting. This command doesn't remove the scenario itself, it only removes the CTS scenario setting.

SYNTAX

Boolean **remove_cts_scenario**

ARGUMENTS

The **remove_cts_scenario** command has no arguments.

DESCRIPTION

Removes the CTS scenario setting on currently defined cts scenario. in MCMM flow, `compile_clock_tree`, `optimize_clock_tree` and `balance_inter_clock_delay` will work on the CTS scenario (if defined), or on current scenario if the CTS scenario is not defined. This command always returns a "true" value.

Multicorner-Multimode Support

This command uses information from the clock tree synthesis scenario.

EXAMPLES

The following example uses **remove_cts_scenario** to remove cts scenario.

```
prompt> create_scenario MODE1
prompt> set_cts_scenario MODE1
MODE1
prompt> get_cts_scenario
MODE1
prompt> remove_cts_scenario
1
prompt> get_cts_scenario

prompt>
```

SEE ALSO

```
balance_inter_clock_delay(2)
clock_opt(2)
compile_clock_tree(2)
get_cts_scenario(2)
optimize_clock_tree(2)
set_cts_scenario(2)
```

remove_dangling_wires

Removes dangling wires and vias on the specified nets.

SYNTAX

```
status remove_dangling_wires  
-nets {collection_of_nets}  
[-route_types {list_of_route_types}]  
[-layers {collection_of_layers}]  
[-no_rerun_drc]
```

Data Types

<i>collection_of_nets</i>	collection
<i>list_of_route_types</i>	list
<i>collection_of_layers</i>	collection

ARGUMENTS

-nets {*collection_of_nets*}

Specifies the target nets from which to remove dangling wires or vias. This is a required option.

-route_types {*list_of_route_types*}

Specifies the target route types of the nets from which to remove dangling wires or vias. You can specify a list of one or more route types. If you do not specify this option, all route types are considered. The valid values for route type are **signal_route_detail**, **signal_user_enter**, **clk_ring**, **clk_strap**, **clk_tie_off**, **clk_user_enter**, **pg_ring**, **pg_strap**, **pg_tie_off**, **pg_user_enter**, **pg_std_cell_pin_conn**, **pg_macro_io_pin_conn**, **bus**. These route types are defined as follows:

- **signal_route_detail** is for detail routed signal nets
- **signal_user_enter** is for user-entered signal signal nets
- **clk_ring** is for clock ring nets
- **clk_strap** is for clock strap nets
- **clk_tie_off** is for detail routed clock tie-off nets
- **clk_user_enter** is for user-entered clock nets
- **pg_ring** is for power or ground ring nets
- **pg_strap** is for power or ground strap nets
- **pg_tie_off** is for detail routed power or ground tie-off nets
- **pg_user_enter** is for user-entered power or ground nets

- **pg_std_cell_pin_conn** is for standard cell power or ground connections
- **pg_macro_io_pin_conn** is for power or ground pin connections on macro cells or I/O cells
- **bus** is for bus nets, whether the net is a signal, clock, power, or ground net. Because of a router limitation, the **-route_types** option does not support all of the Milkyway route types. The following Milkyway route types are not supported by this option: **signal_route_global**, **clk_zero_skew_route**, **fill_track**, **shield_fixed**, and **shield_dynamic**.

-layers {collection_of_layers}

Specifies the target layers from which to remove the dangling wires and vias. If you do not specify this option, the dangling wires and vias are removed from all layers.

-no_rerun_drc

Suppresses the running of design rule checking (the **verify_route** command) before starting the removal of dangling wires and vias. By default, the command runs design rule checking. The design rule checking does not involve preroutes.

DESCRIPTION

This command removes dangling wires and vias on the specified nets without changing the net topologies and connections. Only antenna (dangling) wires and vias are trimmed, and the command does not touch terminals. A wire or via is considered dangling if it has no connection to either an upper or lower via or metal.

In some cases, DRC violations, such as minimum area violations, might be introduced after removing dangling wires and vias on clock, power, and ground nets. This command does not fix these DRC violations.

If you specify the **bus** type in the **-route_types** option, this command eliminates dangling wires and vias on all nets with a route type of **bus**, regardless of whether the net is a signal, clock, power, or ground net.

If the specified nets are preroutes, you must use the **-no_rerun_drc** option because the design rule checking embedded in this command is for detail routing.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes dangling wires and vias from the ck1 clock net. Only dangling wires with a route type of **clk_user_enter** on the metal6 and metal7 layers and floating vias on the via6 layer are removed. It uses the previous design rule checking results and does not rerun design rule checking.

```
prompt> remove_dangling_wires -nets {ck1} \
```

```
-route_types {clk_user_enter} -layers {metal6 metal7 via6} \  
-no_rerun_drc
```

The following example removes dangling wires and vias for a physical bus named B_BUS with the **bus** route type. It uses the previous design rule checking results and does not rerun design rule checking.

```
prompt> remove_dangling_wires \  
-nets [get_nets -of_objects [get_physical_buses B_BUS]] \  
-route_types {bus} -no_rerun_drc
```

SEE ALSO

verify_route(2)

remove_data_check

Removes specified data-to-data checks previously set by set_data_check.

SYNTAX

```
string remove_data_check  
-from from_object  
    | -rise_from from_object  
    | -fall_from from_object  
-to to_object  
    | -rise_to to_object  
    | -fall_to to_object  
[-setup | -hold]  
[-clock clock]
```

Data Types

<i>from_object</i>	collection
<i>to_object</i>	collection

ARGUMENTS

- from** *from_object*
Specifies a pin or port in the current design as the related pin of the data-to-data check to be removed. Both rising and falling checks are removed. You must specify one of **-from**, **-rise_from**, or **-fall_from**.
- rise_from** *from_object*
Similar to the **-from** option, but applies to only rising delays at the related pin. You must specify one of **-from**, **-rise_from**, or **-fall_from**.
- fall_from** *from_object*
Similar to the **-from** option, but applies to only falling delays at the related pin. You must specify one of **-from**, **-rise_from**, or **-fall_from**.
- to** *to_object*
Specifies a pin or port in the current design as the constrained pin of the data-to-data check to be created. Both rising and falling checks are removed. You must specify one of **-to**, **-rise_to**, or **-fall_to**.
- rise_to** *to_object*
Similar to the **-to** option, but applies to only rising delays at the constrained pin. You must specify one of **-to**, **-rise_to**, or **-fall_to**.
- fall_to** *to_object*
Similar to the **-to** option, but applies to only falling delays at the constrained pin. You must specify one of **-to**, **-rise_to**, or **-fall_to**.
- setup**
Indicates that only the setup data check is to be removed. If neither **-setup** nor **-hold** is specified, both setup and hold checks are removed.

-hold

Indicates that only the hold data check is to be removed. If neither **-setup** nor **-hold** is specified, both setup and hold checks are removed.

-clock *clock*

Indicates that the data check for the specified clock at the related pin is to be removed. This option applies only if a previous **set_data_check** command used the **-clock** option to specify the same clock; otherwise, this option is ignored.

DESCRIPTION

The **remove_data_check** command specifies data-to-data check(s) to be removed between the from object and to object for the specified options. These checks were previously set using the **set_data_check** command.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example removes a data-to-data check from and1/B to and1/A with respect to the rising edge of signals at and1/B. Both setup and hold checks are removed.

```
prompt> remove_data_check -rise_from and1/B -to and1/A
```

The following example removes setup data-to-data check between and1/B to and1/A with respect to the rising edge of signals at and1/B, coming from startpoints triggered with clock CK1, falling edge of signals at and1/A.

```
prompt> remove_data_check -rise_from and1/B \  
-fall_to and1/A -setup -clock [get_clock CK1]
```

SEE ALSO

`report_timing(2)`
`set_data_check(2)`
`update_timing(2)`
`timing_enable_multiple_clocks_per_reg(3)`

remove_design

Removes a list of designs or libraries from memory.

SYNTAX

```
status remove_design  
[design_list | -designs | -all]  
[-hierarchy] [-quiet]
```

Data Types

design_list list

ARGUMENTS

- design_list*
Specifies a list of designs or libraries to removed. The default is the current design. Cannot be used with **-designs** or **-all**. The design to remove is either with an absolute path or without. To know the absolute path of designs, use the list **-files**.
- designs**
Indicates that all designs in memory are to be removed. Cannot be used with *design_list* or **-all**.
- all**
Indicates to remove all designs and libraries in memory. Cannot be used with *design_list* or **-designs**.
- hierarchy**
Indicates to remove all designs within the hierarchy of the designs in the *design_list* or in the current design. The default is not to remove any subdesigns.
- quiet**
Turns off verbose mode that prints out messages showing the names of the designs or libraries that are being removed.

DESCRIPTION

Removes a list of designs or libraries from memory. If you do not specify any arguments, the current design is removed. When a design or library is removed, the memory it occupies is freed.

NOTE: Memory is freed to be reused by this same process. However, memory is not returned to the operating system until you exit the tool.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes the current design.

```
prompt> current_design TEST  
Current design is now 'TEST'
```

```
prompt> remove_design  
Removing design 'TEST'
```

The following example removes a specified design and all subdesigns in the hierarchy.

```
prompt> remove_design -hier TEST2  
  
Removing design 'TEST2'  
Removing design 'ADDER'  
Removing design 'MUX8'  
Removing design 'ND17'
```

The following example removes a specified design RIGHT in file TOP.ddc:

```
prompt> remove_design {"/remote/qe5/bill/designs/TOP.ddc:RIGHT"}  
  
Removing design 'RIGHT'
```

The following example removes all design and library objects from memory.

```
prompt> remove_design -all  
Removing design 'A'  
Removing design 'B'  
Removing library 'test_lib'
```

SEE ALSO

`current_design(2)`

remove_diode

deletes the diode cells

SYNTAX

```
integer remove_diode  
[-cells diode_cell_name]  
-nets collection_of_nets  
[-all_clock_net]  
[-dangling_wires]
```

Data Types

<i>diode_cell_name</i>	list
<i>collection_of_nets</i>	list

ARGUMENTS

-cells *diode_cell_name*
Specify one or more diode cells to delete. Default : Search the design library and all its reference libraries for all the diode cells to delete. A diode cell is a cell that has at least one diode port.

-nets *collection_of_nets*
specify either one net or a pattern to match the names of multiple nets of the diode ports to be disconnected

-all_clock_net
disconnect all diode ports on all clock nets. If this option is selected, then the specified nets will be ignored.

-dangling_wires
After deleting the diode cell instances, the wires that were connected to the diode ports, became dangling. This can be cleaned up by setting this option. Note that it removes all dangling wires, even those that existed before running this command.

DESCRIPTION

Prerequisites : The cell must be opened with write permission.

Helps you delete the diode cells inserted for fixing antenna violations. It deletes the diode cell given by the name of the diode cell and the net names. It first disconnects all the diode ports on all the given nets. Then it delete the diode cells that are fully disconnected (that is, no connected port (except power and ground) and no connected diode port). The exact behavior of the command is given in the steps below:

+ Disconnect all the diode port instances in all the cell instances of the given diode cells on all the given nets.

- + Delete the cell instances that are fully disconnected of the given diode cells. A fully disconnected cell instance contains no connected port (except power and ground) and no connected diode port.
- + Delete all dangling wires, if the option is set.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example delete diode cells M1B2H and M1A3K, and disconnect net NET_INV05 from cells. Also, remove any possible dangling wires after clear up cells and nets.

```
prompt> remove_diode -cells {M1B2H M1A3K} -nets NET_INV05 -dangling_wires
```

SEE ALSO

remove_disable_clock_gating_check

For specified cells and pins, restores clock gating checks previously disabled by the `set_disable_clock_gating_check` command.

SYNTAX

```
string remove_disable_clock_gating_check  
object_list
```

Data Types

object_list list

ARGUMENTS

object_list
Specifies a list of cells and pins for which previously-disabled clock gating checks are to be restored.

DESCRIPTION

For specified cells and pins, restores clock gating checks previously disabled by the `set_disable_clock_gating_check` command.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example restores disabled clock gating checks for the object named *an1*.

```
prompt> remove_disable_clock_gating_check an1/A
```

The following example restores all disabled gating checks on the cells named *U1* or *U2*.

```
prompt> remove_disable_clock_gating_check U1/or2
```

SEE ALSO

`set_disable_clock_gating_check(2)`

remove_disable_timing

Enable previously user-disabled timing arcs in the current design. It is an equivalent to `set_disable_timing -restore`.

SYNTAX

```
int remove_disable_timing  
object_list  
[-from from_pin_name -to to_pin_name] [-all_loop_breaking]
```

Data Types

<i>object_list</i>	list
<i>from_pin_name</i>	string
<i>to_pin_name</i>	string

ARGUMENTS

object_list
Specifies a list of cells, pins, ports, library pins, or library cells that are to be enabled. The cells or the cells of pins are no longer size only if size only were not set by the user.

-from *from_pin_name*
Specifies that arcs only from this pin on the specified cell are to be enabled. The *object_list* must contain only cells if **-from** and **-to** are specified.

-to *to_pin_name*
Specifies that arcs only to this pin on the specified cell are to be enabled. The *object_list* must contain only cells if **-from** and **-to** are specified.

-all_loop_breaking
Re-enable all the stored loop-breaking timing arcs. Only applies to the entire design.

DESCRIPTION

The **remove_disable_timing** command enables timing through the specified cells, pins, or ports in the current design.

Any cell, pin, or port in the current design or its subdesigns can be disabled and thus re-enabled. Cells at lower levels in the design hierarchy are specified as *instance1/instance2/cell_name*. Pins at lower levels in the design hierarchy are specified as *instance1/instance2/cell_name/pin_name*. Ports at lower levels in the design hierarchy are specified as *instance1/instance2/cell_name/port_name*.

Library pins and library cells can also be disabled and re-enabled.

Note: For subdesigns in the hierarchy, you must specify instance names instead of design names.

In disabling, when the timing through a cell is disabled, only those cell arcs that lead to the output pins of the cell are disabled. When the timing through a pin or port is disabled, only those cell arcs that lead to or from that pin or port are disabled. The `remove_disable_timing` command has a reverse effect.

When the **-from** and **-to** pins are specified, all arcs between these pins on the cell are enabled.

Use **report_lib -timing_arcs** to list the timing arcs for a library cell, and **report_design** to list the disabled arcs in the current design so as to check the enabling effect of the `remove_disable_timing` command.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

In the following example, all timing arcs between pins 'A' and 'Z' on cell U1/U1 in the current design are enabled.

```
prompt> remove_disable_timing U1/U1 -from A -to Z
```

In the following example, all timing arcs between pins 'D' and 'Q' on cell FD1 in the library 'my_lib' are enabled. This command enables arcs on a library cell; therefore, all instances of that library cell are affected in any design linked to the same library.

```
prompt> remove_disable_timing my_lib/FD1 -from D -to Q
```

SEE ALSO

```
current_design(2)  
remove_attribute(2)  
report_lib(2)  
reset_design(2)  
set_disable_timing(2)
```

remove_distributed_hosts

Remove all the hosts added to the distributed hosts list.

SYNTAX

```
int remove_distributed_hosts
```

ARGUMENTS

The **remove_distributed_hosts** command has no arguments.

DESCRIPTION

The **remove_distributed_hosts** command removes all the hosts that were added to the list of distributed hosts by the **add_distributed_hosts** command. The command will only remove all the hosts provided none of the slave processes have been launched. If any of the slave processes have been launched then the command will fail.

EXAMPLES

In the following example, the following distributed hosts are added platinum1, two 64 bit hosts platinum2, four 32 bit hosts 1 lsf queue, four 64 bit hosts All entries are then removed.

```
prompt> add_distributed_hosts -farm now -num_of_hosts 2 platinum1
1
prompt> add_distributed_hosts -farm now -num_of_hosts 4 platinum2 -32bit
1
prompt> add_distributed_hosts -farm lsf -submission_script "/bin/
lsf_stub"
-options {-R "tmp>300" -q} -num_of_hosts 4
1
prompt> remove_distributed_hosts
1
```

SEE ALSO

[add_distributed_hosts\(2\)](#)

remove_distributed_route

Terminates a job.

SYNTAX

```
integer remove_distributed_route  
-job jobId
```

Data Types

jobId integer

ARGUMENTS

```
-job jobId  
    Specified a job ID.
```

DESCRIPTION

Passes in a job ID, and the program terminates the process that runs the specified job.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example is to kill the process of a job with ID 100.

```
prompt> remove_distributed_route -job 100
```

SEE ALSO

```
set_distributed_route(2)  
report_distributed_route(2)  
close_distributed_route(2)
```

remove_dont_touch_placement

Removes the restriction of fixed placement from a leaf cell and/or cluster.

SYNTAX

```
status remove_dont_touch_placement  
object_list
```

Data Types

object_list list

ARGUMENTS

object_list
Specifies a list of objects, each of which is a leaf cell or a cluster, from which the restriction will be removed.

DESCRIPTION

The **remove_dont_touch_placement** command removes the restriction for fixed placement from a leaf cell and/or a cluster. It can remove FIXED PLACEMENT, defined in the PDEF file for a cell, or the restriction set by the **set_dont_touch_placement** command on a cell.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows how to remove the restriction from the instance INST_1.

```
prompt> remove_dont_touch_placement INST_1
```

SEE ALSO

```
create_placement(2)  
set_cell_location(2)  
set_dont_touch_placement(2)
```

remove_dp_int_round

Remove the rounding attribute from datapath output nets.

SYNTAX

```
int remove_dp_int_round  
nets
```

Data Types

nets list

ARGUMENTS

nets List of nets that the rounding attributes will be removed from.

DESCRIPTION

This command removes the external and internal rounding attributes that are set by **set_dp_int_round** command.

Note: When a design is compiled and is been internally rounded. Constant propagation may happen in the design. If you remove the internal rounding attribute and do compile again. The final netlist may fail Formality check as constant propagation may disconnect some nets. So it is not suggested to remove the internal rounding attribute and do incremental compile.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes the external rounding value on net z1*:

```
prompt> remove_dp_int_round [find net z1*] 7 5
```

SEE ALSO

set_dp_int_round(2)

remove_drc_error

Removes one or more error objects from the specified error view.

SYNTAX

```
status remove_drc_error  
-drc_error drc_error  
[-error_view mw_error_view]  
drc_error
```

Data Types

<i>drc_error</i>	list or collection
<i>mw_error_view</i>	list or collection of one item

ARGUMENTS

```
-drc_error drc_error  
    Specifies the error objects to remove.  
    This is a required option.
```

```
-error_view mw_error_view  
    Specifies the error view from which to remove the error objects.  
    If you do not specify this option, the error objects are removed from the  
    top-level design. You can specify at most only one error view; specifying  
    more than one error view causes an error.
```

DESCRIPTION

This command removes error objects from the specified error view.

EXAMPLES

The following example gets error objects of the "Short" error type in the current top-level design, and then removes them:

```
prompt> set errs [get_drc_error -type "Short"]  
{1280}  
prompt> remove_drc_error -drc_error $errs  
1
```

SEE ALSO

```
create_drc_error(2)  
create_open_drc_error(2)  
create_open_locator_drc_error(2)  
create_short_drc_error(2)  
create_spacing_drc_error(2)  
get_drc_errors(2)
```

remove_driving_cell

Removes driving cell attributes from the specified input or inout ports of the current design.

SYNTAX

```
int remove_driving_cell  
[port_list]
```

Data Types

port_list list

ARGUMENTS

port_list
Specifies a list of names of input or inout ports in the current design, from which the driving cell attributes are to be removed. If more than one port is specified, they must be enclosed in either quotes ({}).

DESCRIPTION

Removes attributes associated with an external driving cell from the specified input or inout ports in the current design.

To view drive information on ports, use **report_port -drive**.

The **characterize** command automatically sets driving cell attributes on subdesign ports based on their context in the entire design.

This command has the same functionality as **reset_design** in removing the attributes on the ports.

Note: **remove_driving_cell** removes any corresponding rise or fall drive resistance attributes (from **set_drive**) from the specified ports. If possible, always use **remove_driving_cell** instead of **set_drive**, because **remove_driving_cell** is more accurate. This command replaces **set_driving_cell -none** in the 1999.05 version of Design Compiler.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example removes driving cell attributes from all input and inout ports.

```
prompt> remove_driving_cell all_inputs()
```

SEE ALSO

`all_inputs(2)`
`characterize(2)`
`current_design(2)`
`report_port(2)`
`reset_design(2)`
`set_drive(2)`
`set_driving_cell(2)`

remove_edit_groups

Removes a list of edit groups.

SYNTAX

```
remainder remove_edit_groups  
[-all]  
[-quiet]  
[objects]
```

ARGUMENTS

-all
Specifies that all edit groups are to be removed.

-quiet
Specifies that error and warning messages should be suppressed.

objects
Specifies the edit groups to be removed.
Note: this option precludes the *-all* option.

RETURNS

remainder
Contains a collection of undeleted objects and children of deleted edit groups.
Any object specified in the *objects* option which was not deleted either due to database failure or because it was not an edit group is added to this collections. The children of any successfully deleted edit groups are also added.
This return value allows the user to chain edit group operations together (See examples below).

DESCRIPTION

Removes a list of edit groups from the current design.

EXAMPLES

The following example removes the edit groups from the currently selected objects:

```
> remove_edit_groups [get_selection]
```

The following example creates a new edit group from the currently selected objects which may already contain existing edit groups:

```
> create_edit_group [remove_edit_groups [get_selection]]
```

SEE ALSO

`remove_objects(2)`

remove_filler_with_violation

Deletes filler cells that have routing violations

SYNTAX

```
integer remove_filler_with_violation  
-name cell_name
```

Data Types

cell_name list

ARGUMENTS

```
-name cell_name  
    The pattern of the names of the cell instances
```

DESCRIPTION

Deletes all the filler cells that have routing design rule violations. Filler cells are standard cells without any connections to signal nets. The command is useful when filler cells are added after signal routing.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes all filler cells with the prefix "xofiller" that have routing violations.

```
prompt> remove_filler_with_violation -name "xofiller.*"
```

SEE ALSO

```
insert_stdcell_filler(2)
```

remove_flip_chip_route

Removes the specified flip-chip wires and contacts.

SYNTAX

```
status remove_flip_chip_route  
[-nets collection_of_nets  
  | -nets_in_file nets_file]  
[-width width]  
[-contact]
```

Data Types

<i>collection_of_nets</i>	collection
<i>nets_file</i>	string
<i>width</i>	float

DATA TYPES

<i>collection_of_nets</i>	collection
<i>nets_file</i>	string
<i>width</i>	float

ARGUMENTS:

-nets *collection_of_nets*
Specifies the flip-chip wires to remove.
This option and the **-nets_in_file** option are mutually exclusive; you can specify only one. If you do not specify either option, all flip-chip nets are removed.

-nets_in_file *nets_file*
Specifies the name of the file that contains the flip-chip wires to remove.
This option and the **-nets** option are mutually exclusive; you can specify only one. If you do not specify either option, all flip-chip wires are removed.

-width *width*
Removes only the flip-chip wires with the specified width. The unit for the width value is the unit specified in the technology file.

-contact
Removes the contacts of the specified wires, as well as the wires. By default, the wires are removed but the contacts are not.

DESCRIPTION

This command removes the specified flip-chip wires and contacts.

EXAMPLES

The following command removes all flip-chip wires.

```
prompt> remove_flip_chip_route
```

The following command removes all flip-chip wires and contacts.

```
prompt> remove_flip_chip_route -contact
```

The following command removes all flip-chip wires with a width of 10 user units.

```
prompt> remove_flip_chip_route -width 10
```

SEE ALSO

```
set_route_flip_chip_options(2)  
route_flip_chip(2)  
create_stack_via_on_pad_pin(2)  
write_flip_chip_nets(2)  
push_flip_chip_route(2)  
display_flip_chip_route_flylines(2)  
optimize_flip_chip_route(2)
```

remove_fp_block_shielding

Removes signal shielding (route blockages) created by create_fp_block_shielding.

SYNTAX

```
status remove_fp_block_shielding  
[-inside_boundary]  
[-outside_boundary]  
[-side_list {left | right | top | bottom}]  
[-metal_layers layer_list]  
[objects]
```

ARGUMENTS

-inside_boundary

Specifies that the command will remove shielding on the inside of plan group(s) and/or soft macro(s). If this option and the -outside_boundary option are not specified, then shielding is removed on both the interior and exterior sides of the plan group(s) and/or soft macro(s).

-outside_boundary

Specifies that the command will remove shielding on the outside of the plan group(s) and/or soft macro(s). If this option and the -inside_boundary option are not specified, then shielding is removed on both the interior and exterior sides of the plan group(s) and/or soft macro(s).

-side_list {left | right | top | bottom}

Indicates the side(s) on which the command will remove shielding rectangles. If this option is not specified, then shielding is removed on all sides of the plan group(s) and/or soft macro(s).

-metal_layers *layer_list*

A layer object collection that specifies the metal layer(s) on which shielding will be removed. If no layers are specified with this option, then the command removes shielding from all layers.

objects

Collection of plan group(s) and/or soft macro(s) from which to remove route blockages (shielding). If omitted, then all plan groups and all soft macros will be processed.

DESCRIPTION

This command removes signal shielding (route blockages) created by create_fp_block_shielding. A list of plan groups and/or soft macros can be specified to be processed. If none are specified, then the command removes shielding from all plan groups and soft macros that contain existing shielding rectangles.

The command can be flagged to remove shielding on the inside and/or outside of a plan group/soft macro.

The command accepts a list of sides that indicates the sides on which shielding should be removed. If no side is specified, the command removes shielding on all sides.

The metal layers where shielding will be removed can be specified. If no layers are specified, then the command will remove shielding on each metal layer. The argument to this option is a collection of layer objects returned by the `get_layer` command (example `[get_layer {metal1 metal2}]`). Layer names or mask names are acceptable. Use of `[get_layer {layer_list}]` is optional... the following option is acceptable: -
`metal_layers {metal1 metal2 metal3}`. Be sure the names are either legal layer names or legal mask names.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes inside shielding on left side of selected plan groups and soft macros:

```
> remove_fp_block_shielding -inside_boundary -side_list left  
[get_cells {large_macro small_macro medium_plan_group}]  
> remove_fp_block_shielding -metal_layers [get_layer {metal1 metal2 metal3}] -  
side_list {left right}  
[get_selection]
```

SEE ALSO

`create_fp_block_shielding(2)`

remove_fp_feedthroughs

Removes feedthrough ports and nets that exist within the design.

SYNTAX

```
status remove_fp_feedthroughs  
[-include {buffered original}]  
[-nets object_list]  
[-blocks object_list]  
[-voltage_areas object_list]
```

Data Types

object_list collection

ARGUMENTS

-include {buffered original}

This option indicates whether the user wants to remove buffered, or original, or both buffered and original feedthroughs in addition to the default. The supported values are described below.

buffered

When **-include buffered** is specified, the command removes buffered feedthrough ports that are determined to be "added" in addition to unbuffered feedthroughs, and removes the connections to these ports. The tool will also remove these feedthrough buffers and inverting buffers.

The default behavior of **remove_fp_feedthroughs** is to preserve the buffered feedthroughs and their connections.

original

When **-include original** is specified, the command removes feedthrough ports that are determined to be "original" in addition to "added" ports. If you specify this option, the tool removes these feedthrough ports and the connections to these ports at the top level.

The default behavior of **remove_fp_feedthroughs** is to preserve "original" ports and their top-level connections.

-nets object_list

Specifies the flat nets from which to remove feedthrough ports. If you do not specify this option, all flat nets are processed for feedthrough removal.

-blocks object_list

Specifies the plan groups and soft macros from which to remove route feedthrough ports and nets. If you do not specify this option, all plan groups and all soft macros are processed.

`-voltage_areas object_list`

Specifies the voltage areas from which to remove route feedthrough ports and nets.

If you do not specify this option, all voltage areas are processed.

DESCRIPTION

This command removes feedthrough ports and nets that exist within the design. This command, when specified with no arguments, removes all "added" feedthrough nets and ports ("original" nets and ports on blocks voltage areas are preserved by default) from all nets on all blocks (including both plan group blocks and soft macro blocks) and all voltage areas. The command determines by examination which ports and nets are feedthrough nets. If you want to apply the command to a subset of nets, use the **-nets** option to specify a collection of flat nets to be processed. Similarly, if you want to apply the command to certain blocks or voltage areas, use the `f-blocks` or `f-voltage_areas` options to specify a collection of plan group and soft macro blocks or voltage areas to be processed.

The command accepts the **-include original** option to instruct the command to remove nets and ports that are deemed to be "original" in addition to "added" nets and ports. The "original" is defined as all ports on blocks and voltage areas that are determined to be feedthrough ports, but are not determined to be "added" ports. By default, these "original" nets and ports are not removed. If you specify this option, the tool removes original feedthrough ports and their connections to the top-level nets.

When **-include buffered** option is specified, the tool can detect the feedthroughs with buffers inserted, remove the "added" feedthrough ports and delete the inserted buffers and inverting buffers where the buffered feedthroughs are removed.

This command also accepts the **-include {buffered original}** option to allow users to remove both buffered and "original" feedthrough nets and ports in addition to the default.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes feedthrough nets and ports on all nets, blocks and voltage areas:

```
prompt> remove_fp_feedthroughs
```

The following example removes feedthrough nets and ports on all specified nets and blocks:

```
prompt> remove_fp_feedthroughs -nets {[get_selection]} \  
      -blocks {[get_cells "BlockA BlockB"]}
```

The following example removes feedthrough nets in the selection on at least one of

PGa, PGb, or VAc:

```
prompt> remove_fp_feedthroughs -nets {[get_selection]} \  
        -blocks {[get_plan_groups {PGa PGb}}] \  
        -voltage_areas [get_voltage_areas VAc]
```

The following example removes feedthrough nets and ports on all specified voltage areas:

```
prompt> remove_fp_feedthroughs \  
        -voltage_areas {[get_voltage_areas "voltage_area_0"]}
```

The following example removes feedthrough nets and ports on all nets, blocks and voltage areas and removes the original feedthrough ports:

```
prompt> remove_fp_feedthroughs -include original
```

The following example removes feedthrough nets and ports on all nets, blocks and voltage areas. In addition, it removes the original feedthrough ports, removes the added buffered feedthrough ports, and deletes the buffers where buffered feedthroughs have been removed:

```
prompt> remove_fp_feedthroughs -include {buffered original}
```

SEE ALSO

`analyze_fp_routing(2)`

remove_fp_pin_constraints

Removes the pin assignment constraints for the specified soft macros, plan groups, or ports.

SYNTAX

```
status remove_fp_pin_constraints  
[-block_level]  
[blocks_or_ports]
```

Data Types

blocks_or_ports collection

ARGUMENTS

-block_level

Removes the block-level pin assignment constraints which are honored when the **-block_level** option of **place_fp_pins** is used. Non-port arguments supplied to **remove_fp_pin_constraints** will be ignored if this option is used.

blocks_or_ports

Specifies the soft macros, plan groups, or ports from which to remove the pin assignment constraints. If you do not specify any argument, and the option **-block_level** is not used the pin assignment constraints are removed from all soft macros and plan groups. The block-level pin assignment constraints are removed if **-block_level** is used and no argument is specified. Port arguments are accepted only when **-block_level** is in use.

Note: Certain pin constraints are global (i.e. they apply to all blocks). These are the constraints set with the following options of **set_fp_pin_constraints**: **-keep_buses_together**, **-bus_ordering**, **-scramble_skip**, **-nets**, and **-exclude_nets**. These options will only be removed if **remove_fp_pin_constraints** is called with no argument (i.e. pin constraints are removed from all blocks).

DESCRIPTION

This command removes the pin assignment constraints for the specified soft macros and plan groups. Pin assignment constraints are specified with the **set_fp_pin_constraints** command. The **place_fp_pins** and **analyze_fp_routing** commands use these constraints when creating pins.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes the top-level pin assignment constraints for all soft

macros and plan groups in the current design.

```
prompt> remove_fp_pin_constraints
```

SEE ALSO

```
analyze_fp_routing(2)  
place_fp_pins(2)  
set_fp_pin_constraints(2)
```

remove_fp_pin_overlaps

Removes overlaps between all pins and pre-routes on the specified soft macros and the specified top-level or soft macro pins.

SYNTAX

```
status remove_fp_pin_overlaps  
[-cells collection]  
[-pins collection]
```

ARGUMENTS

```
-cells collection  
    collection of soft macros to remove pin overlaps for  
  
-pins collection  
    collection of pins to remove overlaps for
```

DESCRIPTION

Removes overlaps between all pins and pre-routes on the specified soft macros and/or specified terminals or soft macro pins. Overlaps that do not result in shorts will not be removed. For example, an overlap between a pin and a pre-route connected to the same net will be left untouched.

If no cells or pins are specified then overlaps are removed from all top level and all soft macro pins.

See the example below if overlap removal is desired only for top-level pins (i.e. terminals).

If overlap removal fails (because there are too few wiretracks available or pin placement is overconstrained) then error messages will be output. This command is a localized operation -- no attempt is made on searching other metal layers or other sides of the cell boundary for available wire tracks.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The example below removes pin overlaps for all pins on the soft macro alu1

```
prompt> remove_fp_pin_overlaps -cells [get_cells alu1]
```

The example below removes pin overlaps for top-level pins (i.e. terminals) only. Soft macro pins are left untouched.

```
prompt> remove_fp_pin_overlaps -pins [get_terminals *]
```

SEE ALSO

`sort_fp_pins(2)`

remove_fp_plan_group_padding

Deletes the padding from plan groups.

SYNTAX

```
status_value remove_fp_plan_group_padding  
plan_groups
```

Data Types

plan_groups collection or list

ARGUMENTS

plan_groups
 Specifies a collection of plan groups on which to remove padding.

DESCRIPTION

This command removes both internal and external padding on all specified plan groups.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes padding on a collection of plan groups named *group_a*.

```
prompt> remove_fp_plan_group_padding group_a
```

SEE ALSO

`create_fp_plan_group_padding(2)`

remove_fp_rail_stacked_via

Removes the selected power network stacked vias from the design based on the stacked via reduction results from the **reduce_fp_rail_stacked_via** command. These stacked vias are removed to reduce congestion.

SYNTAX

```
int remove_fp_rail_stacked_via
```

ARGUMENTS

none

DESCRIPTION

Removes the selected power network stacked vias from the design based on the stacked via reduction results from the **reduce_fp_rail_stacked_via** command. Before you use this command to actually remove the selected vias, you can preview the stacked via removal results in terms of reducing congestion, voltage (IR) drop and Electromigration (EM).

EXAMPLES

The following example removes the selected stacked vias from **reduce_fp_rail_stacked_via**.

```
prompt> remove_fp_rail_stacked_via
```

SEE ALSO

```
reduce_fp_rail_stacked_via(2)
```

remove_fp_rail_voltage_area_constraints

Removes the multi-voltage power network synthesis (PNS) constraints in the specified voltage area(s).

SYNTAX

```
status remove_fp_rail_voltage_area_constraints
[-voltage_area voltage_area]
[-all]
-layer string
```

Data Types

voltage_area collection or list

ARGUMENTS

-voltage_area *voltage_area*
Specify the voltage area name(s). This option is mutually exclusive with the **-all** option; you must specify only one.

-all
Remove multi-voltage PNS constraints in all voltage area(s). This option is mutually exclusive with the **-voltage_area** option; you must specify only one.

DESCRIPTION

This command removes the multi-voltage PNS constraints in specified voltage area(s).

Multicorner-Multimode Support

This command is not supported in a multi-scenario design flow.

EXAMPLES

The following example shows how to remove the multi-voltage PNS constraints in all voltage area(s).

```
prompt> remove_fp_rail_voltage_area_constraints -all
```

SEE ALSO

```
set_fp_rail_voltage_area_constraints(2)
set_fp_rail_constraints(2)
set_fp_block_ring_constraints(2)
synthesize_fp_rail(2)
```

remove_fp_relative_location

Removes previously set relative location constraint(s).

SYNTAX

```
status remove_fp_relative_location
[-name constraint_name]
[-target_cells cells]
```

Data Types

<i>constraint_name</i>	string
<i>cells</i>	collection

ARGUMENTS

-name *constraint_name*
Specifies name of the relative location constraint to remove. By default, if no option is specified for this command, all relative location constraints are removed.

-target_cells *cells*
This option specifies the collection of macro cells whose relative location constraints are to be removed. By default, if no option is specified for this command, all relative location constraints are removed.

DESCRIPTION

The **remove_fp_relative_location** command removes specified or all relative location constraints previously set by **set_fp_relative_location** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows how to relative location constraint named "rp1" and the relative location constraint set on all cells whose names match "k_cell*".

```
prompt>remove_fp_relative_location \
        -name rp1 \
        -target_cells [get_cells "k_cell*"]
```

SEE ALSO

```
set_fp_relative_location(2)
extract_fp_relative_location(2)
```

remove_fp_relative_location
1506

remove_fp_virtual_pad

Remove virtual power or ground pads for PNA and PNS.

SYNTAX

```
status remove_fp_virtual_pad
[-nets nets]
[-layer layer]
[-point {x y}]
[-all]
```

ARGUMENTS

-nets *nets*
Specifies the net name(s) of the power or ground virtual pads to be removed.

-layer *layer*
Specifies the layer name of the virtual pad to be removed.

-point {*x y*}
Specifies the location of the virtual pad to be removed. The unit in the {*x y*} coordinate is micron.

-all
Remove all virtual pads.

DESCRIPTION

Removes the virtual power and ground pads created by **create_fp_virtual_pad** command for PNS or PNA based on nets, layer and location.

Multicorner-Multimode Support

This command is not supported in a multi-scenario design flow.

EXAMPLES

The following example shows how to remove a virtual pad for net VDD on the metal layer M6 at the location {560.00 485.40}.

```
prompt> remove_fp_virtual_pad\
-nets VDD\
-layer M6\
-point {560.00 485.40}
```

SEE ALSO

`analyze_fp_rail(2)`

```
create_fp_virtual_pad(2)
synthesize_fp_rail(2)
```

remove_fp_voltage_area_constraints

Resets the feedthrough constraints for the specified voltage areas to the default values.

SYNTAX

```
status remove_fp_voltage_area_constraints
```

ARGUMENTS

The **remove_fp_voltage_area_constraints** command has no arguments.

DESCRIPTION

This command resets the feedthrough constraints for the specified voltage areas to the default values. You specify the voltage area feedthrough constraints with the **set_fp_voltage_area_constraints** command. The **route_global** and **analyze_fp_routing** commands use these constraints when creating logical pins on voltage areas.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example resets the feedthrough constraints for all voltage areas in the current design to the default values.

```
prompt> remove_fp_voltage_area_constraints
```

SEE ALSO

```
analyze_fp_routing(2)  
report_fp_voltage_area_constraints(2)  
route_global(2)  
set_fp_voltage_area_constraints(2)
```

remove_from_collection

Removes objects from a collection, resulting in a new collection. The base collection remains unchanged.

SYNTAX

```
collection remove_from_collection  
base_collection  
object_spec
```

Data Types

<i>base_collection</i>	collection
<i>object_spec</i>	string

ARGUMENTS

base_collection
Specifies the base collection to be copied to the result collection. Objects matching *object_spec* are removed from the result collection.

object_spec
Specifies a list of named objects or collections to remove. The object class of each element in this list must be the same as in the base collection. If the name matches an existing collection, the collection is used. Otherwise, the objects are searched for in the database using the object class of the base collection.

DESCRIPTION

The **remove_from_collection** command removes elements from a collection, creating a new collection.

If the base collection is homogeneous, any element of the *object_spec* that is not a collection is searched for in the database using the object class of the base collection. If the base collection is heterogeneous, then any element of the *object_spec* that is not a collection is ignored.

If nothing matches the *object_spec*, the resulting collection is a copy of the base collection. If everything in *base_collection* matches the *object_spec*, the result is the empty collection.

For background on collections and querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example from PrimeTime gets all input ports except "CLOCK".

```
prompt> set cPorts [remove_from_collection [all_inputs] CLOCK]
{"in1", "in2"}
```

SEE ALSO

```
add_to_collection(2)
collections(2)
query_objects(2)
```

remove_from_rp_group

Removes an item (cell, relative placement group, or keepout) from the specified relative placement groups.

SYNTAX

```
status remove_from_rp_group  
rp_groups  
-leaf cell_name  
-hierarchy group_name  
[-instance instance_name]  
-keepout keepout_name
```

Data Types

<i>rp_groups</i>	list or collection
<i>cell_name</i>	string
<i>group_name</i>	string
<i>instance_name</i>	string
<i>keepout_name</i>	string

ARGUMENTS

rp_groups
Specifies the relative placement groups from which to remove an item. The groups must all be in the same design.
If you do not specify this argument, no items are removed.

-leaf *cell_name*
Specifies the name of a leaf cell to remove from the specified relative placement groups. This option cannot be used with any other option.

-hierarchy *group_name*
Specifies the name of the relative placement group to remove from the specified relative placement groups.
This option can be used with the **-instance** option, but is mutually exclusive with **-leaf** and **-keepout**.

-instance *instance_name*
Specifies the name of the hierarchical cell that contains the instantiated relative placement group specified in the **-hierarchy** option.

-keepout *keepout_name*
Specifies the name of the keepout to remove from the specified relative placement groups. This option cannot be used with any other option.

DESCRIPTION

This command removes an item from specified relative placement groups. The item can be either a leaf cell, a relative placement group (included or instantiated), or a keepout. This command is supported only for designs that do not contain multiply-

instantiated designs. Relative placement usage is available with Design Compiler Ultra.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses **remove_from_rp_group** to remove a cell from an existing relative placement group.

```
prompt> get_rp_groups ripple::grp_ripple
{ripple::grp_ripple}
prompt> remove_from_rp_group ripple::grp_ripple\
-leaf carry_in_1
1
```

SEE ALSO

```
add_to_rp_group(2)
create_rp_group(2)
get_rp_groups(2)
remove_rp_groups(2)
write_rp_groups(2)
```

remove_generated_clock

Removes a generated_clock object.

SYNTAX

```
string remove_generated_clock  
-all |  
clock_list
```

Data Types

clock_list list

ARGUMENTS

-all
 Specifies that all the generated clocks be removed.

clock_list
 Provides a list of generated clock names.

DESCRIPTION

This command removes the generated clocks in the design. If the generated clocks are expanded, the actual clocks are also deleted. All the attributes set on generated clocks (**false_paths** **multicycle_paths**, and so on) are also removed.

To create generated clock in the design, use the **create_generated_clock** command.

To show information about clocks and generated clocks in the design, use the **report_clock** command.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example removes the generated clock GEN1 from the design.

```
prompt> remove_generated_clock GEN1
```

The following example removes all the generated clocks from the design.

```
prompt> remove_generated_clock -all
```

SEE ALSO

`create_generated_clock(2)`

`remove_generated_clock`
1514


```
report_clock(2)
```

remove_host_options

Remove host or pool definitions created by **set_host_options**.

SYNTAX

```
status remove_host_options  
[-all]  
[-name options_name]  
[name]
```

Data Types

options_name string

ARGUMENTS

- all**
Specifies that all host options should be deleted.
- name** *options_name*
The name of the set of options to be deleted. It is an error to give both **-name** and **-all**.

DESCRIPTION

The **remove_host_options** command is used to unset the parameters set by the **set_host_options** command.

If the **-name** argument is given, any host/pool options of that name will be deleted. If the **-all** argument is given, all host/pool options will be deleted, and the **-max_cores** value for the main (parent) job will be unset.

SEE ALSO

set_host_options(2)
report_host_options(2)

remove_ideal_latency

Removes ideal latency information from the specified objects.

SYNTAX

```
string remove_ideal_latency  
[-rise | -fall] [-min | -max] object_list | -all
```

Data Types

object_list list

ARGUMENTS

-rise | **-fall**
Specifies that rise or fall ideal latency should be removed. By default, ideal latency is removed for both rise and fall ideal latency. The **-rise** and **-fall** options are mutually exclusive.

-min | **-max**
Specifies that maximum or minimum ideal latency should be removed. By default, it is removed for both minimum and maximum delay analysis. The **-min** and **-max** options are mutually exclusive.

object_list
Specifies a list of ports or pins from which to remove ideal latency.

-all
Remove ideal network latency from all objects.

DESCRIPTION

Removes from specified objects the user-specified ideal latency that was previously set by the **set_ideal_latency** command. See the **set_ideal_latency** man page for more details. You must either specify an *object_list* or use **-all**.

You can remove selected portions of ideal latency by specifying applicable **-rise**, **-fall**, **-min**, and **-max** options.

To list ideal latency values, use the **report_ideal_network** command.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example removes ideal latency from the output pin named *Z* of cell instance *U1/U2/U3*.

```
prompt> remove_ideal_latency {U1/U2/U3/Z}
```

The following example removes only rise ideal latency information from a port named A.

```
prompt> remove_ideal_latency -rise {A}
```

SEE ALSO

```
current_design(2)  
remove_ideal_network(2)  
remove_ideal_transition(2)  
report_ideal_network(2)  
report_timing(2)  
set_auto_disable_drc_nets(2)  
set_ideal_net(2)  
set_ideal_network(2)  
set_ideal_transition(2)
```

remove_ideal_net

Restores the ideal nets set by the **set_ideal_net** or **set_ideal_network -no_propagate** command to their initial nonideal state from the specified nets in the current design.

SYNTAX

```
status remove_ideal_net  
net_list
```

Data Types

```
net_list      list
```

ARGUMENTS

```
net_list
```

Specifies a list of net names from which to remove the **ideal_net** attribute. These nets must be visible from the current design.

DESCRIPTION

Removes the **ideal_net** attribute from the individual nets specified in *net_list*; or restores the ideal nets that are set by the **set_ideal_net** command or the **set_ideal_network -no_propagate** command to their initial nonideal state from the specified nets in *net_list*. Those nets must be visible from the current design.

Ideal nets are networks of nets that are free from the max_capacitance and max_fanout design rule constraints. Ideal nets are useful to reduce DRC violations caused by clock trees, because these networks usually have high max_capacitance and max_fanout violations.

To remove the **auto_disable_drc_net** attribute from nets, use **set_auto_disable_drc_nets -none**. The **reset_design** command removes all attributes from the design, including the **ideal_net** and **auto_disable_drc_net** attributes. By default, the tool still treats clock networks as ideal nets.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes the **ideal_net** attribute from individual nets in the design CLOCK_GEN:

```
prompt> current_design CLOCK_GEN  
prompt> remove_ideal_net [get_nets]
```

SEE ALSO

`reset_design(2)`
`set_auto_disable_drc_nets(2)`
`set_dont_touch(2)`
`set_dont_touch_network(2)`
`set_ideal_net(2)`

remove_ideal_network

Removes a set of ports or pins in an ideal network in the current design. Cells and nets in the transitive fanout of the specified objects are no longer treated as ideal.

SYNTAX

```
int remove_ideal_network  
object_list | -all
```

Data Types

object_list list

ARGUMENTS

object_list
Specifies a list of objects (pins or ports) that are sources of an ideal network. If more than one object name is specified, enclose the objects in quotes or braces ({}). The source objects can be input ports on the design or any internal pin except pins at hierarchical boundaries. These objects must be visible from the current design.
This argument and the **-all** option are mutually exclusive.

-all
Removes all ideal networks from the current design.
This option and the *object_list* argument are mutually exclusive.

DESCRIPTION

This command removes the ideal network designation from ports or pins in an ideal network.

Ideal networks, which are an extension of ideal nets, incorporate automatic propagation of the **ideal** attribute. You specify only the source ports or source pins of the network. All nets, cells, and pins on the transitive fanin of these objects are treated as ideal by the **compile** command.

The **ideal_network** attribute is automatically spread by the tool and respread as needed during optimizations. The **remove_ideal_network** command returns to normal part or all of a network that was previously marked as ideal using the **set_ideal_network** command. See the **set_ideal_network** man page for details about the propagation rules.

In addition to disabling timing update and timing optimizations, all cells in the ideal network are set with the **dont_touch** attribute. When part or all of the network is restored to normal using **remove_ideal_network**, the original **dont_touch** status of cells is restored.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets up an ideal network on objects in the design CLOCK_GEN and then removes part of the network:

```
prompt> current_design CLOCK_GEN
prompt> set_ideal_network {port1 port2}
prompt> remove_ideal_network port2
```

SEE ALSO

```
remove_ideal_net(2)
report_ideal_network(2)
reset_design(2)
set_auto_disable_drc_nets(2)
set_dont_touch(2)
set_dont_touch_network(2)
set_ideal_latency(2)
set_ideal_net(2)
set_ideal_network(2)
set_ideal_transition(2)
```


remove_ideal_transition

Removes ideal transition information from the specified objects.

SYNTAX

```
string remove_ideal_transition  
[-rise | -fall] [-min | -max] object_list | -all
```

Data Types

object_list list

ARGUMENTS

-rise | **-fall**
Specifies that rise or fall ideal transition should be removed. By default, ideal transition is removed for both rise and fall ideal transition. The **-rise** and **-fall** options are mutually exclusive.

-min | **-max**
Specifies that maximum or minimum ideal transition should be removed. By default, ideal transition is removed for both maximum and minimum ideal transition. The **-min** and **-max** options are mutually exclusive.

object_list
Specifies a list of ports or pins from which to remove ideal transition.

-all
Remove ideal network transition from all objects.

DESCRIPTION

Removes from specified objects the user-specified ideal transition that was previously set by the **set_ideal_transition** command. See the **set_ideal_transition** man page for more details.

You can remove selected portions of ideal transition by specifying applicable **-rise**, **-fall**, **-min**, and **-max** options.

To list ideal transition values, use the **report_ideal_network** command.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example removes ideal transition from the output pin named Z of cell instance U1/U2/U3.

```
prompt> remove_ideal_transition {U1/U2/U3/Z}
```

The following example removes only rise ideal transition information a port named A.

```
prompt> remove_ideal_transition -rise {A}
```

SEE ALSO

```
current_design(2)  
remove_ideal_latency(2)  
remove_ideal_network(2)  
report_ideal_network(2)  
report_timing(2)  
set_auto_disable_drc_nets(2)  
set_ideal_net(2)  
set_ideal_network(2)  
set_ideal_transition(2)
```

remove_ignore_cell_timing

Undoes the effects of set_ignore_cell_timing

SYNTAX

```
int remove_ignore_cell_timing  
cell_list
```

Data Types

cell_list list

ARGUMENTS

cell_list
Specifies a collection of cells which are no longer should be ignored by timing related commands.

DESCRIPTION

The **remove_ignore_cell_timing** command removes the "ignore_cell" attribute on the cell(s) that was previously marked as ignored by the **set_ignore_cell_timing** command. By removing the ignore_cell attribute, the cells are now part of the network, and should not be ignored by the timing related commands.

Multicorner-Multimode Support

This command uses information from active scenarios only.

EXAMPLES

The following example set an ignored attribute on a cell in the design CLOCK_GEN, this cell thus be ignored by timing related commands:

```
prompt> current_design CLOCK_GEN  
prompt> set_ignore_cell_timing cell12  
prompt> vBremove_ignore_cell_timing cell12
```

SEE AL SO

set_ignore_cell_timing(2).

remove_ignored_layers

Removes ignored routing layers in congestion analysis and RC estimation. This command can also remove the design minimum and maximum routing layers if those layers have been already set.

SYNTAX

```
int remove_ignored_layers  
list_of_layers  
[-all]  
[-min_routing_layer]  
[-max_routing_layer]
```

ARGUMENTS

list_of_layers
Lists the routing layers that should no longer be ignored during congestion analysis and RC estimation.

[-all]
Specifies that no routing layers should be ignored during congestion analysis and RC estimation.

[-min_routing_layer]
Removes the setting for the design minimum routing layer.

[-max_routing_layer]
Removes the setting for the design maximum routing layer.

DESCRIPTION

You can specify routing layers to be ignored during congestion analysis and RC estimation. This command removes the ignored setting from some or all of these layers.

Note that congestion analysis and RC estimation require at least one horizontal and one vertical layer are not ignored.

You can also use this command to remove the settings for the design minimum and maximum routing layers.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

```
prompt> remove_ignored_layers -all
```

SEE ALSO

`set_ignored_layers(2)`
`report_ignored_layers(2)`
`report_lib(2)`

remove_input_delay

Removes input delay on pins or input ports.

SYNTAX

```
int remove_input_delay  
[-clock clock] [-clock_fall] [-level_sensitive]  
[-rise]  
[-fall]  
[-max]  
[-min]  
port_pin_list
```

Data Types

<i>clock</i>	string or collection of one object
<i>port_pin_list</i>	list

ARGUMENTS

- clock *clock***
Removes delay information relative to the specified clock edge. If **-clock** is not specified, all input delay is removed. To remove input delay that has no relative clock, use **remove_input_delay -clock ""**.
The clock can be specified as either a string or a collection of one object.
- clock_fall**
Removes input delay relative to the clock falling edge. If **-clock_fall** is not specified, input delay relative to the clock rising edge is removed.
This option is only used with the **-clock** option.
- level_sensitive**
Removes level-sensitive input delay. If this option is not used, only nonlevel-sensitive input delay is removed.
This option is only used with the **-clock** option.
- rise**
Removes rising delay on *port_pin_list*. If you do not specify **-rise** or **-fall**, both rising and falling delays are removed.
- fall**
Removes falling delay on *port_pin_list*. If you do not specify **-rise** or **-fall**, both rising and falling delays are removed.
- max**
Removes maximum input delays on *port_pin_list*. If you do not specify **-max** or **-min**, both maximum and minimum input delays are removed.
- min**
Removes minimum input delays on *port_pin_list*. If you do not specify **-max** or **-min**, both maximum and minimum input delays are removed.

`port_pin_list`

Specifies list of port or pin names in the current design. Input delay is removed from each object in `port_pin_list`. To specify more than one object, enclose the objects in quotes (") or braces ({}).

DESCRIPTION

This command removes input delay values for objects in the current design. Set the input delay using the **set_input_delay** or the **characterize** command. By default, all input delay on each object in `port_pin_list` is removed. To restrict the input delay values that are removed, use the **-clock**, **-clock_fall**, **-min**, **-max**, **-rise**, or **-fall** options.

Use the **report_port** command to list input delays associated with ports. Use the **report_design** command to list input delays of internal pins.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example removes all input delay values from all input ports in the current design:

```
prompt> remove_input_delay all_inputs()
```

The following example removes input maximum rise delay values from IN1, IN3, and BIDIR12:

```
prompt> remove_input_delay -max -rise {IN1 IN3 BIDIR12}
```

The following example, removes all input delay for the IN1 port, relative to the falling edge of CLK2.

```
prompt> remove_input_delay -clock [get_clocks CLK2] -clock_fall \  
[get_ports "IN1"]
```

The following example removes all input delay not relative to any clock for port IN4.

```
prompt> remove_input_delay -clock "" IN4
```

SEE ALSO

`all_inputs(2)`
`current_design(2)`
`remove_output_delay(2)`
`report_design(2)`
`report_port(2)`
`reset_design(2)`
`set_input_delay(2)`

```
set_output_delay(2)
```


remove_io_antenna_properties

Removes external antenna properties.

SYNTAX

```
status remove_io_antenna_properties
```

DESCRIPTION

This command removes all the external antenna properties from all ports in the current design.

EXAMPLES

```
prompt>remove_io_antenna_properties
```

SEE ALSO

```
define_io_gate_size(2)  
define_io_diode_protection(2)  
define_io_antenna_area(2)  
report_io_antenna_properties(2)  
extract_hier_antenna_property(2)  
verify_zrt_route(2)  
report_antenna_ratio(2)
```

remove_io_constraints

Removes previously set physical constraints

SYNTAX

```
status remove_io_constraints  
[-cell name]  
[-pin_only]  
[-pad_only]  
[-chiplinelevel_pad_only]  
[objects]
```

Data Types

<i>name</i>	string
<i>objects</i>	collection

ARGUMENTS

-cell *name*
Specifies the Milkyway cell name or plan group whose physical constraints will be removed. The default behavior removes the current cell's physical constraints.

-pin_only
Indicates only pin physical constraints will be removed. The **-pin_only**, **-pad_only** and **chiplinelevel_pad_only** options are mutually exclusive. Use only one.

-pad_only
Indicates only pad physical constraints will be removed. The **-pin_only**, **-pad_only** and **chiplinelevel_pad_only** options are mutually exclusive. Use only one.

-chiplinelevel_pad_only
Indicates only chip-level pad physical constraints will be removed. The **-pin_only**, **-pad_only** and **chiplinelevel_pad_only** options are mutually exclusive. Use only one.

objects
Specifies the pin or pad which physical constraints are removed. If this option is not specified, the physical constraints for all pins, pads and chip level pads are removed.

DESCRIPTION

This command allows you to remove all previously set physical constraints in the currently active Milkyway cell or specified Milkyway cell for the specified pins, pads or chip-level pads. If no argument is specified, this command removes all physical constraints for all pins, pads and chip-level pads.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes the physical constraints for all pins and pads in the current design.

```
prompt> remove_io_constraints
```

The following example removes the physical constraints for all pins.

```
prompt> remove_io_constraints -pin_only
```

SEE ALSO

remove_io_pin_overlap

Spreads pins so that they do not overlap one another and have spacing violations.

SYNTAX

```
int remove_io_pin_overlap
```

ARGUMENTS

No arguments needed, the command gets current design.

DESCRIPTION

Spreads pins so that they do not overlap one another and have spacing violations. Does not move the location of fixed pins and respects placement blockages. Pins inside placement blockages are considered illegal and are moved.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example runs the command in default settings:

```
prompt> remove_io_pin_overlap  
1
```

SEE ALSO

remove_isolate_ports

Removes the specified ports from the list of ports that are isolated in the current design.

SYNTAX

```
int remove_isolate_ports  
port_list
```

Data Types

port_list list

ARGUMENTS

port_list
Specifies the ports that will be removed from the list of ports that are isolated in the current design. Port isolation must have been requested for these ports using the **set_isolate_ports** command.

DESCRIPTION

The **remove_isolate_ports** command removes the specified ports from the list of ports that are isolated in the current design.

This command will not remove the isolation cell inserted by a previous **compile** command. The isolation cell will be removed by a subsequent **compile** command issued after the **remove_isolate_ports** command, if the **max_area** attribute has been set on the current design.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes the port "qout" from the list of ports being isolated.

```
prompt> remove_isolate_ports qout
```

SEE ALSO

```
report_isolate_ports(2)  
reset_design(2)  
set_isolate_ports(2)  
set_max_area(2)
```

remove_isolation_cell

Removes specified isolation cell or cells from design.

SYNTAX

```
int remove_isolation_cell  
[-force]  
-object_list cells
```

Data Types

cells list

ARGUMENTS

-force
 Forces deletion of ISO/ELS cells for cells marked dont touch.

-object_list *cells*
 Use this option to specify the list of cells to be removed. This argument is mandatory.

DESCRIPTION

The **remove_isolation_cell** command is used to remove isolation cell or enabled level shifters. This command can be used only on uniquified design.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes all isolation cells with string ISO in their name.

```
prompt> remove_isolation_cell -obj [get_cells *ISO*]
```

SEE ALSO

```
insert_isolation_cell(2)
```

remove_keepout_margin

Removes keepout margin of specified type for the specified cells/lib cells in the design.

SYNTAX

```
int remove_keepout_margin  
[-type hard | soft]  
[-derived]  
object_list
```

Data Types

object_list list

ARGUMENTS

-type hard | soft

Specifies the type of keepout to be removed. The valid values are **hard** and **soft**. By default, keepouts of both soft and hard types are removed for the cells or library cells.

-derived

This option is provided to remove the user specified pin count based derived keepout margin values. User defined pin count based keepout margin values are set using the -tracks_per_macro_pin and/or -max_padding_per_macro and/or -min_padding_per_macro options of **set_keepout_margin** command.

object_list

Describes the list of cells or library cells for which keepouts are to be removed. The keepouts are deleted for all cells with FIXED_PLACEMENT or library cells in this list.

DESCRIPTION

This command removes keepouts for the specified cells/library cells. If you do not use the **-type** option, keepouts of both soft and hard type are deleted for the specified cells/library cells. If this command is run to delete a keepout when none exists for a cell/library cell, a warning message is printed out and no action is taken.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example runs the **remove_keepout_margin** command.

```

prompt> report_keepout_margin MY_CELL -type hard
Cell Name: MY_CELL
*****
    Hard Keepout - 10.0000 10.0000 10.0000 10.0000
prompt> remove_keepout_margin -type hard MY_CELL
prompt> report_keepout_margin MY_CELL -type soft
Cell Name: MY_CELL
*****
    Soft Keepout - 10.0000 10.0000 10.0000 10.0000
prompt> remove_keepout_margin -type hard MY_LIB_CELL
The following example will remove all derived keepouts in the design.
prompt> remove_keepout_margin -derived *S

```

SEE ALSO

```

get_attribute(2)
report_keepout_margin(2)
set_keepout_margin(2)

```


remove_left_right_filler_rule

removes left-right filler cell insertion rules.

SYNTAX

```
integer remove_left_right_filler_rule  
-lib_cell cell_list
```

Data Types

cell_list collection

ARGUMENTS

```
-lib_cell cell_list  
        cell_list must refer to a collection of cell references. The left-right  
        filler cell rule removal will be applied to the cells specified in this list.
```

DESCRIPTION

Use this command to remove the left-right filler rules of the current design. A left (right) filler rule inserts a left (right) filler cell to the immediate left (right) side of a standard cell. A left or right filler cell is often used to avoid certain design rule spacing problems.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes left right filler rule for library cell stdcellA and stdcellB

```
prompt> remove_left_right_filler_rule -lib_cell {stdcellA, stdcellB}
```

SEE ALSO

```
insert_stdcell_filler(2)  
set_left_right_filler_rule(2)  
report_left_right_filler_rule(2)
```

remove_level_shifters

Removes all of the level shifters from the design.

SYNTAX

```
integer remove_level_shifters  
[-force]
```

ARGUMENTS

-force

Removes level shifters even if they are set with the **dont_touch** attribute, or if they were instantiated in the original netlist.

DESCRIPTION

This command removes all level shifters from the design, except those that you explicitly marked as **dont_touch** and those that you instantiated in the original netlist. To remove level shifters that have these restrictions, use the **-force** option.

This command provides the capability to remove pre-existing or tool-inserted level shifters, allowing you to perform custom voltage adjustments by manually inserting different level-shifter cells.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command removes level shifters in a design:

```
prompt> remove_level_shifters
```

SEE ALSO

```
check_level_shifters(2)  
insert_level_shifters(2)  
set_level_shifter_strategy(2)  
set_level_shifter_threshold(2)
```

remove_license

Removes a licensed feature.

SYNTAX

```
status remove_license  
feature_list
```

Data Types

```
feature_list    list
```

ARGUMENTS

```
feature_list  
    Specifies the list of features to remove. If you specify more than one  
    feature, they must be enclosed in braces ({}).  
    By looking at your key file, you can determine all of the features licensed  
    at your site.
```

DESCRIPTION

Removes the specified licensed features from the features you currently are using.

The **list_licenses** command provides a list of the features that you currently are using.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

This example removes the multivoltage license.

```
prompt> remove_license {Galaxy-MV}
```

SEE ALSO

```
check_license(2)  
license_users(2)  
list_licenses(2)  
get_license(2)
```

remove_map_power_switch

Removes library cells from the mapping definition for the specified power switch, which was previously defined by using the **map_power_switch** command. This command is supported only in UPF mode.

SYNTAX

```
status remove_map_power_switch  
power_switch_name  
-all | -lib_cells cells
```

Data Types

<i>power_switch_name</i>	string
<i>cells</i>	collection

ARGUMENTS

power_switch_name
Specifies the name of the power switch whose mapped library cells are to be removed.

-all
Removes the mapping definition for the specified power switch. This option is mutually exclusive with the **-lib_cells** option; you must specify one of these options.

-lib_cells *cells*
Specifies the library cells to be removed from the mapping definition for the specified power switch. This option is mutually exclusive with the **-all** option; you must specify one of these options.

DESCRIPTION

The *remove_map_power_switch* command removes library cells from the mapping definition for the specified power switch, which was previously defined by using the *map_power_switch* command.

This command returns a 1 if successful and returns a 0 otherwise.

EXAMPLES

The following example removes all the library cells from the mapping definition for the SW1 power switch.

```
prompt> remove_map_power_switch SW1 -all 1
```

The following example removes the lib2A/switch2A library cell from the SW1 power switch.

```
prompt>remove_map_power_switch SW1 -lib_cells lib2A/switch2A 1
```

SEE ALSO

```
create_power_switch(2)  
get_power_switches(2)  
map_power_switch(2)  
report_power_switch(2)
```

remove_mim_property

Removes multiply-instantiated-module data for selected instances. These instances will no longer be treated as multiply-instantiated-modules; that is, each will now refer to a unique master.

SYNTAX

```
status remove_mim_property  
{collection_of_cells}
```

Data Types

```
collection_of_cells    collection
```

ARGUMENTS

```
{collection_of_cells}  
    Specifies a list of hierarchical cells under the current parent cell for which  
    multiply-instantiated-module data is to be removed. If the command does not  
    find a cell, it issues a warning.
```

DESCRIPTION

This command deletes any data pointing to a multiply-instantiated-module master for each cell in the specified collection. The cell becomes unique.

This command affects the database. The tool saves the changes it makes if the parent cell is saved.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example first uniquifies the cells by using the **uniquify_fp_mw_cell** command and stores a reference to the master cell for instA, instB, instC and instD. The **report_mim** command then lists these instances as multiply-instantiated modules. The **remove_mim_property** command alters the multiply-instantiated module status of instA and instB. They are now treated as uniquified and using the **report_mim** command does not list them.

```
prompt> uniquify_fp_mw_cell\  
-store_mim_property [get_cells {instA instB instC instD}]  
prompt> report_mim  
instA  
instB  
instC
```

```
remove_mim_property  
1544
```

```
instD  
prompt> remove_mim_property [get_cells {instA instB}]  
prompt> report_mim
```

SEE ALSO

```
report_mim(2)  
uniquify_fp_mw_cel(2)
```

remove_mw_cel

Removes Milkyway designs from the design library.

SYNTAX

```
status remove_mw_cel
[-hierarchy [-check_only]]
[-version_kept count]
[-verbose]
[-all_versions]
[-all_view]
mw_cel_list
```

Data Types

<i>count</i>	integer
<i>mw_cel_list</i>	list

ARGUMENTS

-hierarchy

Deletes both the specified top-level design and all of its subdesigns. By default, the command deletes only the specified designs and retains subdesigns

This option can not work with **-all_views** and **-all_versions**. If **mw_cel_list** option is used with this option, it can only contain one Milkyway design. Moreover, the version number and view name of the Milkyway design will be ignored. As for the view name, the hierarchical remove can only operate on CEL view. As for the version number, hierarchical remove can only remove all versions or retain last version, which is controlled by **-version_kept** option.

-check_only

Prints the list of Milkyway designs, versions, and views, but does not delete any data. If you are unsure about the result of the command, run this option first, before performing the purge operation. This option is valid only when used with the **-hierarchy** option.

-version_kept *count*

Specifies how many recent versions to keep. The value must be equal or greater than 0. Value 0 removes all versions. When used with the **-hierarchy** option, the only valid values are 0 or 1.

The default behavior differs for hierarchical removal (specified by using the **-hierarchy** option) and nonhierarchical removal. For hierarchical removal, all versions are deleted if you do not specify the **-version_kept** option. For nonhierarchical removal, only the latest version is deleted if you do not specify the **-version_kept** option.

-verbose

Prints the possible cause when the command fails to remove a specified Milkyway design. Currently, this option is effective only when you use it with the **-hierarchy** option.

-all_versions

Removes all versions of specified Milkyway designs. This option can not work with **-hierarchy** option.

-all_view

Removes all views of specified Milkyway designs. By default, the command removes only the specified view. This option can not work with **-hierarchy** option.

mw_cel_list

Specifies the Milkyway designs to remove. If you also use the **-hierarchy** option with this option, the designs to remove must be top-level designs. You can specify a Milkyway design by name, pattern, or collection. For more information, see the EXAMPLES section, below.

When use with **-hierarchy** option, only one Milkyway design can be specified. And the version number or view name of the Milkyway design will be ignored. The view name is always CEL view and the version number is controlled by **-version_kept** option.

When not use with **-hierarchy** option, only the specified view and version will be deleted.

DESCRIPTION

This command removes specified Milkyway designs from the current library. Before removing designs, you must make sure that the designs are closed.

The **-version_kept** option allows you to remove older versions of the Milkyway design while keeping newer ones (to free up disk space).

The **-hierarchy** option allows you to remove or purge a Milkyway design and its whole hierarchical tree.

NOTE: Please be aware that the behavior of **remove_mw_cel** differs depending on whether you specify the **-hierarchy** option. The default command behavior deletes the specified version of the Milkyway design, but when called with the **-hierarchy** option, the command deletes all versions or non-latest versions of the Milkyway designs controlled by **-version_kept**.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes the latest version of the Milkyway designs whose names start with *test*.

```
prompt> get_mw_cels test*
{"test1", "test2"}
prompt> remove_mw_cel test* -verbose
Removed mw_cel test1.
```

```
Removed mw_cel test2.  
1
```

The following example removes version **1** of a design named *test1*. In this case, *test1* has two versions. After removing one version, you can still get a Milkyway design collection based on its remaining version.

```
prompt> remove_mw_cel "test1.CEL;1" -verbose  
Removed mw_cel test1.CEL;1.  
1  
prompt> get_mw_cels *  
{"test1"}
```

The following example removes the design named *chip_top* and all of its subdesigns from the current design library.

```
prompt> remove_mw_cel -hierarchy chip_top  
Deleting the cell: nand_macro ...  
Deleting the cell: chip_top ...  
1
```

The following example purges the old versions of the design named *chip_top* and all of its subdesigns from the current design library. Only the latest version of these designs is kept.

```
prompt> remove_mw_cel -hierarchy -version_kept 1 \  
chip_top  
Purging the cell: nand_macro ...  
Purging the cell: chip_top ...  
1
```

SEE ALSO

```
close_mw_cel(2)  
copy_mw_cel(2)  
create_mw_cel(2)  
current_mw_cel(2)  
get_mw_cels(2)  
open_mw_cel(2)  
rename_mw_cel(2)  
save_mw_cel(2)
```

remove_net

Removes nets from the *current design*.

SYNTAX

```
int remove_net  
net_list | -all
```

Data Types

net_list list

ARGUMENTS

net_list

A list of nets to be removed from the current design. Each net name must exist in the current design. You must specify either *net_list* or **-all**.

-all

Causes the removal of all nets in the current design.

DESCRIPTION

This command removes nets or net instances from the *current design*. Net connections to pins or ports are disconnected.

You cannot remove bused nets with the **remove_net** command.

To create nets, use the **create_net** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example in uses **remove_net** in the current design:

```
prompt> get_nets ""  
{NET0 NET1 NET2 NET3 MY_CHECK PARITY}  
  
prompt> remove_net "N*"  
Removing net 'NET0' in design 'my_design'.  
Removing net 'NET1' in design 'my_design'.  
Removing net 'NET2' in design 'my_design'.  
Removing net 'NET3' in design 'my_design'.  
  
prompt> get_nets ""  
{MY_CHECK PARITY}
```

The following example removes all remaining nets in the current design:

```
prompt> remove_net -all  
Removing net 'MY_CHECK' in design 'my_design'.  
Removing net 'PARITY' in design 'my_design'.  
  
prompt> get_nets ""  
{}
```

The following example removes a net instance:

```
prompt> remove_net {U1/MY_NET U2/MY_NET}  
Removing net 'U1' in design 'foo_1'.  
Removing net 'U2' in design 'foo_2'.
```

SEE ALSO

```
create_net(2)  
current_design(2)
```

remove_net_routing

Removes all routing (vias and segments) for specific nets.

SYNTAX

```
int remove_net_routing  
list_of_nets
```

Data Types

```
list_of_nets      list
```

ARGUMENTS

```
list_of_nets  
    Lists the net names for which to remove routing. Substitute the list you want  
    for list_of_nets.
```

DESCRIPTION

This command removes all routing (vias and segments) for specific nets.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes routing for nets named *sig1* and *sig2*.

```
prompt> remove_net_routing {sig1 sig2}
```

SEE ALSO

```
set_net_routing_rule(2)
```

remove_net_routing_layer_constraints

Removes routing layer constraints for specific nets.

SYNTAX

```
int remove_net_routing_layer_constraints  
list_of_nets
```

Data Types

```
list_of_nets      list
```

ARGUMENTS

```
list_of_nets  
    Lists the net names for which to remove the routing layer constraints.  
    Substitute the list you want for list_of_nets.
```

DESCRIPTION

This command removes routing layer constraints for specific nets.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes the routing layer constraints for a net named *netA*.

```
prompt> remove_net_routing_layer_constraints {netA} \
```

SEE ALSO

```
report_net_routing_layer_constraints(2)  
set_net_routing_layer_constraints(2)
```

remove_net_shape

Removes objects of net shapes. The objects could be specified by a collection or by name.

SYNTAX

```
status_value remove_net_shape  
[-verbose]  
net_shapes
```

Data Types

net_shapes list

ARGUMENTS

-verbose
 Prints additional messages.

net_shapes
 Specifies a nonempty collection of handles to net shapes, or the name of the net shape.

DESCRIPTION

This command removes all specified net shapes, and then returns 1 if successful, or 0 if it fails.

If invalid handles occur in the input handle collection, the command line interpreter removes them from the collection and invokes the **remove_net_shape** command with the updated collection.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes net shapes:

```
prompt> set a [get_net_shapes -of_objects n300  
      {"VWIRE#6682", "HWIRE#7196", "VWIRE#6683"}  
  
prompt> remove_net_shape $a  
1
```

The following example removes net shapes specified by a pattern list:

```
prompt> remove_net_shape {HWIRE#7170 HWIRE#7171} -verbose  
Removed net shape HWIRE#7170.  
Removed net shape HWIRE#7171.  
1
```

SEE ALSO

```
create_net_shape(2)  
get_net_shapes(2)  
remove_user_shape(2)
```


remove_net_timing_spacing

Remove timing spacing flags.

SYNTAX

status **remove_net_timing_spacing**

DESCRIPTION

This command removes timing spacing flags for all nets.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

```
prompt> remove_net_timing_spacing
```

remove_objects

Removes a list of objects

SYNTAX

```
status remove_objects  
objects
```

ARGUMENTS

```
objects  
    Specifies the objects to be removed.
```

DESCRIPTION

Removes a list of objects from the current design.

Note: only non-netlist objects can be removed i.e objects which have representations in the netlist cannot be removed.

See `get_edit_property(2)` for a list of deletable objects.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes the selected objects:

```
> remove_objects [get_selection]
```

SEE ALSO

remove_output_delay

Removes output delay on pins or output ports.

SYNTAX

```
int remove_output_delay  
[-clock clock [-clock_fall] [-level_sensitive]]  
[-rise]  
[-fall]  
[-max]  
[-min]  
port_pin_list
```

Data Types

<i>clock</i>	string or collection of one object
<i>port_pin_list</i>	list

ARGUMENTS

-clock *clock*
Removes delay information relative to the specified clock edge. If **-clock** is not specified, all output delay is removed. To remove output delay that has no relative clock, use **remove_output_delay -clock ""**.
You can specify the clock as either a string or a collection of one object.

-clock_fall
Removes output delay relative to the clock falling edge. If you do not specify **-clock_fall**, output delay relative to the clock rising edge is removed.
This option is only used with the **-clock** option.

-level_sensitive
Removes level-sensitive output delay for the specified clock. If this option is not used, non-level-sensitive output delay is removed.

-rise
Removes rising delay on *port_pin_list*. If you do not specify **-rise** or **-fall**, both rising and falling delays are removed.

-fall
Removes falling delay on *port_pin_list*. If you do not specify **-rise** or **-fall**, both rising and falling delays are removed.

-max
Removes maximum output delays on *port_pin_list*. If you do not specify **-max** or **-min**, both maximum and minimum output delays are removed.

-min
Removes minimum output delays on *port_pin_list*. If you do not specify **-max** or **-min**, both maximum and minimum output delays are removed.

`port_pin_list`

Specifies a list of port or pin names in the current design. Output delay is removed from each object in `port_pin_list`. To specify more than one object, enclose the objects in quotes (") or braces ({}).

DESCRIPTION

This command removes output delay values for objects in the current design. Output delay is set by the **set_output_delay** or the **characterize** command. By default, all output delay on each object in `port_pin_list` is removed. To restrict the removed output delay values, use the **-clock**, **-clock_fall**, **-min**, **-max**, **-rise**, or **-fall** option.

To list output delays associated with ports, use the **report_port** command.

To list output delays of internal pins, use the **report_design** command.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example removes all output delay values from all output ports in the current design:

```
prompt> remove_output_delay [all_outputs]
```

This example removes output maximum rise delay values from OUT1, OUT3, and BIDIR12.

```
prompt> remove_output_delay -max -rise {OUT1 OUT3 BIDIR12}
```

This example removes all output delay for port OUT1, relative to the falling edge of CLK2:

```
prompt> remove_output_delay -clock [get_clocks CLK2] \  
-clock_fall [get_ports OUT1]
```

This example removes all output delay not relative to any clock for the OUT2 port:

```
prompt> remove_output_delay -clock "" OUT2
```

SEE ALSO

`all_outputs(2)`
`current_design(2)`
`remove_input_delay(2)`
`report_design(2)`
`report_port(2)`
`reset_design(2)`
`set_input_delay(2)`
`set_output_delay(2)`

`remove_output_delay`

remove_partition

Removes partitions created by **read_partition** command.

SYNTAX

status **remove_partition**

ARGUMENTS

The **remove_partition** command has no arguments.

DESCRIPTION

This command allows you to remove partitions previously created by **read_partition** command.

EXAMPLES

This example shows creating and removing a partition.

```
prompt> read_partition -input_files {myDesign.v}
Partition Name: pfe0
Loading verilog file myDesign.v
MYDESIGN: 73958 instances of 134 modules.
1
prompt> remove_partition
Info: Removing partition pfe0
1
```

SEE ALSO

[read_partition\(2\)](#)

remove_pg_network

Removes or changes the hierarchical power and ground network.

SYNTAX

```
status remove_pg_network  
-net list_of_pg_net  
hier_cell_list | -top
```

Data Types

hier_cell_list collection

ARGUMENTS

-net list_of_pg_net
Specifies an existing power or ground net for network removal. You can specify only one net, which must be a scalar (single bit) net.
The net, specified as its full name, can be at any hierarchy level.
If you do not specify this option, the tool issues an error message and stops.

hier_cell_list
Specifies a list of hierarchical cell instances from which to remove the power and ground network. The tool removes the power and ground network from the specified hierarchical cells and their children.
If any of the specified cells does not exist in the current design or is not a hierarchical cell, the tool issues an error message and stops.
If any of the specified cells exists in the current design and is a hierarchical cell, but is not in the PG network, the tool issues a warning message and continues.

-top
If this option is specified, the tool ignores *hier_cell_list* option and the operation will be performed on the entire design.

DESCRIPTION

This command removes hierarchical PG connections from the specified hierarchical cell instances and their children or from the entire design if you do not specify any hierarchical cell instances.

For each specified hierarchical cell and their children, the command removes the power and ground network by performing the following tasks:

1. Disconnects the PG nets from the leaf pins inside the cell
2. Deletes the PG nets that connect to pins inside the cell
3. Disconnects the PG nets from the ports of the cell
4. Deletes the cell ports connected to the power and ground network

The removed hierarchy cell ports can be recreated by `create_pg_network` command.

This command supports the recovery of tie connections. All signal pins connected on the network are re-connected to tie before the PG network is removed.

This command does not remove top-level PG port. To remove the top-level PG port, you must use the **remove_port** command.

EXAMPLES

The following example uses the **remove_pg_network** command to remove the existing VDD connection from the specified hierarchical cell instances. The **all_connected** command is used to verify the Milkyway net connections before and after running **remove_pg_network**.

```
prompt> all_connected [get_pin -all m2/VDD]
{VDD}
prompt> remove_pg_network -net VDD "m2/m21 m1/m11"
prompt> all_connected [get_pin -all m2/VDD]
Warning: No pin objects matched 'm2/VDD' (SEL-004)
```

The following example uses the **remove_pg_network** command to remove the VDDC net connection from the specified hierarchical cell. The **all_connected** command is used to verify the net connections before and after running **remove_pg_network**.

```
prompt> all_connected [get_pin -all m2/VDDC]
{VDDC}
prompt> remove_pg_network -net VDDC "m2"
prompt> all_connected [get_pin -all m2/VDDC]
Warning: No pin objects matched 'm2/VDDC' (SEL-004)
```

SEE ALSO

```
create_pg_network(2)
recover_tie_connection(2)
create_net(2)
current_design(2)
connect_net(2)
disconnect_net(2)
remove_net(2)
remove_port(2)
```

remove_physical_bus

Removes physical buses.

SYNTAX

```
status remove_physical_bus  
physical_bus_list | -all
```

Data Types

```
physical_bus_list      list
```

ARGUMENTS

physical_bus_list
Specifies the physical buses to remove. The *physical_bus_list* value can include collections of *physical_bus* type, patterns that use wildcards, or certain command names. The *physical_bus_list* and **-all** options are mutually exclusive.

-all
Specifies to remove all physical buses in the current design. The *physical_bus_list* and **-all** options are mutually exclusive.

DESCRIPTION

This command removes all specified physical buses. You can specify physical buses by using their name patterns or collections of *physical_bus* type which can be generated by commands like **get_physical_bus** and **get_selection**.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows the use of a wildcard pattern that removes all physical buses.

```
prompt> remove_physical_bus *  
1
```

SEE ALSO

```
create_physical_bus(2)  
get_physical_buses(2)  
get_selection(2)  
report_physical_bus(2)
```

remove_physical_bus
1562


```
update_physical_bus(2)
```

remove_pin_guides

Deletes the specified pin guides from the design.

SYNTAX

```
status remove_pin_guides  
-all | patterns  
[-verbose]
```

Data Types

patterns collection

ARGUMENTS

-all
Removes all pin guides.

patterns
Specifies the collection of pin guides to delete.

-verbose
Turns on verbose output.
If -all is specified the total number of pin guides deleted is output;
otherwise, the names of the deleted pin guides are output.

DESCRIPTION

This command deletes the specified pin guides from the design.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes all pin guides.

```
prompt> remove_pin_guides -all
```

The following example removes selected pin guides.

```
prompt> remove_pin_guides [get_selection]
```

SEE ALSO

get_pin_guides(2)

remove_pin_guides
1564

report_pin_guides(2)

remove_pin_name_synonym

Removes pin name synonym definitions.

SYNTAX

```
status remove_pin_name_synonym
[-all]
[synonym_list]
```

Data Types

synonym_list list

ARGUMENTS

-all

Removes all the pin name synonym definitions. One of **-all** and **synonym_list** is required, and they are mutually exclusive.

synonym_list

Removes pin name synonym definitions that match one of the pin name synonym strings in the list. One of the **-all** and **synonym_list** is required, and they are mutually exclusive.

DESCRIPTION

The **remove_pin_name_synonym** command deletes some or all pin name synonym definitions.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes pin name synonyms.

```
prompt> set_pin_name_synonym SYN_CD CD
1
prompt> set_pin_name_synonym SYN_QN QN
1
prompt> set_pin_name_synonym -full_name mid1/bot1/LSR0P_at_bot/SYN_R mid1/bot1/
LSR0P_at_bot/R
1
prompt> set_pin_name_synonym -full_name this/is/a/synonym mid1/FJK2SP_at_mid/TI
1
prompt> set_pin_name_synonym -full_name mid1/FJK2SP_at_mid/J mid2/bot2/LSR0P_at_bot/
Q
```

remove_pin_name_synonym
1566

```

1
prompt> report_pin_name_synonym

*****
Report : pin name synonym
Design : top
Version: X-2005.09
Date   : Wed Jun 29 10:29:04 2005
*****

```

Synonym		Pin Name
Type		

this/is/a/synonym		mid1/FJK2SP_at_mid/
TI	full name	
mid1/FJK2SP_at_mid/J		mid2/bot2/LSR0P_at_bot/
Q	full name	
mid1/bot1/LSR0P_at_bot/SYN_R		mid1/bot1/LSR0P_at_bot/
R	full name	
SYN_QN		QN
	simple name	
SYN_CD		CD
	simple name	

```

1
prompt> remove_pin_name_synonym SYN_QN
1
prompt> remove_pin_name_synonym this/is/a/synonym
1
prompt> report_pin_name_synonym

```

```

*****
Report : pin name synonym
Design : top
Version: X-2005.09
Date   : Wed Jun 29 10:29:04 2005
*****

```

Synonym		Pin Name
Type		

mid1/FJK2SP_at_mid/J		mid2/bot2/LSR0P_at_bot/
Q	full name	
mid1/bot1/LSR0P_at_bot/SYN_R		mid1/bot1/LSR0P_at_bot/
R	full name	
SYN_CD		CD
	simple name	

```

1
prompt> remove_pin_name_synonym -all
1

```

```
prompt> report_pin_name_synonym
```

```
*****  
Report : pin name synonym  
Design : top  
Version: X-2005.09  
Date   : Wed Jun 29 10:29:04 2005  
*****
```

```
No pin name synonym  
1
```

SEE ALSO

```
set_pin_name_synonym(2)  
report_pin_name_synonym(2)
```

remove_placement

Unplace cells in the core area.

SYNTAX

```
status remove_placement
[-object_type {standard_cell | macro_cell | all}]
[-new_location {top_right | origin | center}]
```

ARGUMENTS

-object_type {standard_cell | macro_cell | all}
Unplaces macro cells, standard cells, or all of them. Default is to unplace all types of cells.

-new_location {top_right | origin | center}
The location where tool moves the cells. Default is top_right, unplaced macro cells are on the top side of chip, and unplaced standard cells are on the right side of chip. If "origin" is specified, unplaced standard cells coordinates are set to (0,0). If "center" is specified, unplaced standard cells coordinates are set to the center of the chip.

DESCRIPTION

Unplace cells in the core area. The command does not unplace fixed & dont_touch cells.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example moves all standard cells outside the chip at the right boundary. Macros are not moved.

```
prompt> remove_placement -object_type standard_cell
1
```

SEE ALSO

remove_placement_blockage

Remove placement blockages.

SYNTAX

```
int remove_placement_blockage  
[-verbose]  
[-name name]  
patterns | -all
```

Data Types

patterns list

ARGUMENTS

- verbose**
Prints more messages.
- name *name***
Specifies a name by which this command finds and removes a placement blockage. Notice that the name should be a real name specified by you when creating the same placement blockage with the `create_placement_blockage -name` command, not a run time name, such as "pb#7789". The "-name", "patterns" and "-all" are mutually exclusive.
- patterns**
Specifies the placement blockages to remove. The patterns can be a collection handle of blockage, or other formats, such as the following:
 * or pb#* Means all placement blockages.
 pb#7789 Means one placement blockage whose object_id is 7789.
The "patterns", "-all" and "-name" are mutually exclusive.
- all**
If specified, this command removes all placement blockages in the current design. The "-all", "patterns" and "-name" are mutually exclusive.

DESCRIPTION

This command removes all specified placement blockages.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example create a blockage for placement.

```
remove_placement_blockage  
1570
```



```
prompt> remove_placement_blockage -all  
1
```

```
prompt> remove_placement_blockage [get_placement_blockages \  
? * -within {{2 2} {25 25}}  
1
```

SEE ALSO

```
create_route_guide(2)  
get_placement_blockages(2)  
get_route_guides(2)  
remove_route_guide(2)
```

remove_plan_groups

Removes plan groups from the current design.

SYNTAX

```
int remove_plan_groups  
[-verbose]  
-all objects
```

Data Types

objects string

ARGUMENTS

-verbose
 Specifies to print information about what are being deleting.

-all
 Specifies to remove all plan groups from the current design.

objects
 Specifies plan group objects to be removed. You can use the *get_plan_groups* command to specify the objects.

DESCRIPTION

This command removes plan groups from the current design. If you specify the **-all** option, all plan groups are to be removed from the design.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes plan groups named *timer/decode0* and *timer/decode1*.

```
prompt> remove_plan_groups -verbose {timer/decode0 timer/decode1  
prompt> remove_plan_groups [get_plan_groups *
```

SEE ALSO

get_plan_groups(2)

remove_pnet_options

Removes options set by the **set_pnet_options** command.

SYNTAX

```
int remove_pnet_options
```

ARGUMENTS

The **remove_pnet_options** command has no arguments.

DESCRIPTION

This command removes the options set on the metal layers and the design using the **set_pnet_options** command. The pnet options are removed from the design and are not set to defaults. For more details about defaults, see the man page for the **set_pnet_options** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example prints out the different options set on the metal layers with the **set_pnet_options** command.

```
prompt> remove_pnet_options
1
```

SEE ALSO

```
legalize_placement(2)
report_pnet_options(2)
set_pnet_options(2)
```

remove_port

Removes ports from the current design or its subdesign.

SYNTAX

```
int remove_port  
port_list
```

Data Types

port_list list

ARGUMENTS

port_list
Specifies a list of ports to be removed from the current design. Each port name must exist in the current design.

DESCRIPTION

Removes ports from the current design or its subdesign.

Scalar (single bit) ports that are components of a bused port cannot be removed using this command.

To create ports, use **create_port**.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses **remove_port** to remove ports from the current design.

```
prompt> get_ports ""  
{A1 A2, B1, B2, OUT1}  
  
prompt> remove_port "B*"  
Removing port 'B1' in design 'my_design'.  
Removing port 'B2' in design 'my_design'.  
1  
prompt> get_ports ""  
{A1 A2 OUT1}
```

In the following example, ports are removed from the subdesign MID. U1 is an instance of the design MID.

```
prompt> remove_port [get_ports U1/p1 U1/p2]  
Removing port 'U1/p1' in design 'MID'.  
Removing port 'U1/p2' in design 'my_design'.  
1
```

SEE ALSO

```
create_port(2)  
current_design(2)
```

remove_power_domain

Removes the specified power domains.

SYNTAX for UPF mode

status **remove_power_domain**

[-verbose]

[-scope *instance_name*]

[*power_domains*]

Data Types for UPF mode

<i>instance_name</i>	string
<i>power_domains</i>	list

SYNTAX for Non-UPF Mode

status **remove_power_domain**

domain_name | -all

Data Types for Non-UPF mode

<i>domain_name</i>	string

ARGUMENTS

-verbose

This option is available only in UPF mode. Prints additional messages.

-scope *instance_name*

This option is available only in UPF mode. Specifies the scope in which the power domain is to be deleted. The instance name is the name of a hierarchical cell.

By default, the power domain is deleted from the current scope.

If this option is specified, the **power_domains** option must also be specified and it only accepts simple name, not collection.

domain_name

This option is available only in non-UPF mode. Specifies the name of the power domain to be deleted. The name should be a simple (non-hierarchical) name. This option is mutually exclusive with the **-all** option.

If the specified power domain does not exist in the current design, the command fails.

If there are any supply ports, supply nets, or power switches associated with the specified power domain, the command fails.

power_domains

This option is available only in UPF mode. Specify the power domains to be

remove_power_domain

deleted. The option value could be a collection of power domains or name patterns.
The command fails if it could not find any specified power domain.
If this option is not specified, the command will remove all power domains under current scope.
If **-scope** option is also specified, the value of this option must be a simple name, not collection.
If there are any supply ports, supply nets, or power switches associated with the specified power domain, the command fails.

-all

This option is available only in non-UPF mode. Deletes all power domains in the current design. This option is mutually exclusive with the **domain_name** argument.

DESCRIPTION

The **remove_power_domain** command enables you to remove existing power domains (or all power domains, in non-UPF mode). Before you can delete a power domain, you must remove all relationships between the power domain and any supply ports, supply nets, power switches, or child power domains.

This command returns a 1 if successful and returns a 0 otherwise.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following UPF-mode example removes the PD1 power domain in current scope.

```
prompt> remove_power_domain PD1
1
```

The following UPF-mode example removes power domains whose name start with PD in all scopes:

```
prompt> remove_power_domain [get_power_domains -hier PD*] -verbose
Removed power domain 'PD1' successfully.
Removed power domain 'REG_BLK/PD2' successfully.
Removed power domain 'STACK_BLK/PD_child' successfully.
1
```

The following non-UPF mode example creates the TOP_DOMAIN power domain and then deletes it.

```
prompt> create_power_domain TOP_DOMAIN
1
prompt> create_power_domain SUB_DOMAIN -power_down \
        -power_down_ctrl [get_nets pd_ctrl] \
        -power_down_ack [get_nets pd_ack] \
```

```

1                                -object_list [get_cells mid1]
prompt> get_power_domains
{TOP_DOMAIN SUB_DOMAIN}
prompt> remove_power_domain TOP_DOMAIN
Error: Design-level power domain TOP_DOMAIN cannot be removed until
all instance power domains are removed. (MV-074)
0
prompt> remove_power_domain SUB_DOMAIN
Removing Power Domain 'SUB_DOMAIN'
1
prompt> remove_power_domain TOP_DOMAIN
Removing Power Domain 'TOP_DOMAIN'
1

```

SEE ALSO

```

connect_power_domain(2)
create_power_domain(2)
get_power_domains(2)
report_power_domain(2)
set_scope(2)

```


remove_power_net_info

Deletes power net info.

SYNTAX

```
int remove_power_net_info  
name  
-all
```

Data Types

name string

ARGUMENTS

name The name of the power net info to be deleted. Mutually exclusive with -all option.

-all Deletes all the power nets in the current design. Mutually exclusive with *name* option.

DESCRIPTION

The **remove_power_net_info** command deletes a named power net info or all the power net info for the current design.

EXAMPLES

The following examples use the command to create and remove power nets.

```
prompt> create_power_net_info VSS_net -gnd  
1  
prompt> create_power_net_info VDD_net -power  
1  
prompt> remove_power_net_info VSS_net  
1  
prompt> remove_power_net_info -all  
1
```

SEE ALSO

report_power_net_info (2)
create_power_net_info (2)

remove_power_switch

Removes a power switch from the specified power domain. This command is supported only in UPF mode.

SYNTAX

```
status remove_power_switch
[-verbose]
[power_switches]
[-domain domain_name]
```

Data Types

<i>power_switches</i>	list
<i>domain_name</i>	string

ARGUMENTS

-verbose
Prints additional messages.

-domain *domain_name*
Specifies the power domain of the power switch to be deleted.
If the specified power domain does not exist in the current scope, the command fails.
If this option is specified, the *power_switches* option must be a simple name, not collection.

DESCRIPTION

The *remove_power_switch* command removes existing power switches.

This command returns a 1 if successful and returns a 0 otherwise.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes the SW1 power switch.

```
prompt> remove_power_switch SW1 -verbose
Removed power switch 'SW1' from power domain 'PD1' successfully.
1
```

SEE ALSO

[create_power_switch\(2\)](#)

[remove_power_switch](#)
1580

```
get_power_switches(2)
report_power_switch(2)
set_scope(2)
```

remove_preferred_routing_direction

Removes the preferred routing direction for the given routing layer(s).

SYNTAX

```
string remove_preferred_routing_direction  
-layers list_of_layers
```

ARGUMENTS

```
-layers list_of_layers  
    Specify the list/collection of layer(s) for which the preferred routing  
    directions is to be removed.
```

DESCRIPTION

This command removes the preferred routing direction for the routing layer(s) specified from the design & the preferred routing direction for those routing layers defaults to the one specified in the reference library.

Note: This command will issue warning for layers that do not have a defined preferred routing direction in the reference library, as there needs to exist atleast one preferred routing direction setting for each layer.

Note: This command will issue suitable warnings whenever adjacent layers have the same routing direction after the execution of the command.

Note: This command removes the setting from the design, & NOT the reference library. The change is to the persistent data and will be reflected in the design database whenever the user saves the design.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes the preferred routing direction for layers "m1" & "m2"

```
prompt> remove_preferred_routing_direction -layers {m5 m6}
```

SEE ALSO

```
report_preferred_routing_direction(2)  
set_preferred_routing_direction(2)
```

remove_propagated_clock

Removes a propagated clock specification.

SYNTAX

```
string remove_propagated_clock  
object_list
```

Data Types

object_list list

ARGUMENTS

object_list
 Lists clocks, ports, pins, or cells.

DESCRIPTION

Removes the propagated clock specification from clocks, ports, pins, or cells in the current design. The **set_propagated_clock** command specifies that delays be propagated through the clock network to determine latency at register clock pins. If this is not specified, ideal clocking is assumed. Ideal clocking means clock networks have a user-specified latency (from the **set_clock_latency** command), or zero latency by default. Propagated clock latency is normally used for post layout, after final clock tree generation. Ideal clock latency provides an estimate of the clock tree for pre-layout.

To specify an ideal latency, you can also use **set_clock_latency**; this overrides the propagated clock specification for an object.

To see **propogated_clock** attributes on clocks, use **report_clock -skew**.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The example removes propagated clock specifications from all clocks in the design.

```
prompt> remove_propagated_clock all_clocks[]
```

SEE ALSO

```
set_clock_latency(2)  
set_propagated_clock(2)  
create_clock(2)  
remove_attribute(2)
```

```
reset_design(2)  
report_clock(2)
```

remove_qor_snapshot

Removes an existing QoR snapshot of timing, drc, area, power, etc. The remove utility deletes snapshots from the location specified by the **icc_snapshot_storage_location** variable.

NOTE: This is the new version of the **remove_qor_snapshot** command that supports both multicorner-multimode designs as well as non-multicorner-multimode designs.

SYNTAX

```
void remove_qor_snapshot  
[-name name]  
[-all name]
```

ARGUMENTS

-name *name*
The name of the qor snapshot to be removed.

-all *name*
Remove all snapshots for this design.

DESCRIPTION

This command removes one or more qor snapshot(s).

Multicorner-Multimode Support

This command uses information from active scenarios only.

EXAMPLES

In the following example, **remove_qor_snapshot** will remove QoR snapshot named *preroute* which is saved under *design/snapshot* directory.

```
prompt> remove_qor_snapshot -name preroute
```

SEE ALSO

```
create_qor_snapshot(2)  
report_qor_snapshot(2)
```

remove_rail_maps

Removes all rail map data from the current session.

SYNTAX

```
status remove_rail_maps
```

ARGUMENTS

This command has no arguments.

DESCRIPTION

This command removes all rail map data from the current IC Compiler session, releasing any memory previously allocated for the maps. This command can be helpful in reducing the memory in use when you no longer need to see the rail maps. Use the **read_rail_maps** command to reload the rail maps, if needed.

EXAMPLES

The following example shows how to remove a rail map from memory.

```
prompt> remove_rail_maps
```

SEE ALSO

```
analyze_rail(2)  
read_rail_maps(2)
```


remove_route_by_type

Remove route by type.

SYNTAX

```
status remove_route_by_type
[-nets nets]
[-signal_detail_route]
[-signal_user]
[-clock_ring]
[-clock_strap]
[-clock_tie_off]
[-clock_user]
[-pg_ring]
[-pg_strap]
[-pg_tie_off]
[-pg_user]
[-pg_std_cell_pin_conn]
[-pg_macro_io_pin_conn]
[-keep_pg_vias]
[-keep_pg_pins_at_boundary]
[-bus]
[-shield]
[-keep_frozen_net]
```

Data Types

nets string

ARGUMENTS

-nets *nets*
Remove route by type for a collection of nets. The value can be a string or a list of strings.

-signal_detail_route
Remove signal routes that are of type detail wire.

-signal_user
Remove signal routes that are of type user.

-clock_ring
Remove clock ring.

-clock_strap
Remove clock strap.

-clock_tie_off
Remove clock tie off detail route.

-clock_user
Remove clock user entered.

-pg_ring
Remove pg ring.

-pg_strap
Remove pg strap.

-pg_tie_off
Remove pg tie off.

-pg_user
Remove pg user entered.

-pg_std_cell_pin_conn
Remove pg std cell connections.

-pg_macro_io_pin_conn
Remove pg macro/io pin connections.

-keep_pg_vias
Don't remove pg vias.

-keep_pg_pins_at_boundary
Keep pg boundary pins generated by preroute std cell.

-bus
Remove bus type.

-shield
Remove shield type.

-keep_frozen_net
Don't remove frozen net.

DESCRIPTION

This command deletes routes of specified types of all nets or that of specified net names.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

```
prompt> remove_route_by_type -signal_detail_route
```

remove_route_guide

Removes route guides.

SYNTAX

```
int remove_route_guide  
[-verbose]  
[-name name]  
patterns | -all
```

Data Types

patterns list

ARGUMENTS

-verbose
 Prints additional messages.

-name *name*
 Specifies a name by which this command finds and removes a route guide.
 Notice that the name should be a real name specified by you when creating the
 same route guide with the `create_route_guide -name` command, not a run time
 name, such as "rg#7689".
 The "-name" and "patterns" are mutually exclusive.

patterns
 Specifies the route guides to remove. The patterns can be a collection handle
 of route guides, or other formats, such as the following examples:
 * or *rg** removes all route guides.
 rg#7789 removes one route guide whose `object_id` is 7789.
 The "patterns" and "-name" options are mutually exclusive.

-all
 Specifies to remove all route guides in the *current design*.

DESCRIPTION

This command removes all specified route guides.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes the route guides specified by the `get_route_guides` command.

```
prompt> remove_route_guide [get_route_guides *] \  
      -within {{2 2} {25 25}}  
1
```

The following example removes all route guides.

```
prompt> remove_route_guide -all  
1
```

SEE ALSO

```
create_placement_blockage(2)  
create_route_guide(2)  
get_placement_blockages(2)  
get_route_guides(2)  
remove_placement_blockage(2)
```

remove_routing_blockage

Removes the specified routing blockages.

SYNTAX

```
status remove_routing_blockage
[-verbose]
patterns
```

Data Types

patterns list

ARGUMENTS

-verbose

Prints additional messages.

patterns

Specifies the routing blockages. You can specify the patterns by using the following formats:

- * A collection of routing blockages
 - * An asterisk (*), which indicates all routing blockages
 - * A string
- For example, RB_1234 matches the routing blockage named RB_1234

DESCRIPTION

This command removes the specified routing blockages.

EXAMPLES

The following example removes all routing blockages within the rectangle with the lower-left corner at {2 2} and the upper-right corner at {25 25}.

```
prompt> remove_routing_blockage \
[get_routing_blockages -within {{2 2} {25 25}}]
1
```

The following example removes all metal routing blockages.

```
prompt> remove_routing_blockage [get_routing_blockages -type metal]
1
```

SEE ALSO

```
create_routing_blockage(2)
get_routing_blockages(2)
```

remove_routing_rules

Removes nondefault routing rules in a design defined by the **define_routing_rule** command.

SYNTAX

```
int remove_routing_rules  
[-all]  
rule_name_list
```

Data Types

rule_name_list list

ARGUMENTS

-all
Removes all nondefault routing rules.

rule_name_list
Removes nondefault routing rules with the specified names.

DESCRIPTION

This command removes the nondefault routing rules defined in the design. If you use the **-all** option, all nondefault routing rules in the design are removed. If you specify a list of names, the nondefault routing rules in the list are removed.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes the nondefault routing rule named *wire_rule1*.

```
prompt> remove_routing_rules wire_rule1
```

SEE ALSO

```
define_routing_rule(2)  
report_routing_rules(2)
```

remove_row_type

Remove row type attribute on the specified rows.

SYNTAX

```
status remove_row_type  
[-site site_name row_name_list]
```

Data Types

<i>site_name</i>	string
<i>row_name_list</i>	list

ARGUMENTS

-site *site_name*
Specifies the name of the site to remove row type of all rows with this name. This option and the *row_name_list* are mutually exclusive; use only one.

row_name_list
Specifies row list with name pattern or a collection containing site rows to remove row type attribute. This argument and the **-site** option are mutually exclusive; use only one.

DESCRIPTION

The **remove_row_type** command removes row type attribute on a specified list of rows or on the rows of the specified site.

When you want to remove the row type of all the rows in a site, use the **-site** option.

EXAMPLES

The following example shows how to remove row type on the specified rows.

```
prompt>remove_row_type {ROW_0 ROW_1 ROW_2 ROW_4}
```

The following example shows how to remove row type for all rows in a specified site.

```
prompt>remove_row_type -site unit
```

SEE ALSO

set_row_type(2)

remove_rp_group_options

Removes relative placement (RP) group attributes from the specified relative placement groups.

SYNTAX

```
collection remove_rp_group_options  
rp_groups  
[-ignore]  
[-x_offset]  
[-y_offset]  
[-compress]
```

Data Types

rp_groups list or collection

ARGUMENTS

rp_groups
 Specifies the relative placement groups from which to remove the specified relative placement group attributes.

-ignore
 Removes the **-ignore** attribute, which indicates that the tool is to ignore this relative placement group during placement and not treat it as an relative placement group.

-x_offset
 Removes the **-x_offset** attribute of the relative placement group.

-y_offset
 Removes the **-y_offset** attribute of the relative placement group.

-compress
 Removes the **-compress** attribute of the relative placement group.

DESCRIPTION

This command removes the attributes of the relative placement groups. This command is supported only for designs that do not contain multiply-instantiated designs. This command returns a collection containing the relative placement groups for which attributes have changed. If attributes have not changed for any group, an empty string is returned.

Relative Placement usage is available with Design Compiler Ultra.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

remove_rp_group_options
1594

EXAMPLES

The following example removes attributes from a relative placement group.

```
prompt> remove_rp_group_options ripple::grp_ripple -ignore \  
-x_offset -y_offset  
{ripple::grp_ripple}
```

SEE ALSO

```
add_to_rp_group(2)  
all_rp_groups(2)  
create_rp_group(2)  
get_rp_groups(2)  
remove_from_rp_group(2)  
remove_rp_groups(2)  
set_rp_group_options(2)  
write_rp_groups(2)
```

remove_rp_groups

Removes a list of relative placement (RP) groups.

SYNTAX

```
status remove_rp_groups  
rp_groups | -all  
[-hierarchy]  
[-quiet]
```

Data Types

rp_groups list or collection

ARGUMENTS

rp_groups
Specifies a list of relative placement groups to remove.
This argument and the **-all** option are mutually exclusive.

-all
Specifies that all relative placement groups will be removed.
This option cannot be used with either the *rp_groups* argument or the **-hierarchy** option.

-hierarchy
Specifies that all relative placement groups within the hierarchy of the groups in *rp_groups* will be removed. By default, subgroups are not removed.
This option cannot be used with the **-all** option.

-quiet
Turns off messages that would otherwise be issued as relative placement groups are removed.

DESCRIPTION

The **remove_rp_groups** command removes a list of relative placement groups. When a relative placement group is removed, the memory it occupies is freed, so the object is no longer in the design.

Relative placement usage is available with Design Compiler Ultra.

Note that memory is freed to be reused by this same process. However, memory is not returned to the operating system until you exit the tool.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses **remove_rp_groups** to delete a relative placement group.

```
prompt> get_rp_groups
{mul::grp_mul ripple::grp_ripple example3::top_group}

prompt> remove_rp_groups ripple::grp_ripple
Removing rp group 'ripple::grp_ripple'
1

prompt> get_rp_groups ripple::grp_ripple
Error: Can't find object 'grp_ripple'. (UID-109)

prompt> remove_rp_groups -all
Removing rp group 'mul::grp_mul'
Removing rp group 'example3::top_group'
1
```

SEE ALSO

```
add_to_rp_group(2)
create_rp_group(2)
write_rp_groups(2)
```

remove_scaling_lib_group

Removes any previously specified scaling_lib_group from the current design, or from a subdesign.

SYNTAX

```
status remove_scaling_lib_group  
[-object_list objects]
```

Data Types

objects list

ARGUMENTS

```
-object_list objects  
    Specifies the cells or top level ports on which to remove any requested  
    scaling library groups. If you do not use this option, group(s) are removed  
    from the top level design.
```

DESCRIPTION

The **remove_scaling_lib_group** command removes the scaling library group set on the design objects previously by set_scaling_lib_group commands. After this removal, the tool will not do scaling between libraries for voltage and/or temperature.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

```
prompt> remove_scaling_lib_group -object_list top/inv
```

SEE ALSO

```
define_scaling_lib_group(2)  
set_scaling_lib_group(2)  
create_scenario(2)  
set_operating_conditions(2)
```

remove_scan_def

Removes any scan chain data stored in Milkyway.

SYNTAX

```
status remove_scan_def
```

DESCRIPTION

This command removes any scan chain data stored in the Milkyway based on the current open CEL. This scan chain data has been previously initialized by using the **read_def** command to read a scan Design Exchange Format (DEF) file or by using the **trace_scan_chain** command. When you use the **remove_scan_def** command, the netlist remains the same.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

SEE ALSO

```
check_scan_chain(2)  
optimize_dft(2)  
read_def(2)  
report_scan_chain(2)  
trace_scan_chain(2)
```

remove_scan_pin_type

Removes the scan-in or scan-out type if set on the specified pin.

SYNTAX

```
status remove_scan_pin_type  
[-pin pin | -ref_pin physical_lib_pin]
```

Data Types

<i>pin</i>	collection
<i>physical_lib_pin</i>	collection

ARGUMENTS

-pin *pin*
Specifies a collection of one pin.

-ref_pin *physical_lib_pin*
Specifies a collection of one physical lib pin.

DESCRIPTION

This command removes the scan-in or scan-out type if it is set on the given pin or physical lib pin. The type is removed from the reference library cell so corresponding pins on all instances of the reference cell have the scan type removed. You can use the **is_scan_in** and **is_scan_out** attributes to check the scan_type of a pin for scan-in and scan-out, respectively.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes the scan type from pin Z of a buffer *timer/buffd1*.

```
prompt> remove_scan_pin_type -pin [get_pins timer/buffd1/Z]
```

SEE ALSO

```
get_attribute(2)  
get_physical_lib_pins(2)  
get_pins(2)  
set_scan_pin_type(2)  
trace_scan_chain(2)
```

remove_scan_pin_type
1600

remove_scenario

Removes a scenario from memory.

SYNTAX

```
status remove_scenario  
[scenario_name | -all]
```

ARGUMENTS

scenario_name
Specifies the name of the scenario to be deleted.

-all
Specifies that all scenarios are to be deleted.

DESCRIPTION

Removes a scenario from memory. All the scenario-specific constraints defined in that scenario are deleted.

Multicorner-Multimode Support

This command uses information from active scenarios only.

EXAMPLES

The following example uses **remove_scenario** to delete a scenario.

```
prompt> create_scenario MODE1  
prompt> create_scenario MODE2  
prompt> all_scenarios  
  
MODE 1 MODE 2  
  
prompt> remove_scenario -all  
prompt> all_scenarios  
  
>
```

SEE ALSO

`all_scenarios(2)`
`current_scenario(2)`
`remove_scenario(2)`

remove_sdc

Removes all Synopsys Design Constraints (SDC).

SYNTAX

```
int remove_sdc  
[-keep_parasitics]
```

ARGUMENTS

-keep_parasitics
Use this option, when the rc network has been created and there is no need to remove it.

DESCRIPTION

Removes all constraints as defined in the current version of Synoptys Design Constraints.

The removal of annotated parasitics is expected when using the **remove_sdc** command. If you need this information after removing SDC information, reannotate the parasitics data by running **extract_rc** or **read_parasitics** again.

Multicorner-Multimode Support

This command uses information from active scenarios only.

EXAMPLES

The following command removes all SDC information from the current design:

```
prompt> remove_sdc
```

SEE ALSO

```
current_design(2)  
remove_attribute(2)  
remove_clock(2)  
remove_ideal_latency(2)  
remove_ideal_network(2)  
remove_ideal_transition(2)  
report_constraint(2)  
reset_design(2)  
reset_path(2)  
set_max_area(2)  
set_max_delay(2)  
set_max_fanout(2)  
set_max_time_borrow(2)  
set_max_transition(2)
```


remove_site_row

Removes specified site rows or all site rows defined in the current design.

SYNTAX

```
string remove_site_row  
[row_name_list]
```

Data Types

```
row_name_list      list
```

ARGUMENTS

```
row_name_list  
    Specifies one or more row names to remove. The default is to remove all site  
    rows in the current design.
```

DESCRIPTION

This command removes site rows in the current design. If you do not specify a row name, all site rows and corresponding base array in the current design are removed.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes all site rows in the current design.

```
prompt> remove_site_row
```

SEE ALSO

```
create_site_row(2)
```

remove_skew_group

Removes the user-defined skew group.

SYNTAX

```
remove_skew_group  
[-name skew_group_name]
```

Data Types

skew_group_name string

ARGUMENTS

```
-name skew_group_name  
      Removes the specified skew group from the design.
```

DESCRIPTION

This command removes the specified skew group from the design.

EXAMPLES

The following example removes the skew group named grp1.

```
prompt> remove_skew_group -name grp1  
in -0.25i
```

SEE ALSO

```
set_skew_group(2)  
report_skew_group(2)
```

remove_stdcell_filler

Deletes standard, pad, and tap filler cells.

SYNTAX

```
status remove_stdcell_filler  
-stdcell | -pad | -tap  
[-bounding_box {{llx lly} {urx ury}}]
```

Data Types

<i>llx</i>	float
<i>lly</i>	float
<i>urx</i>	float
<i>ury</i>	float

ARGUMENTS

`-stdcell | -pad | -tap`
Specifies the type of filler cells to delete. Specify `stdcell` to delete standard filler cells, `pad` to delete pad cells, and `tap` to delete tap cells, respectively.

`-bounding_box {{llx lly} {urx ury}}`
Indicates the lower left and upper right coordinates of the rectangle from where the filler cells are to be removed. If you do not specify this option, the working area is the entire chip.

DESCRIPTION

The **remove_stdcell_filler** command removes filler cells from the design.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes standard cell fillers from the specified area

```
prompt> remove_stdcell_filler -stdcell -boundingbox{{10 10} {500 500}}
```

SEE ALSO

```
insert_diode(2)  
insert_ng_filler(2)  
insert_pad_filler(2)  
insert_stdcell_filler(2)  
insert_well_filler(2)
```

remove_supply_net

Removes supply nets from the specified power domain. This command is supported only in UPF mode.

SYNTAX

```
status remove_supply_net
[-verbose]
[supply_nets]
[-domain domain_name]
```

Data Types

<i>supply_nets</i>	list
<i>domain_name</i>	list

ARGUMENTS

-verbose
Prints additional messages.

supply_nets
Specifies the supply nets to be deleted. The option value could be a collection of supply nets or name patterns.
The default value is '*' which matches all supply nets under current scope.
The command fails if it could not find any specified supply nets.

-domain *domain_name*
Since one supply_net could be defined in more than one power_domains, this option specify in which power_domains the supply nets are to be deleted.
The command fails if it fails to find any specified power_domains.
This option is optional.
If this option is specified, all specified supply_nets are removed from these specified power_domains only. If there is any specified supply_nets which are not defined in the specified power_domains, these supply_nets will be skipped and they won't be removed from any power_domains.
If this option is not specified, all specified supply_nets will be removed from all power_domains in which they're defined.

DESCRIPTION

The *remove_supply_net* command removes an existing supply net in a power domain.

This command returns a 1 if successful and returns a 0 otherwise.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes the A_VDD supply net from the PD1 power domain.

```
prompt> remove_supply_net A_VDD -domain PD1
1
```

The following example removes all supply nets that are defined in the PD1 power domain.

```
prompt> remove_supply_net [get_supply_nets -hier] \
      -domain [get_power_domains PD1] -verbose
Removed supply_net 'A_VDD' from power_domain 'PD1' successfully.
Removed supply_net 'A_VSS' from power_domain 'PD1' successfully.
Warning: Skipped to remove supply_net 'REG_BLK/A_VDD' since it is not
defined in power_domain 'PD1'.
Warning: Skipped to remove supply_net 'REG_BLK/A_VSS' since it is not
defined in power_domain 'PD1'.
1
```

You could also write the above example as follows:

```
prompt> remove_supply_net \
      [get_supply_nets -hier -of_objects [get_power_domains PD1]] \
      -domain [get_power_domains PD1] -verbose
Removed supply_net 'A_VDD' from power_domain 'PD1' successfully.
Removed supply_net 'A_VSS' from power_domain 'PD1' successfully.
1
```

SEE ALSO

connect_supply_net(2)
create_supply_net(2)
get_supply_nets(2)
report_supply_net(2)
set_scope(2)
get_power_domains(2)

remove_supply_port

Removes a supply port from the scope of the specified power domain or the current scope. This command is supported only in UPF mode.

SYNTAX

```
status remove_supply_port
[-verbose]
[supply_ports]
[-domain domain_name]
```

Data Types

<i>supply_ports</i>	list
<i>domain_name</i>	string

ARGUMENTS

-verbose
Prints additional messages.

supply_ports
Specify the supply ports to be deleted. The option value could be a collection of supply ports or name patterns.
The default value is '*' which matches all supply ports under current scope.
The command fails if it could not find any specified supply port.

-domain *domain_name*
Specifies the power domain in whose scope the supply port is to be deleted.
If the specified power domain does not exist in the current scope, the command fails.
If this option is specified, the *supply_ports* option must be a simple name, not collection.

DESCRIPTION

The *remove_supply_port* command removes an existing supply port in the specified scope

The command returns 1 on success, and returns 0 otherwise.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes the VN1 supply port from the PD1 power domain.

```
remove_supply_port
1608
```

```
prompt> remove_supply_port VN1 -domain PD1
1
```

The following example removes all supply ports in the current scope whose name begins with "VN".

```
prompt> remove_supply_port [get_supply_ports -hier VN*]
1
```

SEE ALSO

```
create_supply_port(2)
get_supply_ports(2)
report_supply_port(2)
set_scope(2)
```

remove_target_library_subset

Removes target library subset constraints (including both the target library subset specified by `library_list` option and `-milkyway_reflibs` option) from root design or from specified instances.

SYNTAX

```
int remove_target_library_subset  
[-object_list cells]  
[-top]
```

ARGUMENTS

`-object_list cells`

Specifies the cells from which to remove the target library subset. The cells should be instances of hierarchical designs, subset restriction applies to those instances and their children. You must specify at least one from **-top** and **-object_list**. If neither **-top** nor **-object_list** is specified, command errors out.

`-top`

If specified, the target library subset is removed from the root design. You must specify at least one from **-top** and **-object_list**. If neither **-top** nor **-object_list** is specified, command errors out.

DESCRIPTION

Removes target library subset or milkyway reference library subset from the given instances, or the root design.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes the `target_library` subset from the root design and from block `u1`.

```
prompt> remove_target_library_subset -object_list [get_cells u1] -top
```

SEE ALSO

```
set_target_library_subset(2)  
report_target_library_subset(2)  
check_target_library_subset(2)  
set_operating_conditions(2)  
target_library(3)
```


remove_terminal

Removes terminals.

SYNTAX

```
status_value remove_terminal  
terminals  
[-verbose]
```

Data Types

terminals list

ARGUMENTS

terminals
Specifies patterns or a list of nonempty collections of handles to terminals.

-verbose
Prints more information.

DESCRIPTION

This command removes all specified terminals. If the input collection is empty, the command returns 0 to indicate failure.

If there are invalid handles in the input handle collection, the command-line interpreter removes them from the collection and invokes the **remove_terminal** command with the updated collection.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes all terminals whose name matches *clock**:

```
prompt> remove_terminal clock*  
1
```

SEE ALSO

```
create_terminal(2)  
get_terminals(2)
```

remove_text

Removes text.

SYNTAX

```
int remove_text  
text_list | -all
```

Data Types

text_list list

ARGUMENTS

text_list

Specifies the text to remove. The *text_list* value can include collections of text type, patterns that use wildcards, or certain command names. The *text_list* and **-all** options are mutually exclusive.

-all

Specifies to remove all text in the current design. The *text_list* and **-all** options are mutually exclusive.

DESCRIPTION

This command removes all specified text. Users can specify text by using the name of text or collections of text type which can be generated by commands like *get_text* and *get_selection*.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows the use of a pattern that removes the text object with *object_id 6400*.

```
prompt> remove_text TEXT#6400  
1
```

The following example shows the use of a wildcard pattern that removes all text.

```
prompt> remove_text *  
1
```

The following example shows the use of a wildcard pattern that removes all text that start with *TEXT#*.

```
remove_text  
1612
```

```
prompt> remove_text TEXT#*  
1
```

The following example firstly creates a collection containing text's bounding box within the specified rectangle {10 9} {45 15} then removes text in the collection.

```
prompt> remove_text [get_text \  
? * -within {{10 9} {45 15}}  
1
```

The following example removes all text.

```
prompt> remove_text -all  
1
```

SEE ALSO

```
create_text(2)  
get_text(2)
```

remove_tie_cells

Removes tie cells and changes their connected pins and ports in the Milkyway database.

SYNTAX

```
status remove_tie_cells  
[-use_default_tie_net]  
tie_cell_list
```

Data Types

tie_cell_list collection

ARGUMENTS

-use_default_tie_net

Connects the signal pins that were originally driven by the deleted tie cells to the default SNPS_LOGIC0 or SNPS_LOGIC1 nets. If you do not specify this option, the tool keeps the original nets driven by the tie pins and sets the net_type attribute for those nets to tie-low or tie-high.

tie_cell_list

Specifies a list of tie cell instances to be removed.
If any of the specified cells do not exist in the design or they are not tie cells, the tool issues a warning message and continues processing other cells in the specified tie cell list.

DESCRIPTION

Removes the specified tie cells. The signal pins that were originally driven by the deleted tie cells are connected to Milkyway tie-low or tie-high nets.

The signal pins can be either leaf pins or hierarchical pins.

By default, the command keeps the original net driven by the tie cells and sets the net_type attribute to tie-low or tie-high, as determined by the tie pin attribute.

If you specify the **-use_default_tie_net** option, tie nets become the default SNPS_LOGIC1 or SNPS_LOGIC0 nets. The original nets driven by the tie cells are deleted. If a tie cell is removed for a hierarchical port, the tie net name uses the port name as the prefix in addition to SNPS_LOGIC1 or SNPS_LOGIC0. For example, if a hierarchical port is named "Q" and it is tied low inside the hierarchical module, the tie net name becomes "Q_SNPS_LOGIC0".

EXAMPLES

The following example uses the **remove_tie_cells** command to remove all the existing tie cells.

```
remove_tie_cells  
1614
```

The **get_cells** command is used to verify the tie cells before and after running the **remove_tie_cells** command.

```
prompt> all_tieoff_cells
{TIELOU1 TIELOU2 TIELOU3}
prompt> remove_tie_cells -use_default_tie_net [all_tieoff_cells]
prompt> get_cells {TIELOU1 TIELOU2 TIELOU3}
Warning: No cell objects matched 'TIELOU1' (SEL-004)
Warning: No cell objects matched 'TIELOU2' (SEL-004)
Warning: No cell objects matched 'TIELOU3' (SEL-004)
```

The following example uses the **remove_tie_cells** command to remove a collection of tie cells.

The **report_tie_nets** command is used to verify the new logic zero net after running the **remove_tie_cells** command.

The **get_nets** command is used to verify that the old nets connected to the deleted tie cells are also deleted.

```
prompt> get_nets -all U2/TIELOU1_net0
{U2/TIELOU1_net0}
prompt> remove_tie_cells -use_default_tie_net {U2/TIELOU0 U2/TIELOU1}
prompt> report_tie_nets
Design has 1 tie low net(s):
    net U2/SNPS_LOGIC0 connects to the following port(s) or pin(s):
        U2/U1/D
        U2/U1/NT
        U2/U4/S
prompt> get_nets -all U2/TIELOU1_net0
Warning: No net objects matched 'U2/TIELOU1_net0' (SEL-004)
```

SEE ALSO

```
get_cells(2)
all_tieoff_cells(2)
report_tie_nets(2)
connect_tie_cells(2)
disconnect_net(2)
remove_cell(2)
```

remove_track

Removes tracks from the current design.

SYNTAX

```
status remove_track  
-all | patterns | -layer layer [-dir X | Y]  
[-verbose]
```

Data Types

<i>patterns</i>	list
<i>layer</i>	collection of one item

ARGUMENTS

-all

Removes all tracks in the current design.

The *patterns*, **-all**, and **-layer** arguments are mutually exclusive; you must specify one of these arguments.

patterns

Matches the track names in the current design against the specified patterns. You can specify the patterns by using the following formats:

- An asterisk (*), which indicates all tracks
- "TRACK_*", which indicates all tracks
- Track names, which are in the format TRACK_*object_id*

The *patterns*, **-all**, and **-layer** arguments are mutually exclusive; you must specify one of these arguments.

-layer *layer*

Specifies the routing layer from which to remove the track. You can specify only one layer, either by layer name, layer number, or a collection containing one layer.

The *patterns*, **-all**, and **-layer** arguments are mutually exclusive; you must specify one of these arguments.

-dir X | Y

Specifies the direction of the routing tracks to be removed. The valid values are **X** and **Y**.

By default, the direction is the routing direction of the layer specified in the physical library.

The option must be used with the **-layer** option.

-verbose

Prints additional messages.

DESCRIPTION

This command removes the specified tracks from the current design.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes all tracks that are inside {{2 2} {25 25}}.

```
prompt> remove_track [get_tracks -within {{2 2} {25 25}}]  
1
```

The following example removes all tracks that are located on the metal2 layer.

```
prompt> remove_track [get_tracks -of_objects METAL2] -verbose  
Removing track TRACK_4683  
1
```

The following example removes the routing tracks from the routing layer named m3 on the floorplan.

```
prompt> remove_track -layer m3  
Warning: Direction is not specified. Using the layer preferred direction.  
(MWUI-125)  
1  
prompt> remove_track -layer m3 -dir X  
1
```

SEE ALSO

create_track(2)
get_tracks(2)
report_track(2)

remove_unconnected_ports

Removes unconnected ports or pins from cells, references, and subdesigns.

SYNTAX

```
int remove_unconnected_ports  
cell_list  
[-blast_buses]
```

Data Types

cell_list list

ARGUMENTS

cell_list
Specifies a list of cells or instances whose unconnected ports or pins are to be removed.

-blast_buses
Indicates that if a bus has a port that is unconnected, the bus is to be deleted, the unconnected ports are to be removed, and single-bit buses are to be created for the remaining ports.

DESCRIPTION

This command removes unconnected ports or pins from cells, references, and subdesigns. The current design must be linked and uniquified before you run this command. A port is considered unconnected if it is not connected on the inside of the design. The unconnected ports or pins of each cell in **cell_list** are removed from the cell, its reference, and its subdesign simultaneously so that the current design links after applying **remove_unconnected_ports**.

After a port is removed, any unused nets in the top design and in the design from where the port was removed are cleaned up.

Unless you specify the **-blast_buses** option, bused ports are removed only if all ports of the bus are unconnected.

Note: If your design contains references to subdesigns, there is a possibility of encountering link errors when using **remove_unconnected_ports**. For example, consider the situation where two designs, A and B, both reference subdesign C. Executing **remove_unconnected_ports** on design A might remove ports from subdesign C, so that design B then fails to link. The same situation occurs when design A is read in, **remove_unconnected_ports** is applied, then the original version of design A is read in again. One workaround is to ungroup one of the designs completely so that it does not have references to subdesigns.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes all unconnected ports that exist in the current design.

```
prompt> remove_unconnected_ports find( -hierarchy cell, "")  
Removing port 'CI' from design 'uport1_DW01_addsub_5_0'  
Removing port 'CO' from design 'uport1_DW01_addsub_5_0'  
1
```

The following example removes all unconnected ports that exist in the current design, any bus that has an unconnected port is deleted, and the unconnected ports of the bus are removed.

```
prompt> remove_unconnected_ports\  
-blast_buses find( -hierarchy cell, "")  
Removing port 'A_4' from design 'uport1_DW01_addsub_5_0'  
Removing port 'A_3' from design 'uport1_DW01_addsub_5_0'  
Removing port 'B_4' from design 'uport1_DW01_addsub_5_0'  
Removing port 'B_3' from design 'uport1_DW01_addsub_5_0'  
Removing port 'CI' from design 'uport1_DW01_addsub_5_0'  
Removing port 'SUM_4' from design 'uport1_DW01_addsub_5_0'  
Removing port 'SUM_3' from design 'uport1_DW01_addsub_5_0'  
Removing port 'CO' from design 'uport1_DW01_addsub_5_0'  
1
```

SEE ALSO

`remove_port(2)`

remove_user_shape

Remove objects of user shapes. The objects can be specified by a collection or by name.

SYNTAX

```
status_value remove_user_shape
[-verbose]
user_shapes
```

ARGUMENTS

-verbose

If this option not specified, suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

user_shapes

Specifies a nonempty collection handles to user shapes or the name of the user shape.

DESCRIPTION

This command removes all specified user shapes, and then returns 1 if successful, or 0 if it fails.

If invalid handles occur in the input handle collection, the command line interpreter removes them from the collection and invokes the `remove_user_shape` command with the updated collection.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes user shapes.

```
> set a [get_user_shape]
{"RECTANGLE#6682", "TRAPEZOID#7196", "POLYGON#6683"}
> remove_user_shape $a
1
```

SEE ALSO

```
create_user_shape(2)
get_user_shapes(2)
remove_net_shape(2)
```

remove_via

Removes vias.

SYNTAX

```
status_value remove_via
[-verbose]
vias
```

Data Types

vias list

ARGUMENTS

-verbose
Prints additional messages.

vias
Specifies a nonempty collection of handles to vias.

DESCRIPTION

This command removes all specified vias. If the input collection is empty, the command returns 0 to indicate failure. If any invalid handles occur in the input handle collection, the command line interpreter removes them from the collection and invokes the **remove_via** command with the updated collection.

This command returns 1 if successful, or 0 if it fails.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes the vias specified by the **get_vias** command:

```
prompt> remove_via [get_vias -of_objects n300
1
```

SEE ALSO

```
create_net_shape(2)
create_via(2)
get_net_shapes(2)
get_vias(2)
remove_net_shape(2)
```

```
set_via_array_size(2)
```

remove_voltage_area

Removes voltage areas from the current design.

SYNTAX

```
int remove_voltage_area  
[-verbose]  
-all | patterns
```

Data Types

patterns list

ARGUMENTS

-verbose
Prints additional messages.

-all
Removes all voltage areas in the current design.

patterns
Specifies the voltage areas. The patterns can be a collection handle of voltages or names of patterns. You can use the *get_voltage_areas* command. to specify objects.

DESCRIPTION

This command removes voltage areas from the design.

When a voltage area is removed, all its hierarchical cells are also deassociated with the voltage area. After one hierarchical cell is deassociated, its all leaf child cells will be reassociated with the its nearest ascent's voltage area, if any. If none of its ascents is associated with any voltage area, or it doesn't have any ascents, its all leaf child cells will be free.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes voltage areas whose names begin with *tap*.

```
prompt> remove_voltage_area tap*
```

SEE ALSO

create_voltage_area(2)

```
get_voltage_areas(2)
report_voltage_area(2)
update_voltage_area(2)
```

remove_vt_filler_rule

removes multiple threshold voltage filler cell insertion rules.

SYNTAX

```
integer remove_vt_filler_rule  
-threshold_voltage vt_type_1_vt_type_2
```

ARGUMENTS

```
-threshold_voltage vt_type_1_vt_type_2  
    Specifies the threshold voltage types of the rule to be removed.
```

DESCRIPTION

Use this command to remove multiple threshold voltage filler rules of the design.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example will remove threshold voltage rule for "vt_type_A" "vt_type_B"

```
prompt> remove_vt_filler_rule -threshold_voltage "vt_type_A" "vt_type_B"
```

SEE ALSO

```
insert_stdcell_filler(2)  
set_cell_vt_type(2)  
set_left_right_filler_rule(2)  
set_vt_filler_rule(2)  
report_vt_filler_rule(2)
```

remove_well_filler

Deletes all well fillers in a specified layer.

SYNTAX

```
int remove_well_filler  
-layer {layer_name}  
[-core_only]
```

Data Types

layer_name string

ARGUMENTS

```
-layer {layer_name}  
    Specifies layer from which well fillers are to be removed. This can be any  
    valid  
  
-core_only  
    Specifies that the core area is to be considered as the bounding-box for  
    purging the well fillers.
```

DESCRIPTION

The *remove_well_filler* command removes well fillers from a specified layer. Well filler are inserted in the design after placement through *insert_well_filler* command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes well fillers from layer NWELL:
prompt> remove_well_filler -layer NWELL

SEE ALSO

```
insert_diode(2)  
insert_ng_filler(2)  
insert_pad_filler(2)  
insert_stdcell_filler(2)  
insert_well_filler(2)
```


remove_xtalk_prop

remove aggressor/victim list properties

SYNTAX

status **remove_xtalk_prop**

DESCRIPTION

This command removes aggressor/victim list properties.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example will remove the net xtalk info.

```
prompt> remove_xtalk_prop
```

SEE ALSO

remove_zrt_filler_with_violation

Deletes filler cells that have Zroute routing violations.

SYNTAX

```
status remove_zrt_filler_with_violation  
[-name cell_name]
```

Data Types

cell_name list

ARGUMENTS

-name *cell_name*
 Specifies the pattern of the cell instance names.
 If you do not specify this option (if you specify "*" or "."), the pattern defaults to ".*xofiller.*". If you use this default pattern, all instance names that have the "xofiller" substring in them, including hierarchical names, are considered for removal. If cells to be deleted match the name pattern, but are not filler cells, an information message alerts you about the first 50 cells that are deleted.

DESCRIPTION

This command deletes all filler cells that have Zroute routing design rule violations. Filler cells are standard cells without any connections to signal nets.

This command is useful when you add filler cells after performing signal routing with Zroute.

Filler cells that have power or ground (PG) pins that are not assigned to a PG net are deleted if they touch other PG objects, such as rails or pins assigned to a PG net. Therefore, you should connect these pins to PG nets before running the **remove_zrt_filler_with_violation** command. For information about connecting pins to PG nets, see the man pages for the **connect_pg_nets** or **insert_stdcell_filler** commands.

Internal violations in filler cells (between objects of the same cell) are not considered for filler cell removal. If the tool detects this type of violation, a warning is issued once per individual cell.

The tool outputs an information message about the violation type that caused the deletion for the first 50 deleted cells. If there are multiple violations for the same cell, only one of them is listed.

It is a known limitation of the **remove_zrt_filler_with_violation** command that when filler cells are removed, new violations might be introduced with neighboring filler cells. Therefore, you should run the **remove_zrt_filler_with_violation** command multiple times until no more filler cells are removed. In other words, until the informative message "Deleted 0 cell instances" is displayed.

EXAMPLES

The following example removes all filler cells with the prefix "xofiller" that have Zroute routing violations.

```
prompt> remove_zrt_filler_with_violation -name "xofiller.*"
```

SEE ALSO

`insert_stdcell_filler(2)`

remove_zrt_redundant_shapes

Dangling wires and vias on user-specified nets will be removed.

SYNTAX

```
status remove_zrt_redundant_shapes
[-nets {collection_of_nets}]
[-route_types {list_of_route_types}]
[-layers {collection_of_layers}]
[-initial_drc_from_input true | false]
[-remove_loop_shapes true | false]
[-remove_floating_shapes true | false]
[-remove_dangling_shapes true | false]
```

ARGUMENTS

- nets {collection_of_nets}
Specifies the target nets with dangling wires or vias to be removed. The default is all nets.
- route_types {list_of_route_types}
Specifies a list of the target route types. The valid values for route_types are fixed_route and nonfixed_route. The default is both fixed_route and nonfixed_route.
- layers {collection_of_layers}
Specifies the target layers of the dangling wires or vias to be removed. Metal and cut layer names in technology file are supported. The default is all layers.
- initial_drc_from_input true | false
Specifies if running the drc checker before starting removing of dangling wires and vias. The default is true.
- remove_loop_shapes true | false
Specifies if loops will be removed. The default is false.
- remove_floating_shapes true | false
Specifies if floating shapes will be removed. The default is true.
- remove_dangling_shapes true | false
Specifies if dangling shapes will be removed. The default is true.

DESCRIPTION

The **remove_zrt_redundant_shapes** command remove dangling wires and vias on the specified nets.

EXAMPLES

The following example sets the **-nets** option for n1

```
remove_zrt_redundant_shapes
1630
```

```
prompt> remove_zrt_redundant_shapes -nets n1
```

The following example sets the **-route_types** option to `nonfixed_route`

```
prompt> remove_zrt_redundant_shapes -route_types {nonfixed_route}
```

The following example sets the **-layers** option to `METAL2`, `VIA23` and `METAL3`

```
prompt> remove_zrt_redundant_shapes -layers {METAL2 VIA23 METAL3}
```

The following example sets the **-initial_drc_from_input** option to be `false`

```
prompt> remove_zrt_redundant_shapes -initial_drc_from_input false
```

The following example sets the **-remove_loop_shapes** option to be `true`

```
prompt> remove_zrt_redundant_shapes -remove_loop_shapes true
```

The following example sets the **-remove_floating_shapes** option to be `true`

```
prompt> remove_zrt_redundant_shapes -remove_floating_shapes true
```

The following example sets the **-remove_dangling_shapes** option to be `true`

```
prompt> remove_zrt_redundant_shapes -remove_dangling_shapes true
```

SEE ALSO

rename

Rename or delete a command.

SYNTAX

```
string rename  
oldName newName  
string oldName
```

ARGUMENTS

DESCRIPTION

Rename the command that used to be called *oldName* so that it is now called *newName*. If *newName* is an empty string then *oldName* is deleted. *oldName* and *newName* may include namespace qualifiers (names of containing namespaces). If a command is renamed into a different namespace, future invocations of it will execute in the new namespace. The **rename** command returns an empty string as result.

Note that the **rename** command cannot be used on permanent procedures. Depending on the application, it can be used on all basic builtin commands. In some cases, the application will allow all commands to be renamed.

WARNING: **rename** can have serious consequences if not used correctly. When using **rename** on anything other than a user-defined Tcl procedure, you will be warned. The **rename** command is intended as a means to wrap other commands: that is, the command is replaced by a Tcl procedure which calls the original. Parts of the application are written as Tcl procedures, and these procedures can use any command. Commands like **puts**, **echo**, **open**, **close**, **source** and many others are often used within the application. Use **rename** with extreme care and at your own risk. Consider using **alias**, Tcl procedures, or a private namespace before using **rename**.

EXAMPLES

This example renames `my_proc` to `my_proc2`:

```
prompt> proc my_proc {} {echo "Hello"}  
prompt> rename my_proc my_proc2  
prompt> my_proc2  
Hello  
prompt> my_proc  
Error: unknown command 'my_proc' (CMD-005).
```

SEE ALSO

`define_proc_attributes(2)`

rename_mw_cel

Renames a Milkyway design.

SYNTAX

```
int rename_mw_cel  
old_name  
new_name  
[-all_version]
```

Data Types

<i>old_name</i>	string
<i>new_name</i>	string

ARGUMENTS

old_name
Specifies the name of the Milkyway design to be renamed. The value of *old_name* should contain only the base name of an `mw_cel`. It cannot have a view name and version.

new_name
Specifies the new name of the Milkyway design. The *new_name* should not match the name of any Milkyway designs referenced by a instance in the *old_name* option. The *new_name* should contain only the base name of a `mw_cel`. It cannot have a view name and version.

`-all_version`
Indicates to rename all versions of the Milkyway design. By default, this command renames the newest version of the Milkyway design specified by *old_name* to *new_name*.

DESCRIPTION

This command renames a Milkyway design. By default, it renames the newest version of the Milkyway design specified by *old_name* to *new_name*. It renames only the latest version of all views of an `mw_cel`, unless you use the **-all_version** option; then it renames all versions of all views of an `mw_cel`.

Each Milkyway design name must be unique within the library containing that Milkyway design. An error occurs if the *new_name* you specify matches the name of a Milkyway design that is referenced by a instance in *old_name*, as this will cause a recursive reference to exist.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example renames a Milkyway design named *test* to *bill*:

```
prompt> get_mw_cels *
{"top", "test"}
prompt> rename_mw_cel test bill
1
prompt> get_mw_cels *
{"top", "bill"}
```

SEE ALSO

```
close_mw_cel(2)
copy_mw_cel(2)
create_mw_cel(2)
current_mw_cel(2)
get_mw_cels(2)
mw_cel_collection(2)
open_mw_cel(2)
remove_mw_cel(2)
```


rename_mw_lib

Renames a Milkyway library.

SYNTAX

```
status_value rename_mw_lib  
-from lib_name  
-to lib_name
```

Data Types

```
lib_name      string
```

ARGUMENTS

```
-from lib_name  
    Specifies the name of the Milkyway library that is to be renamed. The  
    specified Milkyway library must not be open in current session.  
  
-to lib_name  
    Specifies the new name of the Milkyway library.
```

DESCRIPTION

This command renames a Milkyway library.

If this library is a reference library of another main library, you may need to use the **set_mw_lib_reference** command to update reference library information in the main library.

A status indicating success or failure is returned.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example changes the name of the library from *access05* to *access06*:

```
prompt> rename_mw_lib -from access05 -to access06
```

SEE ALSO

`copy_mw_lib(2)`

replace_power_switch

Replaces the reference cells of the specified header or footer cell instances with the specified reference cell and performs IR drop analysis.

SYNTAX

```
status_value replace_power_switch  
-cells {instances}  
-lib_cell {lib_cell_or_power_switch}  
[-virtual_pg_net virtual_net]  
[-real_pg_net real_net]  
[-no_analyze_power]
```

Data Types

<i>instances</i>	collection or list
<i>lib_cell_or_power_switch</i>	collection or list of one item
<i>virtual_net</i>	collection or list of one item
<i>real_net</i>	collection or list of one item

ARGUMENTS

```
-cells {instances}  
    Specifies the header or footer instances to be replaced.
```

```
-lib_cell {lib_cell_or_power_switch}  
    Specifies the header or footer library cell to be used for replacement. The  
    lib_cell_or_power_switch argument can contain only one header or footer  
    library cell. The specified library cell must be a multithreshold CMOS  
    (MTCMOS) cell and must be functionally equivalent to the cells being  
    replaced. Use lib cell for non-United Power Format (non-UPF) mode, and use  
    power switch for UPF mode.
```

```
-virtual_pg_net virtual_net  
    Specifies a virtual power or ground net on which to perform IR drop analysis.  
    Virtual power nets connect to the virtual power pins of the MTCMOS cells. The  
    virtual power net can be powered down by the MTCMOS header or footer cells  
    to reduce leakage power.  
    Under UPF mode, this option is optional. The default is to get the output  
    supply net from the output port of the power switch.
```

```
-real_pg_net real_net  
    Specifies a real power or ground net to perform power network analysis. Real  
    power nets connect to the real power pins of MTCMOS cells. The real power net  
    is not powered down.  
    Under UPF mode, this option is optional. The default is to get the input  
    supply net connectivity from the input port of the power switch.
```

```
-no_analyze_power  
    Skips the analyzing of power networks.
```

DESCRIPTION

This command replaces existing header or footer cells with the specified library cell and performs IR drop analysis with the power network analysis engine. It generates an IR drop report before and after cell replacement.

Header and footer cells typically contain low leakage P/N transistors to reduce the leakage current during power down mode.

You can use this command at the MTCMOS power planning stage to optimize the IR drop of the design. Use it when the MTCMOS cells and power network already physically exist in the design.

The specified instances to be replaced should be connected to the specified virtual and real power and ground nets. The tool attempts to adjust the location of the incoming cells so that the metal connections of the pins are preserved. The **replace_power_switch** command does not perform routing.

It assumes that the cells to be replaced are functionally equivalent, have the same cell height, and that the locations of the pins are compatible. Otherwise, the replacement can result in power and ground connections that are physically broken and impossible to realize. This can cause incorrect IR drop analysis.

If the replacement cell is larger than the original cell, the tool does not perform legalization. You must perform legalization later if the design from this command is to be accepted.

PREREQUISITES

Before executing this command, perform the following tasks:

1. Define the power-on resistance of the header or footer cell by using the **set_attribute** command to set the **mtcmos_resistance** attribute.
2. Specify the metal layer for the power pins by using the **set_attribute** command to set the **mtcmos_pin_layers** attribute.
3. Define the power pad or virtual power pad by using the **create_fp_virtual_pad** command.
4. Verify that the virtual and real power and ground nets of all standard cells have been properly connected logically in the netlist. Use the **derive_pg_connection** command if necessary. A standard cell to be powered down should connect to a virtual power or ground net. A standard cell that always stays on should be connected to a real power or ground net.
5. Use **set_mtcmos_pna_strategy** to define power network analysis related options.

The steps described above are necessary for the power network analysis engine to compute the IR drop.

EXAMPLES

The following example replaces the reference cell of the instances named *HEAD_8* with the HEAD_16 library cell. The power nets to be analyzed are VIRTUAL_VDD and VDD. The command generates IR drop reports before and after replacement.

```
prompt> replace_power_switch\  
-cells [get_cells "*HEAD_8*"]\  
-lib_cell HEAD_16 -virtual_pg_net VIRTUAL_VDD\  
-real_pg_net VDD
```

The following example is for under UPF mode, so it uses PS_A for the library cell.

```
prompt> replace_power_switch\  
-cells [get_cells "*HEAD_8*"]\  
-lib_cell PS_A -virtual_pg_net VIRTUAL_VDD\  
-real_pg_net VDD
```

SEE ALSO

```
create_power_switch_array(2)  
analyze_fp_rail(2)  
derive_pg_connection(2)  
explore_power_switch(2)  
optimize_power_switch(2)  
report_mtcmos_pna_strategy(2)  
set_mtcmos_pna_strategy(2)
```

report_adjusted_endpoints

Generates a report of the adjusted endpoints in feasibility mode.

SYNTAX

```
int report_adjusted_endpoints  
[-zero_path]  
[-zero_wire_load]  
[-io]  
[-slack_threshold]  
[-all]  
[-scenarios scenarios]  
[-verbose]
```

Data Types

scenarios list

ARGUMENTS

-zero_path

Reports only on endpoints that have been adjusted for zero_path violations, and the zero_path margins have been added to the required times of the endpoints.

-zero_wire_load

Reports only on endpoints that have been adjusted for zero_wire_load violations, and the zero_wire_load margins have been added to the required times of the endpoints.

-io

Reports only on endpoints that have been adjusted due to I/O margins that were added to the endpoint required times.

-slack_threshold

Reports only on endpoints that have been adjusted for slack_threshold, and slack threshold margins have been added to the required times of the endpoints.

-all

Reports on all adjusted endpoints irrespective of the reasons for the adjustment.

-scenarios *scenarios*

Reports only on adjusted endpoints of the given scenarios. The default is to report only on adjusted endpoints of the current scenarios.

-verbose

Reports all the adjusted endpoints individually and the reason of the adjustment for the given scenarios. The default is non-verbose mode, which generates a summary of the adjustments for the given scenario or the current scenario otherwise.

DESCRIPTION

Feasibility mode enables constraint relaxation of unachievable endpoints for the reporting commands and the **place_opt** command. During feasibility mode, unachievable endpoints are identified and the required time on the unachievable endpoints are relaxed based on the unachievable timing portion. Endpoints with zero path violations and zero wire load violations are considered unachievable endpoints.

This command allows you to obtain information about the endpoints that have been adjusted for the specified reasons and scenarios. The default issues reports on the adjusted endpoints of the current scenario.

EXAMPLES

The following examples show how to use the **report_adjusted_endpoints** command.

```
prompt> report_adjusted_endpoints -zero_path -scenario s1
Scenario s1
There are a total of 3 endpoints adjusted.

prompt> report_adjusted_endpoints -zero_path -scenario s1 -verbose
Scenario s1
Top/B_A/B/C1/a (zero path)
Top/B_A/C2/b (zero path)
Top/C_A/C2/b (zero path)
There are a total of 3 endpoints adjusted.
```

SEE ALSO

```
place_opt(2)
set_feasibility_options(2)
report_feasibility_options(2)
```

report_ahfs_options

Writes a report about the automatic high-fanout synthesis (AHFS) options.

SYNTAX

integer **report_ahfs_options**

ARGUMENTS

The **report_ahfs_options** command has no arguments.

DESCRIPTION

This command reports all of the options set for running automatic high-fanout synthesis (AHFS). See the **set_ahfs_options** command man page for the default values of the options and a description of how each option can control the AHFS flow.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports the current AHFS options:

```
prompt> report_ahfs_options

Report AHFS options:
*****
AHFS options for the design:
  High-fanout(HF)  threshold: 50
  Medium-fanout(MF) threshold: -1
  Remove effort: medium
  Enable port punching: ON
  Default Reference : buffer_1
1
```

SEE ALSO

set_ahfs_options(2)

report_annotated_check

Displays all annotated timing checks on the current design.

SYNTAX

```
int report_annotated_check  
[-nosplit]
```

ARGUMENTS

-nosplit

Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column. **-nosplit** prevents line splitting and facilitates writing software to extract information from the report output.

DESCRIPTION

Lists all annotated timing checks in the current design. Pin-to-pin timing checks reported are setup and hold.

To list annotated delays, use **report_annotated_delay**.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following is an example of an annotated report.

```
prompt> report_annotated_check
```

```
*****  
Report : annotated_check  
Design : counter  
Version: v3.1  
Date   : Tue Apr 14 19:42:25 1992  
*****
```

Cell Name	From	To	Rise	Fall	Timing Check
U1	c	d	-0.20	-0.20	hold
U1	c	d	2.20	2.20	setup

SEE ALSO

`remove_annotated_check(2)`
`report_annotated_delay(2)`
`reset_design(2)`
`set_annotated_check(2)`

report_annotated_delay

Displays all annotated delays on cells and nets of the current design.

SYNTAX

```
int report_annotated_delay  
[-cell] [-net] [-nosplit] [-summary]
```

ARGUMENTS

-cell

Reports all data annotated on cells. By default, both cell- and net-annotated data are reported.

-net

Reports all data annotated on nets. By default, both cell- and net-annotated data are reported.

-nosplit

Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column. The **-nosplit** option prevents line splitting and facilitates writing software to extract information from the report output.

-summary

Reports the total number of cell/net delay arcs and how many of them have got annotation. An arc will be considered as annotated only if all the four delay values (rise, fall, min, max) are present on it.

DESCRIPTION

report_annotated_delay lists all annotated data on cells and nets in the current design. The cell-annotated data reported are the pin-to-pin delays. The net-annotated data reported are pin-to-pin delays, net capacitances, and net resistances. Note that the delay values reported are the resulting cell and net delays used in **report_timing** and might not be your annotated values if delays were annotated using the **-load_delay net** option of the **read_sdf** or **set_annotated_delay** command.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following is an example of an annotated report.

```
prompt> report_annotated_delay
```

```
report_annotated_delay  
1644
```

```

*****
Report : annotated -cell
Design : counter
Version: v3.0
Date   : Tue Apr 14 19:42:25 1992
*****

```

Cell Name	From	To	Rise	Fall
CO	A	Z	100.00	100.00

```

*****
Report : annotated -net
Design : counter
Version: v3.0
Date   : Tue Apr 14 19:42:25 1992
*****

```

Net Name	From	To	Rise	Fall	Load	Res.
h	ffc/QN	w/A	200.00	200.00	50.00	200.00
h	ffc/QN	m/B	200.00	200.00	50.00	200.00
h	ffc/QN	r/B			50.00	200.00
m	m/Z	CO/A	200.00	200.00	40.00	100.00

SEE ALSO

```

read_sdf(2)
set_annotated_delay(2)
remove_annotated_delay(2)
reset_design(2)

```

report_annotated_transition

Displays annotated transitions on all pins of the current design.

SYNTAX

```
int report_annotated_transition  
-nosplit
```

ARGUMENTS

-nosplit
Prevents line splitting and facilitates writing software to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

DESCRIPTION

The **report_annotated_transition** command lists annotated transition times at every pin in the current design.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following is an example of an annotated transition time report.

```
prompt> report_annotated_transition
```

```
report_annotated_transition
```

Pin Name	max_rise	max_fall	min_rise	min_fall
U0/A	10.00	-	-	17.00
U1/A	-	-	-	9.10
U1/Z	-	-	-	7.00

1

SEE ALSO

```
remove_annotated_transition(2)  
reset_design(2)  
set_annotated_transition(2)
```

report_antenna_ratio

Runs the antenna checker in the router and dumps a detailed report of antenna violations into the log file.

SYNTAX

integer **report_antenna_ratio**

ARGUMENTS

The **report_antenna_ratio** command has no arguments.

DESCRIPTION

The **report_antenna_ratio** command runs the antenna checker in the router of an open cell and dumps a detailed report of antenna violations into the log file.

No options are provided.

The log file should provide reports similar to the following:

```
Antenna Violation for Net flmod_0/l1extflops_0/n2598 Port [flmod_0/l1extflops_0/
U647,I] GateArea = 0.19 Layer [METAL2] sw-ratio = 559.95 (over 395) Antenna
Violation for Net flmod_0/l1extflops_0/io_PB1_Dout_Q1378_57_ Port [flmod_0/
l1extflops_0/U641,I] GateArea = 0.19 Layer [METAL4] sw-ratio = 1455.07 (over 395)
Antenna Violation for Net flmod_0/l1extflops_0/n2671 Port [flmod_0/l1extflops_0/
U639,I] GateArea = 0.19 Layer [METAL4] sw-ratio = 515.66 (over 395) 'Top-layer
Antenna' Violation for Net flmod_0/sl1extflops_0/n611 Total GateArea = 0.36 Layer
[METAL5] sw-ratio = 3426.94 (over 2670.56)
```

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

```
report_antenna_ratio
```

report_antenna_rules

Reports all of the antenna rules stored in the library.

SYNTAX

```
status_value report_antenna_rules  
[mw_lib]  
[-output file_name]
```

Data Types

<i>mw_lib</i>	list
<i>file_name</i>	string

ARGUMENTS

mw_lib
Specifies the Milkyway library to be updated. The value of *mw_lib* can be a library name or a one-element collection of a library. The *mw_lib* option is optional. The default is to use the current Milkyway library.

-output *file_name*
The name of a specified output file.

DESCRIPTION

This command reports all of the antenna rules stored in the library to a file or to the window. The command returns a status indicating success or failure.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

```
prompt> report_antenna_rules
```

SEE ALSO

```
define_antenna_layer_ratio_scale(2)  
define_antenna_layer_rule(2)  
define_antenna_rule(2)  
remove_antenna_rules(2)
```

report_aocvm

Displays information about advanced on-chip variation (OCV) derate factor tables and coefficients, and displays details of the advanced OCV derate calculation.

SYNTAX

```
status report_aocvm
[-early] [-late]
[-rise] [-fall]
[-cell_delay] [-net_delay]
[-list_annotated] [-list_not_annotated]
[-nosplit]
[-arc_details]
[object_list]
```

Data Types

object_list list

ARGUMENTS

-early
Reports early derate tables.

-late
Reports late derate tables.

-rise
Reports rise derate tables.

-fall
Reports fall derate tables.

-cell_delay
Reports cell delay derate tables.

-net_delay
Reports net delay derate tables.

-list_annotated
Generates a list of leaf cells and global nets that are annotated with advanced OCV derate tables.

-list_not_annotated
Generates a list of leaf cells and global nets that are not annotated with advanced OCV derate tables.

-nosplit
Prevents split lines when columns overflow.

-arc_details
Reports the computed depth and derate values of the timing arcs that start,

pass through, or end at the cells listed in *object_list*.

object_list

Specifies the design objects to be reported, on which advanced OCV derate tables or coefficients have been annotated. Only cells should be specified when used with the **arc_details** option.

DESCRIPTION

Displays user-specified advanced on-chip variation (OCV) information. The type of information displayed in the report depends on the type of advanced OCV information that has been annotated.

If the design has been annotated with advanced OCV derate tables using the **read_aocvm** command, then the **report_aocvm** command generates a summary showing the numbers of leaf cells and global nets annotated with advanced OCV derate tables. Lists of fully annotated, partially annotated, and not annotated leaf cells and global nets may be displayed by using the **-list_annotated** and **-list_not_annotated** options.

If the design has also been annotated with advanced OCV derate coefficients using the **set_aocvm_coefficient** command, then the **report_aocvm** command also displays these coefficients.

If the *arc_details* option is used along with *object_list*, then metrics for the arcs of those cells are displayed. Note that the depths shown in the report are scaled by any advanced OCV random coefficients that have been set.

EXAMPLES

In the following example, the design has been annotated with advanced OCV derate factors using the **read_aocvm** command.

```
prompt> report_aocvm
*****
Report : aocvm
Design : my_design
*****
```

	Total	Fully annotated	Partially annotated	Not annotated
Leaf cells	6	0	4	2
Nets	7	0	0	7
	13	0	4	9

1

The following example displays metrics and advanced OCV derate factors for a cell specified in the *object_list*. The arc depth displayed in the report is the most

```
report_aocvm
1650
```


pessimistic depth for all possible paths through the arc.

```
prompt> report_aocvm -arc_details [get_cell u4]
```

```
*****
```

```
Report : aocvm  
        object_set
```

```
Design : top
```

```
*****
```

```
From pin: u4/A  
To pin:   u4/Z  
Arc type: cell (clock network)
```

AOCVM arc metrics	Launch	Capture
Depth	10.50	10.50

AOCVM arc derates	Launch	Capture
Early rise	0.9145	0.9145
Early fall	0.9043	0.9043
Late rise	1.0890	1.0890
Late fall	1.0998	1.0998

```
From pin: CLK  
To pin:   u4/A  
Arc type: net (clock network)
```

AOCVM arc metrics	Launch	Capture
Depth	5.00	5.00

AOCVM arc derates	Launch	Capture
Early rise	0.9032	0.9032
Early fall	0.8940	0.8940
Late rise	1.1010	1.1010
Late fall	1.1106	1.1106

```
From pin: u4/Z  
To pin:   u5/A  
Arc type: net (clock network)
```

AOCVM arc metrics	Launch	Capture
Depth	5.00	5.00

AOCVM arc derates	Launch	Capture
Early rise	0.9032	0.9032
Early fall	0.8940	0.8940
Late rise	1.1010	1.1010
Late fall	1.1106	1.1106

SEE ALSO

```
read_aocvm(2)
remove_aocvm(2)
report_timing(2)
set_aocvm_coefficient(2)
timing_aocvm_enable_analysis(3)
```

report_area

Displays area information for the current design or instance.

SYNTAX

```
integer report_area  
[-nosplit]  
[-physical]  
[-hierarchy]
```

ARGUMENTS

-nosplit

Prevents line-splitting and facilitates writing software to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

-physical

Reports the size of the core area and the aspect ratio of the design.

-hierarchy

Reports the area used by cells across the design hierarchy. Reports the absolute value and the percentage of area consumed by each of the cells across the hierarchy. This option also reports the details of area contribution by combinational, non-combinational, and black box cells.

DESCRIPTION

The **report_area** command lists current instance or current design statistics including combinational, non-combinational, and total area. If you set the **current_instance** command, the report is generated for the design of that instance. Otherwise the report is generated for the current design.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example generates an area report:

```
prompt> report_area  
  
*****  
Report : area  
Design : CONTROL  
Version: v2.0
```

Date : Fri Mar 16 14:09:12 1991

Library(s) Used:

tech_lib (File: /usr/synopsys/libraries/tech_lib.db)

Number of ports: 33
Number of nets: 50
Number of cells: 22
Number of references: 8

Combinational area: 110.00
Noncombinational area: 64.00
Net Interconnect area: 0.00

Total area: 174.00

Information: This design contains unmapped logic. (RPT-7)

Information: This design contains black box (unknown) components. (RPT-8)

The following example generates an area report for a hierarchical design using the -
hierarchy option:

prompt> **report_area -hierarchy**

Report : area
Design : top
Version: Y-2006.06
Date : Mon Dec 12 04:25:09 2005

Library(s) Used:

lsi_10k (File: /usr/synopsys/libraries/tech_lib.db)

Number of ports: 13
Number of nets: 37
Number of cells: 11
Number of references: 4

Combinational area: 16.000000
Noncombinational area: 168.000000
Net Interconnect area: undefined (No wire load specified)

Total cell area: 184.000000
Total area: undefined

Hierarchical area distribution

Global cell area		Local cell area		
-----		-----		
Absolute	Percent	combi-	noncombi-	black

Hierarchical cell	Total	Total	national	national	boxes	Design
-----	-----	-----	-----	-----	-----	-----
top	184.0000	100.0	16.0000	0.0000	0.0000	top
mid1	56.0000	30.4	0.0000	0.0000	0.0000	mid_0
mid1/low1	28.0000	15.2	0.0000	28.0000	0.0000	low_0
mid1/low2	28.0000	15.2	0.0000	28.0000	0.0000	low_5
mid2	56.0000	30.4	0.0000	0.0000	0.0000	mid_2
mid2/low1	28.0000	15.2	0.0000	28.0000	0.0000	low_4
mid2/low2	28.0000	15.2	0.0000	28.0000	0.0000	low_3
mid3	56.0000	30.4	0.0000	0.0000	0.0000	mid_1
mid3/low1	28.0000	15.2	0.0000	28.0000	0.0000	low_2
mid3/low2	28.0000	15.2	0.0000	28.0000	0.0000	low_1
-----	-----	-----	-----	-----	-----	-----
Total			16.0000	168.0000	0.0000	

SEE ALSO

report_design(2)
set_max_area(2)

report_attribute

Reports the attributes on one or more objects.

SYNTAX

```
string report_attribute  
[-application]  
[-class class_name]  
[-quiet]  
[object_list]
```

Data Types

<i>class_name</i>	string
<i>object_list</i>	list

ARGUMENTS

-application
Lists application attributes and user-defined attributes.

-class *class_name*
Specifies the class name for the object specified in *object_list*, if the element of *object_list* is a name. Valid classes are design, port, cell, net, and so forth.

-quiet
Indicates that all error and warning messages are not to be reported.

object_list
Specifies a list of objects to report. Each element in the list is either a collection or a pattern that is combined with the *class_name* to find the objects.

DESCRIPTION

The **report_attribute** command generates a report of attributes on the specified objects. By default, only user-defined attributes display.

Using the **-application** option adds application attributes to the report.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports the attributes on a specified net object:

```
report_attribute  
1656
```

```
prompt> report_attribute [get_nets clk2] -app
```

```
*****
Report : Attribute
LIBRARY: astro_tcl
DESIGN : top
Date   : Wed Sep  8 10:26:17 2004
*****
```

Design	Object	Type	Attribute Name	Value
top	clk2	string	base_name	clk2
top	clk2	boolean	dont_touch	false
top	clk2	boolean	ideal_net	false
top	clk2	boolean	is_physical	false
top	clk2	boolean	is_tie_high_net	false
top	clk2	boolean	is_tie_low_net	false
top	clk2	string	net_type	Clock
top	clk2	int	num_overall_pins	2
top	clk2	int	num_pins	2
top	clk2	string	owner	top
top	clk2	string	route_length	{{METAL3 28.000} {METAL2 8.110}}
top	clk2	int	cell_id	3
top	clk2	string	full_name	clk2
top	clk2	string	mw_name	clk2
top	clk2	string	name	clk2
top	clk2	int	number_of_wires	8
top	clk2	string	object_class	net
top	clk2	int	object_id	2816

The following example limits the reporting to user-defined attributes only:

```
prompt> define_user_attribute -class net attr_s -type string
Info:User-defined attribute 'attr_s' on class 'net'.
1
prompt> set_attribute -class net clk2 attr_s hello
Info:Setting attribute 'attr_s' on net 'clk2'.
{"clk2"}
prompt> report_attribute -class net clk2
```

```
*****
Report : Attribute
LIBRARY: astro_tcl
DESIGN : top
Date   : Wed Sep  8 10:41:31 2004
*****
```

Design	Object	Type	Attribute Name	Value
top	clk2	string	attr_s	hello

SEE ALSO

`define_user_attribute(2)`
`get_attribute(2)`
`remove_attribute(2)`
`report_attribute(2)`
`report_cell(2)`
`report_design(2)`
`report_net(2)`
`report_port(2)`
`report_reference(2)`
`report_timing(2)`
`set_attribute(2)`

report_bounds

Reports bounds in the design.

SYNTAX

```
int report_bounds  
-all | bound | -name name_list
```

Data Types

<i>bound</i>	list
<i>name_list</i>	list

ARGUMENTS

-all
Specifies to report all bounds that were created with `create_bounds`. Bounds from an input PDEF file will not be reported with this command.

bound
Specify the bound objects. The bound objects can be specified by the command `get_bounds`.

-name *name_list*
Reports bounds with the specified names.

DESCRIPTION

The **report_bounds** command generates reports on the user-specified bounds constraints in the design, as specified by the **create_bounds** command. If **-all** is used, all bounds in the design created by **create_bounds** will be reported. Bounds created by an input PDEF file will not be reported. If a list of bound names is specified, those bounds will be reported. This report includes the bound name, the type of the bound, the effort level, and the list of cells that belong to the bound.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports bounds of names "foo_1" and "foo_2".

```
prompt> report_bounds -name {foo_1 foo_2}  
Report bounds:  
*****  
Bound Name: foo_1, type: high effort auto group bound  
Cells: a b c d  
Bound Name: foo_2, type: soft moving bound {10 20 30 40}
```

```
Cells: e f
*****
```

SEE ALSO

```
create_placement(2)
create_bounds(2)
update_bounds(2)
set_cell_location(2)
remove_bounds(2)
get_bounds(2)
```

report_buffer_tree

Reports the buffer tree and its level information at the given driver pin.

SYNTAX

```
status report_buffer_tree  
[-from start_point_list | -net net_list]  
[-depth max_depth]  
[-connections]  
[-hierarchy]  
[-physical]  
[-nosplit]
```

Data Types

<i>start_point_list</i>	list
<i>net_list</i>	list
<i>max_depth</i>	int

ARGUMENTS

-from *start_point_list*
Specifies the starting point of the buffer tree. The starting point is defined as level 0 of the tree. The *start_point_list* must contain only pins or ports.

-net *net_list*
Specifies a net as the starting point of the buffer tree. The *net_list* must contain only nets.

-depth *max_depth*
Reports only the *max_depth* levels of buffers in the tree. The default is to report the buffer tree until a real load or hierarchical boundary is reached.

-connections
Shows the net connections in the report. By default, the net connections are not shown.

-hierarchy
Shows the buffer tree across the hierarchy. By default, the report terminates at hierarchical boundaries.

-physical
Displays the location for each pin, with the coordinates in microns.

-nosplit
Prevents linesplitting, making it easier to write scripts to extract information from the report output. By default, the design information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

DESCRIPTION

The **report_buffer_tree** command reports a buffer tree at the given driver pins or driver nets. It reports the full name of the driver pin (which includes the name of the cell), the name of the library cell, and the level of the driver relative to the starting point. The starting point is defined as level 0 of the tree.

Reporting terminates at the non-buffer pins, hierarchical boundaries (except when the **-hierarchy** option is used), or when the level reported exceeds the value specified by *max_depth*.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows how to report a buffer tree:

```
prompt> report_buffer_tree -from cell1/0
```

The following example shows how to report a buffer tree with net connections:

```
prompt> report_buffer_tree -connections -from cell1/0
```

SEE ALSO

```
all_fanout(2)  
create_buffer_tree(2)  
remove_buffer_tree(2)  
report_constraint(2)  
report_hierarchy(2)  
report_net_fanout(2)
```

report_buffer_tree_qor

Displays quality related properties of the buffer trees at the given driver pins.

SYNTAX

```
int report_buffer_tree_qor  
[-from list_of_driving_pins_or_nets]
```

Data Types

list_of_driving_pins_or_nets list

ARGUMENTS

-from *list_of_driving_pins_or_nets*
Specifies the starting points of the buffer trees to be reported.

DESCRIPTION

The **report_buffer_tree_qor** command reports quality related properties of the buffer trees at given driver pins or driver nets. The report is presented as a spreadsheet containing the following columns: 1) Number of sinks (non-buffer loads); 2) Number of buffers and inverters; 3) Number of buffer levels; 4) Buffers' area; 5) Number and value of design rule violations - max_tran, max_cap and max_fanout 6) Total negative slack on the sinks; 7) Worst slack among the sinks; 8) Maximum immediate fanout in the tree (max. sub-tree fanout) 9) Name of the driving pin

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows how to report quality of all buffer trees with number of sinks exceeding 100.

```
prompt> report_buffer_tree_qor \ -from [all_high_fanout -nets -threshold 100  
-through]  
=====
```

# of	# of	# of	max	buf.	MT violation	MC violation	MF vi
olation		Wrst	Max	source			
sinks	buff.	inv.	lvl	area	#	value	#
: value	TNS	slck	sbtr	pin			
2023:	0:	0:	0:	0:	0:	0.00:	0
:	0.00:	0.0:	**	2023: SysClk			
2023:	0:	0:	0:	0:	0:	0.00:	0

```
=====
```

```

: 0.00: 0.0: ** : 2023: SliceReset_
   136: 9: 9: 3: 73: 0: 0.00: 0: 0.00: 0
: 0.00: 10.0: -0.12: 20: Egress/EPortShiftReg/U464/Z
   744: 53: 2: 4: 418: 0: 0.00: 0: 0.00: 0
: 0.00: 177.1: -0.49: 20: EnMgr/egressShiftEn
-----
   4926: 62: 11: 4: 490: 0: 0.00: 0: 0.00: 0
: 0.00: 187.1: -0.49: 2023: =TOTAL=
=====
=====

```

SEE ALSO

```

all_high_fanout(2)
create_buffer_tree(2)
remove_buffer_tree(2)
report_net_fanout(2)
report_constraint(2)

```

report_bus

Lists the bused ports and nets in the current instance or in the current design.

SYNTAX

```
integer report_bus  
[-nosplit]  
[-cell cel_name]  
[object_list]
```

ARGUMENTS

-nosplit

Prevents line-splitting and facilitates writing software to extract information from the output report. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

-cell *cel_name*

Only the bus related to the specified cell is reported. If the specified cell is a hierarchical cell, the port bus of the cell and net bus enclosed inside this cell are reported. If the specified cell is a leaf cell, the port bus of the leaf cell is reported.

This argument is optional. This argument supersedes *object_list* if both are selected.

object_list

Specifies a list of bus ports or bus nets to be reported.

This argument is optional. This argument is ignored if **-cell** option is selected.

DESCRIPTION

This command displays information about buses in the current instance or the current design. If the current instance has been set, the report is generated for the design of that instance. Otherwise, the report is generated for the current design.

Buses are indexed in an ascending order in the report. If a bus is named in a descending order, the numbers in the "Step" column are negative. Bused pins are not listed in the report because the bus commands operate only on design ports (not reference ports).

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command displays bus information on the current design:

```
prompt> report_bus
```

```
Hier cell: SUBTRACTOR
```

Bussed Port	Dir	From	To	Step	Width	Type
A	in	0	3	1	4	array
B	in	0	3	1	4	array
DIFF	out	0	3	1	4	array

Bussed Net	From	To	Step	Width	Type
A	0	3	1	4	array
B	0	3	1	4	array
DIFF	0	3	1	4	array

SEE ALSO

```
define_bus(2)  
undefine_bus(2)
```


report_case_analysis

Reports case analysis on ports or pins.

SYNTAX

```
string report_case_analysis  
[-all] [-nosplit]
```

ARGUMENTS

-all

Specifies to report the built-in constant pins of the design that are considered for startpoints of constant propagation. Propagation of logic constants is only done if the **set_case_analysis** command was executed, or the variable **case_analysis_with_logic_constants** or **disable_case_analysis** is set to true.

-nosplit

Prevents line splitting and facilitates writing software to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

DESCRIPTION

The **report_case_analysis** command reports case analysis entries specified with the **set_case_analysis** command. The report lists all pins and ports on which case analysis is specified. The list does not report where the logic constant values are propagated.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example specifies that pins U1/U2/A and U1/U3/CI are set to a constant of logic 1.

```
prompt> set_case_analysis 1 {U1/U2/A U1/U3/CI}  
prompt> report_case_analysis
```

```
*****  
Report : case_analysis  
Design : middle  
Version: 1997.01-development  
Date   : Mon Jul 22 17:04:17 1996  
*****
```

	Pin name	User case analysis value

-	U1/U2/A	1
	U1/U3/CI	1

SEE ALSO

[remove_case_analysis\(2\)](#)
[set_case_analysis\(2\)](#)

report_cbt_options

Reports options used by the **create_buffer_tree** command.

SYNTAX

```
int report_cbt_options
```

ARGUMENTS

The **report_cbt_options** command has no arguments.

DESCRIPTION

The **report_cbt_options** command reports options used by the **create_buffer_tree** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

SEE ALSO

```
create_buffer_tree(2)  
set_cbt_options(2)
```

report_cell

Displays information about cells in the current instance or in the current design.

SYNTAX

```
status report_cell
[-nosplit]
[-connections]
[-verbose]
[-physical]
[-only_physical]
[-significant_digits digits]
[cell_list]
```

Data Types

<i>digits</i>	integer
<i>cell_list</i>	list

ARGUMENTS

-nosplit
Prevents line-splitting and facilitates writing software to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

-connections
Displays all pins of the cell and the connected nets.

-verbose
Displays verbose connection information. For each input pin, the tool displays the net and the driver pins on that net. For each output pin, the tool displays the net and the load pins on that net.

-physical
Reports the orientation and location of the cell.

-only_physical
Reports information for physical only cells.

-significant_digits *digits*
Specifies the number of digits to the right of the decimal point that are reported. Allowed values are 0 through 13 and the default is 6. Using this option overrides the value set by the **report_default_significant_digits** variable.

cell_list
Specifies a list of cells and instances to show in the report. If not specified, all cells in the current instance are listed.

DESCRIPTION

This command displays information and statistics about cells in the current instance or current design. If *current_instance* is set, the report is generated for the design of that instance; otherwise the report is generated for the current design.

Parameterized cells are attributed by the letter *p* and contain parameters that control aspects of the designs they instantiate. The actual values of a cell's parameters appear at the bottom of the report.

Control logic cells are designated as type *c* in the report generated by **report_cell**. Use this information to identify control logic cells when using the **group** command. Group control logic cells with the cells that they control.

This command does not display constant cells. To display constant cells, use the following command:

```
get_cell -hier * -filter "@ref_name==**logic_0** || @ref_name==**logic_1**"
```

Use the **-connections** option to list the nets connected to each pin.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following is an example of a cell report:

```
prompt> report_cell

*****
Report : cell
Design : CONTROL
Version: v3.1
Date   : Fri Mar 20 14:09:15 1991
*****

Attributes:
  b - black box (unknown)
  bo - allows boundary optimization
  BO - reference allows boundary optimization
  c - control logic
  d - dont_touch
  D - reference or design is dont_touched
  h - hierarchical
  n - noncombinational
  p - parameterized
  r - removable
  S - synthetic module
  u - contains unmapped logic
```

Cell	Reference	Library	Area	Attributes

IDIN0	NAND2	tech_lib	1.00	
IDIN1	JPMP		0.00	h, u
IDIN2	JMM		0.00	b
IOUT1	NAND2	tech_lib	1.00	
IOUT2	NAND2	tech_lib	1.00	
IOUT3	NAND2	tech_lib	1.00	
R1	NAND3	tech_lib	2.00	
R2	INV	tech_lib	1.00	
U0	MX1P	tech_lib	8.00	n
U1	MX1P	tech_lib	8.00	d, n, 1
U3	MX1P	tech_lib	8.00	d, n
U5	MX1P	tech_lib	8.00	n
U6	MX1P	tech_lib	8.00	n, 2
U7	MX1P	tech_lib	8.00	n
U9	JMM1		0.00	b
U10	JMM1		0.00	b
U15	CTLX		101.00	h
P1	DW01_add		50.00	S, p

Total 18 cells			206.00	

Flip-Flop Types:

- 1 - Default type MX1, MX1P, MX2, MX2P, MX3, MX3P, MX4, MX4P
- 2 - Exact type LMN01

HDL Parameters:

P1 - width => 16

The following is an example of a verbose cell connection report for the *t_bar* cell:

```
prompt> report_cell -connections -verbose {t_bar}
```

```
*****
```

```
Report : cell
```

```
    -connections
```

```
    -verbose
```

```
Design : counter
```

```
Version: v3.1
```

```
Date   : Fri 1993
```

```
*****
```

```
Connections for cell 't_bar':
```

```
Reference:      IVA
Library:        lsi_10k
Area:           1
dont_touch:     FALSE
```

Input Pins	Net	Net Driver Pins	Driver Pin Type

A	t	t/Z	Output Pin (EO)

	Output Pins	Net	Net Load Pins	Load Pin Type
	-----	-----	-----	-----
1	Z	t_bar	zero/B	Input Pin (OR2)

The following command reports on all of the registers in the current design:

```
prompt> report_cell all_registers()
```

SEE ALSO

```
all_registers(2)
current_design(2)
current_instance(2)
report_design(2)
report_hierarchy(2)
report_net(2)
report_reference(2)
report_default_significant_digits(3)
```

report_cell_physical

Reports information about the specified cells.

SYNTAX

```
status report_cell_physical  
[-connections [-verbose]]  
[cell_list]
```

Data Types

cell_list collection

ARGUMENTS

-connections

Reports all pins of the cells and the connected nets. Connections to power and ground pins are reported separately. For each power and ground pin, the report includes the pin name and the connected nets.

-verbose

Reports the connection information in verbose mode. For each input pin, the tool reports the net and the driver pins on the corresponding net. For each output pin, the tool reports the net and the load pins on the corresponding net.

This option is valid only when you also specify the **-connections** option. Verbose information is not reported for power and ground pins.

cell_list

Specifies the cells and instances to be reported. If not specified, all cells in the current instance are reported.

DESCRIPTION

This command reports physical information about the specified cells. If the **current_instance** command is set, the tool generates a report for the design of that instance. Otherwise, the tool generates a report for the current design.

By default, the command reports cell instance information, such as cell name, parent name, reference name, library name, site name, bounding box, width, height, location, origin, logical area, physical area, keepout margins, orientation, and names of voltage area.

Use the **-connections** option to list the nets connected to each pin.

EXAMPLES

The following is an example of a cell report.

```
prompt> report_cell_physical U17
```

```
report_cell_physical  
1674
```

Report : cell
Version : C-2009.06
Cell Name : mydesign.CEL;1
Date : Fri Mar 13 02:33:19 2009

Attribute Name	Value
name	U17
full_name	U17
design	mydesign
object_id	81928
cell_id	6
number_of_pins	2
object_class	cell
ref_name	INVXLMTL
ref_lib_name	/usr/mylibs/MW/refLib
ref_view_name	FRAM
is_hierarchical	false
is_preserved	false
ref_view_name	FRAM
bbox	{419.600 402.800} {420.800 405.600}
origin	420.800 405.600
boundary	{420.800 405.600} {419.600 405.600} \ {419.600 402.800} {420.800 402.800} \ {420.800 405.600}
area	3.360000
is_physical_only	false
aspect_ratio	2.333333
height	2.800000
width	1.200000
orientation	S
mask_layout_type	std
movebound	
site_name	unit
location	419.600 402.800
Keepout margin	Does not exist
Voltage area name	N/A
sm_estimation_mode	unestimated
is_fixed	false
is_io	false
is_jtag	false
is_physical	true
is_placed	true
is_spare_cell	false
is_soft_fixed	false

The following is an example of reporting the cell connections in verbose mode for the II_ALU/U1 cell:

```
prompt> report_cell_physical 1 I_ALU/U1 -connections -verbose
*****
```

```
Report : cell
        -connections
        -verbose
Version : C-2009.06
Cell Name : top.CEL;1
Date : Fri Mar 13 00:21:36 2009
```

```
*****
```

Input Pins	Net	Net Driver Pins	Driver Pin Type
-----	-----	-----	-----
A	I_ALU/Oprnd_A[1]	I_DATA_PATH/Oprnd_A_reg_1_/Q	Output pin (SEDFHGX4)
Output Pins	Net	Net Load Pins	Load Pin Type
-----	-----	-----	-----
Y	I_ALU/U1Net33	I_ALU/sub_87_minus/U2_1/A	Input pin (ADDFHX4)
PG Pins	Net		
-----	-----		
VSS			
VDD			

SEE ALSO

```
current_instance(2)
report_design(2)
report_hierarchy(2)
report_net(2)
report_reference(2)
report_default_significant_digits(3)
report_design_physical(2)
```

report_cell_vt_type

prints out voltage threshold type of a cell master or all cell masters of a library. Voltage threshold type is used for mixed voltage threshold filler cell insertion

SYNTAX

```
integer report_cell_vt_type  
-library library_name | -lib_cell cell_name
```

Data Types

cell_name string

ARGUMENTS

```
-library library_name | -lib_cell cell_name  
    Use -library to specify a library name or use -lib_cell to specify a master  
    cell name.
```

DESCRIPTION

Use this command to print out vt type string of a master cell or master cells of a library.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example will prints out vt type settings from all master cells of library RefLib1

```
prompt> report_cell_vt_type -library "RefLib1"
```

Below shows the output that says library cell CELL1 in library RefLib1 are set to vt type "vtType2": ## Voltage threshold type of cells in library RefLib1
set_cell_vt_type -lib_cell "CELL1" -vt_type "vtType2"

SEE ALSO

```
set_cell_vt_type(2)  
remove_cell_vt_type(2)  
insert_stdcell_filler(2)  
set_vt_filler_rule(2)
```

report_change_list

Reports the ECO changes after running incremental optimization.

SYNTAX

```
status report_change_list
```

ARGUMENTS

There are no arguments to this command.

DESCRIPTION

The **report_change_list** command reports changes to cells in a design. If there are changes to nets (pin swapping), only new nets that are added can be detected.

To turn off this feature, set the **physopt_change_list** variable to FALSE before running the **physopt -incremental** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example runs the **report_change_list** command.

```
prompt> report_change_list
```

```
report_change_list
```

```
Information: Updating design information... (UID-85)
```

Status	Cell name	Reference name
deleted	U3196	buf1a3
orientation changed	U21	or2c3
location changed	U6164	or2b2
net changed	U4169	mx2a3
modified	U4546	buf1a6
new	U6387	xor2a6

```
Total cells changed: 6
```

SEE ALSO

```
physopt_change_list(3)
```

report_check_library_options

Reports the values or the status of the options set by the **set_check_library_options** command. The **report_check_library_options** command provides information about options used for checking between logical libraries, options used for checking between physical libraries, and options used for checking between logical libraries and physical libraries.

SYNTAX

```
status report_check_library_options  
[-physical]  
[-logic_vs_physical]  
[-logic]  
[-default]
```

ARGUMENTS

-physical

Reports the values or the status of the options set by the **set_check_library_options** command to check between physical libraries.

-logic_vs_physical

Reports the values or the status of the options set by the **set_check_library_options** command to check between logical libraries and physical libraries.

See the **-logic_vs_physical** option in the **set_check_library_options** man page for more information.

-logic

Reports the values or the status of the options set by the **set_check_library_options** command to check between logical libraries. See the **-logic** option in the **set_check_library_options** man page for more information.

-default

Reports the default values of all the library-checking options. If this option is not specified, the current values of the options are reported.

DESCRIPTION

The **report_check_library_options** command reports the values or the status of the options set by the **set_check_library_options** command. The **report_check_library_options** command provides information about options used for checking between logical libraries, options used for checking between physical libraries, and options used for checking between logical libraries and physical libraries.

Use the **report_check_library_options** command after running **set_check_library_options**.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

In the following example, **report_check_library_options** checks all data for the specified library, test.mw, and then reports the option values:

```
prompt> set_check_library_options -cell_footprint -bus_delimiter
```

```
prompt> report_check_library_options -logic_vs_physical
```

```
1
```

```
Report options for check_library
```

```
*****
```

Check cell area	: false
Check cell footprint	: true
Check bus delimiter	: true
Cell names	: (null)
Logic libraries	: (null)
Physical libraries	: (null)

```
1
```

SEE ALSO

check_library(2)

set_check_library_options(2)

report_clock

Displays clock-related information on the current design.

SYNTAX

```
status report_clock  
[-attributes] [-skew] [-nosplit] [-groups]  
[-scenario]
```

ARGUMENTS

- attributes**
Provides a list of all the clocks in the current design. The information for each clock includes: source type, signal rise and fall times, and attributes. This report is the default.
- skew**
Reports the clock network skew information set on the design by the **set_clock_skew** command. This information includes rise delay, fall delay, plus uncertainty and minus uncertainty of the clock network.
- nosplit**
Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column. This option prevents line-splitting and facilitates writing software to extract information from the report output.
- groups**
Reports the clock groups information set on the design by the **set_clock_groups** command.
- scenario *scenario_list***
Reports clocks for given list of scenarios of a multiscenario design. Inactive scenarios will be skipped in the report. Each scenario is reported separately. If this option is not given, only the current scenario is reported.

DESCRIPTION

Displays all clock-related information on a design. The report contents are controlled by the options used. If you don't specify an option, the **attributes** report is displayed.

The **report_transitive_fanout -clock_tree** report shows the fanout network of every clock source in the design.

To obtain a list of all clocks in the design, use the **all_clocks** command.

Multicorner-Multimode Support

By default, this command uses information from the current scenario. You can select different scenarios by using the **-scenario** option.

EXAMPLES

The following are examples of clock reports.

```
prompt> report_clock -attributes
```

```
***** Report : clocks Design : top Version: v2.3
Date : Mon 1992 *****
```

Attributes: d - dont_touch_network p - propagated_clock G - Generated clock

```
Clock Period Waveform Attrs Sources -----
----- phi1 10.00 {0 5} {phi1} phi2 10.00 {5 10} {phi2}
off_chip_clk 50.00 {0 25} {} -----
-----
```

```
prompt> report_clock -skew
```

```
***** Report : clock_skew Design : top Version:
v2.3 Date : Mon 1992 *****
```

```
Rise Fall Plus Minus Object Delay Delay Uncertainty Uncertainty -----
----- phi1 0.50 0.40 0.20
0.10 phi2 0.20 - - - ff1/CP - - 0.10 0.15 ff2/CP - - 0.10 0.15 ff3/CP - - 0.10 0.15
ff4/CP - - 0.10 0.15
```

```
prompt>report_clock -groups
```

```
***** Report : clocks Design : test Version:
2006.06 Date : Wed *****
```

Attributes: d - dont_touch_network f - fix_hold p - propagated_clock G - generated_clock

```
Clock Period Waveform Attrs Sources -----
----- clk1 4.00 {0 2} {clk1} clk2 4.00 {0 2} {clk2} clk3
4.00 {0 2} {clk3} clk4 4.00 {0 2} {clk4} -----
-----
```

Clock Groups :

Total logically exclusive groups: 1 NAME : clk4_clk2 False timing paths. -groups {clk4} -groups {clk2}

Total physically exclusive groups: 1 NAME : clk3_1 False timing paths. -groups {clk3} -groups {clk4}

Total asynchronous groups: 1 NAME : clk1_clk2 False timing paths. -groups {clk1} -groups {clk2 clk4}

1

SEE ALSO

all_clocks(2)
create_clock(2)
remove_clock(2)
report_constraint(2)
report_design(2)
report_transitive_fanout(2)
set_clock_groups(2)

report_clock_gating

Reports information about clock gating performed by Power Compiler.

SYNTAX

```
status report_clock_gating
[-no_hier]
[-verbose]
[-gated]
[-ungated]
[-gating_elements]
[-only cell_list]
[-nosplit]
[-physical]
[-multi_stage]
[-style]
[-structure]
[-scenario scenario_list]
```

Data Types

```
cell_list          list
scenario_list      list
```

ARGUMENTS

- no_hier**
Reports clock-gating information for only the top level of the current design. Without this option, all levels of hierarchy starting from the current design are reported.
- verbose**
Reports more detailed information; for example, test information and a list of related registers to the clock-gating cells. By default, this information is not included in the report.
- gated**
Reports the names of all gated registers, along with the names of the corresponding clock-gating elements.
- ungated**
Reports the names of all ungated registers, along with the unmet clock-gating conditions.
- gating_elements**
Reports the names of all clock-gating elements, along with their style, setup and hold times, inputs, and outputs.
- only *cell_list***
Reports information only for the clock-gating cells or gated and ungated registers listed in *cell_list*. By default, information is displayed for all clock-gating cells and ungated and gated registers.

- nosplit
Prevents line-splitting and facilitates writing software to extract information from the report output. Most design information is listed in fixed-width columns. If information for a field exceeds the column width, the next field begins on a new line, starting in the correct column.
- physical
Reports the physical locations and distance statistics of gating cells and gated registers, along with other clock-gating information. This option can be used only with the **-gating_elements** option.
- multi_stage
Reports multistage clock-gating statistics in the clock-gating summary.
- style
Reports the clock-gating style of all clock gates in the current design.
- structure
Produces the Clock Gating Structure Summary and Clock Gating Structure Details reports, where the information about clock-gating cells and gated registers is classified according to the clock domains to which they belong. Additional information about fanout and clock latency set on each clock-gating element is also displayed.
This option can only be combined with the **-scenario** option.
- scenario *scenario_list*
Produces a set of reports for the specified list of scenarios of a multiscenario design, when used with the **-structure** option. To get reports for all active scenarios, use "[all_scenarios]" as the *scenario_list* argument.
If this option is not specified, only the current scenario is reported.

DESCRIPTION

The **report_clock_gating** command reports information about clock-gating cells and gated and ungated registers of the current design. To generate the report, **report_clock_gating** uses clock-gating attributes placed on the clock-gating cells by Power Compiler. Any clock gating added by other means (for example, clock gating inferred by or instantiated in the RTL code) lacks these attributes and does not appear in the report.

When the command is issued without options, the report lists the number of clock-gating elements, the number of gated and ungated registers with percentages of the total, and the total number of registers.

When the **-multi_stage** option is specified, the summary also includes multistage clock-gating statistics. These statistics include the number of multistage clock-gating elements (for example, clock gates that have other clock gates in their fanout), the average multistage fanout, the number of gated cells (registers and modules), and the maximum and average number of stages per gated cell. Use one or more of the other options to obtain more detailed information.

When issued with the **-structure** option, the command generates the Clock Gating Structure Summary and Clock Gating Structure Details reports. The Clock Gating

Structure Summary lists, for each clock domain, the total number of registers (gated and ungated), and the number of clock gates and gated elements for each clock gating stage. The Clock Gating Structure Details shows, for each clock domain and each clock gating stage, all clock-gating cells, their respective fanout, clock latency, and gated elements. This option cannot be combined with any other option.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example shows a report generated when the **report_clock_gating** command is issued with no options: The report shows that out of 6 registers, 4 are gated and 2 are ungated.

```
prompt> report_clock_gating
```

```
*****
Report : clock_gating
Design : low_design
Version: 2000.11
Date   : Thu Jun 21 18:52:39 2001
*****
```

Clock Gating Summary		
Number of Clock gating elements	1	
Number of Gated registers	4 (66.67%)	
Number of Ungated registers	2 (33.33%)	
Total number of registers	6	

The following example shows a report generated with the **-multi_stage** and **-no_hier** options for a hierarchical multistage clock-gated design. A multistage clock gate is a clock-gating cell that is driving another clock gating cell. The report shows 3 clock-gating elements, 8 gated and no ungated registers at the top level. Two of the 3 clock gates are multistage, and their average fanout is 1.0, indicating that the clock path consists of a chain of 3 clock gates. There is 1 gated module in addition to the 8 gated registers. The 8 registers have 3 stages on their clock path, but the module has only 2, bringing the average number of stages to $2.9 = ((8*3 + 2*1)/9)$.

```
prompt> report_clock_gating -multi_stage -no_hier
```

```
*****
Report : clock_gating
        -no_hier
        -multi_stage
```

```

Design : top_level
Version: 2003.12
Date   : Tue Oct 21 15:45:26 2003
*****

```

Information : The design is hierarchical. Omit -no_hier option for more than one level.

Clock Gating Summary

Number of Clock gating elements	3
Number of Gated registers	8 (100.00%)
Number of Ungated registers	0 (0.00%)
Total number of registers	8
Number of multi-stage clock gates	2
Average multi-stage fanout	1.0
Number of gated cells	9
Maximum number of stages	3
Average number of stages	2.9

The following example shows a report generated with the **-gating_elements** option. The values of STYLE, MIN, MAX, HOLD, SETUP, and OBS_DEPTH for the gating cell are the default values.

```
prompt> report_clock_gating -gating_elements
```

```

*****
Report : clock_gating
        -gating_elements
Design : low_design
Version: 2000.11
Date   : Thu Jun 21 18:57:56 2001
*****

```

Clock Gating Cell Report

```

-----
Clock Gating Bank : clk_gate_lowout_reg
-----

```

STYLE = latch, MIN = 3, MAX = 2048, HOLD = 0.00, SETUP = 0.00, OBS_DEPTH = 5

```

INPUTS :
    clk_gate_lowout_reg/CLK = clk

```

clk_gate_lowout_reg/EN = n_80

OUTPUTS :

clk_gate_lowout_reg/ENCLK = n45

Clock Gating Summary

Number of Clock gating elements	1
Number of Gated registers	4 (66.67%)
Number of Ungated registers	2 (33.33%)
Total number of registers	6

The following example displays additional information about the gated register and its gating cell:

prompt> **report_clock_gating -gated**

Report : clock_gating
 -gated
Design : low_design
Version: 2000.11
Date : Thu Jul 26 20:41:33 2001

Gated Register Report

Clock Gating Bank	Gated Register
clk_gate_lowout_reg	lowout_reg[3] lowout_reg[2] lowout_reg[1] lowout_reg[0]

Clock Gating Summary

Number of Clock gating elements	1
Number of Gated registers	4 (66.67%)
Number of Ungated registers	2 (33.33%)
Total number of registers	6

The following example displays additional information about ungated registers and the reason they were not gated. The report shows that the registers did not meet the minimum bit width condition for clock gating.

prompt> **report_clock_gating -ungated**

```
*****
Report : clock_gating
        -ungated
Design  : low_design
Version: 2000.11
Date    : Thu Jul 26 21:02:45 2001
*****
```

Ungated Register Report

Ungated Register	enable	ctrl	width	removed
lowout_reg[0]	-	-	NO	-
lowout_reg[1]	-	-	NO	-
lowout_reg[2]	-	-	NO	-
lowout_reg[3]	-	-	NO	-
lowstate_reg[0]	-	-	NO	-
lowstate_reg[1]	-	-	NO	-

Clock Gating Summary

Number of Clock gating elements	0
Number of Gated registers	0 (0.00%)
Number of Ungated registers	6 (100.00%)
Total number of registers	6

The following example shows a report of the physical information for a gating cell with its gated register. The report shows location and distance statistics.

prompt> **report_clock_gating -physical -gating_elements**

```
*****
Report : clock_gating
        -physical
        -gating_elements
Design  : timer
Version: 2001.08
Date    : Thu Jul 26 16:43:04 2001
*****
```

Clock Gating Cell Report

Clock Gating Bank : clk_gate_ccr_reg

STYLE = latch, MIN = 3, MAX = 8, OBS_DEPTH = 5

INPUTS :

clk_gate_ccr_reg/CLK = pclk
clk_gate_ccr_reg/EN = n_156
clk_gate_ccr_reg/TE = ir[3]

OUTPUTS :

clk_gate_ccr_reg/ENCLK = n747

LOCATION = (52.80,0.00)

Gating group:

Bounding box: (18.40, 0.00) (52.80, 32.00)

Max Distance: 46.98 Min Distance: 17.60 Average Distance: 33.84

Register	Distance	Dist./Avg.
ccr_reg[3]	35.06	103.6%
ccr_reg[5]	32.09	94.8%
ccr_reg[4]	46.98	138.8%
ccr_reg[2]	36.91	109.1%
ccr_reg[1]	34.40	101.7%
ccr_reg[0]	17.60	52.0%

Clock Gating Summary

Number of Clock gating elements	12
Number of Gated registers	92 (90.20%)
Number of Ungated registers	10 (9.80%)
Total number of registers	102

The following example shows a report generated with the **-structure** option for a hierarchical multistage clock gated design with 2 clock domains. In this case the field "Maximum number of stages" is also included in the summary printed at the end.

prompt> **report_clock_gating -structure**

Report : clock_gating
-structure

Design : icg_test

Version: B-2008.09

Date : Tue Aug 5 14:41:11 2008

report_clock_gating

1690

Clock Gating Structure Summary

Clock	Total Registers	CG Stage	# of Clock Gates	# of Gated Cells
clka	10	1	2	6
		2	1	5
		3	1	2
clkb	10	1	2	6
		2	1	5
		3	1	2

Clock Gating Structure Details

Clock	CG Stage	Gating Element	Fanout	Latency	Gated Cells
clka	1	U1/clk_gate_y_reg_1	3	2.500	U1/y_reg[2] U1/y_reg[1] U1/y_reg[0]
		U1/clk_gate_y_reg	3	2.500	U1/y_reg[5] U1/y_reg[4] U1/y_reg[3]
	2	U1/clk_gate_y_reg_0	5	0.700	U1/y_reg[8] U1/y_reg[9] U1/y_reg[7] U1/y_reg[6] U1/clk_gate_y_reg
	3	U1/clk_gate_m1	2	0.450	U1/clk_gate_y_reg_1 U1/clk_gate_y_reg_0
clkb	1	U2/clk_gate_y_reg_1	3	2.500	U2/y_reg[2] U2/y_reg[1] U2/y_reg[0]
		U2/clk_gate_y_reg	3	2.500	U2/y_reg[5] U2/y_reg[4] U2/y_reg[3]
	2	U2/clk_gate_y_reg_0	5	0.700	U2/y_reg[9] U2/y_reg[8] U2/y_reg[7] U2/y_reg[6] U2/clk_gate_y_reg
	3	U2/clk_gate_m1	2	0.450	U2/clk_gate_y_reg_0 U2/clk_gate_y_reg_1

Clock Gating Summary

Number of Clock gating elements	8
Number of Gated registers	20 (100.00%)
Number of Ungated registers	0 (0.00%)
Maximum number of stages	3
Total number of registers	20

SEE ALSO

report_clock_gating_check

Prints a report of the clock gating checks.

SYNTAX

```
string report_clock_gating_check  
[-nosplit]  
[-significant_digits digits]  
[instance_list]
```

Data Types

<i>digits</i>	integer
<i>instance_list</i>	list

ARGUMENTS

-nosplit

Prevents line splitting and facilitates writing software to extract information from the report output. By default, most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line in the same column.

-significant_digits *digits*

Specifies the number of digits to the right of the decimal point that are reported. Allowed values are from 0 through 13; the default is 2. Using this option overrides the value set by the variable **report_default_significant_digits**.

instance_list

Specifies designs (all gated clock checks in the specified design), clocks, cells, or pins for which to generate a clock gating report. By default, the report is generated for the current design.

DESCRIPTION

The **report_clock_gating_check** command reports the clock gating checks and the setup hold values. Information displayed about the specified objects include: instance of the cell, enable pin, clock pin, setup/hold time for rise, setup/hold time for fall, user-specified high/low active waveform and clock gating check attributes. The attributes shows from where the clock gating check was original defined:

i	Automatically inferred by Design Compiler
p	Power Compiler inserted the check
l	Defined by the library cell

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example reports user-defined clock gating checks.

```
prompt> set_clock_gating_check -setup 0.27 -hold 0.38
prompt> report_clock_gating_check
```

```
*****
Report : clock gating check
Design : top_design
Version: X-2005.09
Date   : Fri Nov 11 14:47:08 2005
*****
```

Cell	Enable	Clock	Rise		Fall		Low/High	Attr
			Setup	Hold	Setup	Hold		
g1/U1	A	B	0.27	0.38	0.27	0.38	High	i
g1/U2	A	B	0.27	0.38	0.27	0.38	Low	i
g1/U3	B	A	0.27	0.38	0.27	0.38	High	i
g1/U4	A	B	0.27	0.38	0.27	0.38	Low	i

```
Disabled:
None
```

```
Attr:
i:auto inferred, p:power compiler inserted, l:library cell defined
```

In the following example, the first command sets clock gating checks on the design 'test'. The second command sets a clock gating check on cell 'g1/U2' with a non-controlling interval of 'high'. The third command disables the clock gating check on pin 'g1/U3/B'. The fourth command generates a clock gating check report.

The (*) indicates user override of the non-controlling interval for cell 'g1/U2', which is different from what was determined internally. There are no Power Compiler clock gating checks in this design. There is one clock gating check that is disabled on pin 'g1/U3/B'.

```
prompt> set_clock_gating_check -setup 0.33 -hold 0.42 test
prompt> set_clock_gating_check -high -setup 0.21 -hold 0.29 g1/U2
prompt> set_disable_clock_gating_check g1/U3/B
prompt> report_clock_gating_check
```

```

*****
Report : clock gating check
Design : test
Version: X-2005.09
Date   : Fri Nov 11 14:52:18 2005
*****

```

Cell	Enable	Clock	Rise		Fall		Low/High	Attr
			Setup	Hold	Setup	Hold		

g1/U1	A	B	0.33	0.42	0.33	0.42	High	i
g1/U2	A	B	0.21	0.29	0.21	0.29	High (*)	i
g1/U4	A	B	0.33	0.42	0.33	0.42	Low	i
Disabled:								
g1/U3	B	A	0.33	0.42	0.33	0.42	High	
Attr:								
i:auto inferred, p:power compiler inserted, l:library cell defined								

SEE ALSO

```

set_clock_gating_check(2)
set_disable_clock_gating_check(2)
remove_clock_gating_check(2)
remove_disable_clock_gating_check(2)
report_default_significant_digits(3)

```

report_clock_timing

Reports the timing attributes of clock networks.

SYNTAX

```
string report_clock_timing  
-type report_type  
[-clock clock_list]  
[-from_clock from_clock_list]  
[-to_clock to_clock_list]  
[-to to_list]  
[-from from_list]  
[-setup] | [-hold]  
[-launch] | [-capture]  
[-rise] | [-fall]  
[-min] | [-max]  
[-nworst worst_entries]  
[-greater_than lower_limit]  
[-lesser_than upper_limit]  
[-slack_lesser_than slack_upper_limit]  
[-include_uncertainty_in_skew]  
[-verbose]  
[-show_clocks]  
[-nosplit]  
[-significant_digits digits]  
[-nets]  
[-capacitance]  
[-attributes]  
[-physical]  
[-max]  
[-fall]  
[-capture]  
[-hold]
```

Data Types

<i>report_type</i>	string
<i>clock_list</i>	list
<i>from_clock_list</i>	list
<i>to_clock_list</i>	list
<i>to_list</i>	list
<i>from_list</i>	list
<i>worst_entries</i>	integer
<i>lower_limit</i>	float
<i>upper_limit</i>	float
<i>slack_upper_limit</i>	float
<i>digits</i>	integer

ARGUMENTS

-type *report_type*
Specifies the type of report to be generated. Allowed values are as follows:

report_clock_timing
1696

- **transition** - specifies a transition time report.
- **latency** - specifies a latency report.
- **skew** - specifies a skew report. You cannot use the **-launch**, **-capture**, **-rise**, **-fall**, **-min**, **-max**, and **-lesser_than** options if you specify a skew report. You can use the **-include_uncertainty_in_skew** option only in a skew, interclock_skew, or summary report.
- **interclock_skew** - specifies an interclock skew report. You cannot use the **-launch**, **-capture**, **-rise**, **-fall**, **-min**, **-max**, and **-lesser_than** options if you specify an interclock skew report. You can use the **-include_uncertainty_in_skew** option only in a skew, interclock_skew, or summary report.
- **summary** - specifies a summary report that shows the worst instances of transition time, latency, and skew over the clock networks or subnetworks of interest. You can use only the **-clock**, **-to_list**, **-from_list**, **-include_uncertainty_in_skew**, **-nosplit**, and **-significant_digits** options if you specify a summary report.
For non-summary reports, report entries are ordered with respect to the specified attribute of interest (transition time, latency, or skew). All skews reported are "local" skews. For an explanation of local skew, see the DESCRIPTION section.

-clock *clock_list*

Specifies a list of clock networks to be used in the report. A subreport is produced for every clock in *clock_list*, unless you additionally specify a *to_list* or a *from_list* that has no network intersection with a given clock. In that case, the tool drops these clocks from the *clock_list* and issues a warning. By default, if you do not specify *clock_list*, all clocks in the design that have associated clock networks are used in the report.

-from_clock *from_clock_list*

Specifies a list of clock networks to be used as from-clocks in the current interclock skew report. This option can be used only in an interclock skew report. The report considers every clock in the *from_clock_list*, unless you additionally specify a *from_list* that has no network intersection with a given clock. In that case, the tool drops these clocks from the *from_clock_list* and issues a warning. By default, if you do not specify a *from_clock_list*, all clocks in the design that have associated clock networks are used as from-clocks in the report.

-to_clock *to_clock_list*

Specifies a list of clock networks to be used as to-clocks in the current interclock skew report. This option can be used only in an interclock skew report. The report considers every clock in the *to_clock_list*, unless you additionally specify a *to_list* that has no network intersection with a given clock. In that case, the tool drops these clocks from the *to_clock_list* and issues a warning. By default, if you do not specify a *to_clock_list*, all clocks in the design that have associated clock networks are used as to-clocks in the report.

-to *to_list*

Specifies the list of sequential clock network pins or ports to consider as to-pins in the current report. If any named pin is not the clock pin of a sequential device (such as a sink pin), the tool replaces that pin with all sequential clock pins in the transitive fanout of the named pin. If there are no sequential clock pins in the pin's transitive fanout, the pin is dropped from the list with a warning message.

For skew reports, the from-to skew sense is defined by the pins in the *from_list* and the *to_list*, respectively. If the *to_list* is not specified, by default all sink pins in the given clock networks are used. Thus, specifying *to_list* reduces the topological scope of the report. For transition time and latency reports, *from_list* and *to_list* are concatenated and treated as a single list. If neither list is specified, *to_list* is assumed to be populated with all sink pins in the given clock networks.

-from *from_list*

Specifies a list of sequential clock network pins or ports to consider as from-pins in the current report. If a named pin is not the clock pin of a sequential device, the tool replaces that pin with all sequential clock pins in the transitive fanout of the named pin. If there are no sequential clock pins in the pin's transitive fanout, the pin is dropped from the list with a warning message.

For skew reports, the from-to skew sense is defined by the pins in the *from_list* and the *to_list*, respectively. If the *from_list* is not specified, by default all sink pins in the given clock networks are used. Specifying the *to_list* reduces the topological scope of the report. For transition time and latency reports, the *from_list* and *to_list* are concatenated and treated as a single list. If neither is specified, the *to_list* is assumed to be populated with all sink pins in the given clock networks.

-setup

Indicates that only the setup data path is to be used in the report, and that the skews, latencies, or transition times reported must correspond to those used by the **report_timing** command in the verification of setup constraints. The **-setup** and **-hold** options are mutually exclusive. If neither option is specified, **-setup** is assumed. This option cannot be used in summary reports.

-launch

Indicates that only pins launching data are to be used in the report, and that latencies or transition times reported must correspond to those used by the **report_timing** command for sequential device clock pins that are launching data. In skew reports, the role is implicit from the from-to sense indicated by the *from_list* and the *to_list*. In all other reports, the **-launch** and **-capture** options are mutually exclusive. If neither option is specified, **-launch** is assumed. This option cannot be used in summary or skew reports.

-rise

Indicates that the active transition is a rising edge for sequential device clock pins in the current report. The **-rise** and **-fall** options are mutually exclusive. If neither option is specified, the active transition at a latch or flip-flop is deduced from the launch or capture role and the behavior of the sequential device itself. This option enables you to answer "what if" questions regarding latency and transition time at sequential device clock pins. This option cannot be used in summary or skew reports.

-min

Specifies that the reports are sorted by minimum latencies or transition times. The **-min** and **-max** options are mutually exclusive. If neither option is specified, the reports are sorted in the same manner as described for the **-nworst** option. This option cannot be used in summary or skew reports, or if the **-launch**, **-capture**, **-setup**, or **-hold** options are used. To check the quality of a balanced clock tree network, you can generate two latency reports, one with **-min** and one with **-max**.

-nworst *worst_entries*

Specifies the number of worst report entries to be reported per clock domain. The default value is 1. Entries are sorted with respect to the attribute of interest (transition time, latency, or skew) specified with **-type** *report_type*.

The worst entries are those most likely to cause a violation. For example, if you request a latency report using **-setup** and **-capture**, the smallest *worst_entries* are listed, sorted in ascending order, because small capture latencies for setup paths are more likely to lead to a violation than large capture latencies. For skew reports, the worst entries always correspond to the largest skews over the specified domain. The above definition of "worst" can be overridden by use of the **-min** or **-max** options. This option cannot be used in summary reports.

-greater_than *lower_limit*

Indicates that only those entries whose attribute value (latency, transition time, or skew) is greater (more positive) than *lower_limit* are shown. This option cannot be used in summary reports.

-lesser_than *upper_limit*

Indicates that only those entries whose attribute value (latency, transition time, or skew) is less (more negative) than *upper_limit* are shown. This option cannot be used in summary or skew reports.

-slack_lesser_than *slack_upper_limit*

Indicates that only those entries whose slack value is less (more negative) than *slack_upper_limit* are shown. For skew report entries, slack is the worst slack over all paths launched by the from-pin and captured by the to-pin. For transition time or latency report entries, slack is defined as the worst slack of all paths either launched by a transition at the sink pin (if **-launch** is used) or captured by a transition at the sink pin (if **-capture** is used). Using this filter can greatly increase the runtime of this report. However, when used with **-capture** the effect on runtime should be small, since the tool is able to make use of cached slack values to avoid expensive recomputation. This option cannot be used in summary reports.

-include_uncertainty_in_skew

Indicates that the user-defined uncertainty between the sequential devices in a launch/capture pair is to be considered a component of skew. This option can be used only in skew or summary reports.

-verbose

Indicates that a more detailed report is to be generated. Instead of a single line per pin, verbose reports trace the entire source-to-sink path through a clock network to show how the final reported attribute (skew, latency, or transition time) was accumulated over the course of the path. This option

cannot be used in summary reports.

-show_clocks

Indicates that the launching and capturing clocks are shown for every interclock skew entry in the report. This is useful if the *from_clock_list* or the *to_clock_list* contains more than one clock each. This option can only be used in interclock skew reports.

-nosplit

Prevents line-splitting and facilitates writing software to extract information from the report output. By default, most of the design information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

-significant_digits *digits*

Specifies the number of digits after the decimal point to be displayed for time values in the generated report. Allowed values are 0-13. The default value is determined by the **report_default_significant_digits** variable, whose default value is 2. Use this option if you want to override the default. This option controls only the number of digits displayed, not the precision used internally for analysis. For analysis, the tool uses the full precision of the platform's fixed-precision, floating-point arithmetic capability.

-nets

Shows nets in verbose path traces. The default is not to show nets. To show the delay for the nets, use the **-input_pins** option. This option is similar to its counterpart in the **report_timing** command.

-capacitance

Indicates that total (lump) capacitance be shown in verbose path traces. The default is not to show capacitance. For each driver pin, the total capacitance driven by the driver is displayed in a column preceding both incremental path delay and transition time. When **-nets** is specified, the capacitance is printed on the lines with nets instead of the lines with driver pins. This option is similar to its counterpart in the **report_timing** command.

-attributes

Shows in verbose path traces the attributes specified in the **timing_report_attributes** variable. The current set of attributes supported are **dont_touch**, **dont_use**, **map_only**, and **size_only** for cells, and **dont_touch** and **ideal_net** for nets. This option is similar to its counterpart in the **report_timing** command.

-physical

Shows the locations of the pins and the capacitive loads for the pins and nets in verbose path traces. The loads are displayed as a pair of values of which the first is the wire capacitance of the net and the second value is the total capacitance driven by the driver. If the pin location cannot be determined, the cell location is displayed, with the coordinates in microns. This option is similar to its counterpart in the **report_timing** command.

-max

Specifies that the reports are sorted by maximum latencies or transition times. The **-min** and **-max** options are mutually exclusive. If neither option

is specified, the reports are sorted in the same manner as described for the **-nworst** option. This option cannot be used in summary or skew reports, or if the **-launch**, **-capture**, **-setup**, or **-hold** options are used. To check the quality of a balanced clock tree network, you can generate two latency reports, one with **-min** and one with **-max**.

-fall

Indicates that the active transition is a falling edge for sequential device clock pins in the current report. The **-rise** and **-fall** options are mutually exclusive. If neither option is specified, the active transition at a latch or flip-flop is deduced from the launch or capture role and the behavior of the sequential device itself. This option enables you to answer "what if" questions regarding latency and transition time at sequential device clock pins. This option cannot be used in summary or skew reports.

-capture

Indicates that only pins capturing data are to be used in the report, and that the latencies or transition times reported must correspond to those used by the **report_timing** command for sequential device clock pins that are capturing data. In skew reports, the role is implicit from the *from_list* and the *to_list*. In all other reports, the **-launch** and **-capture** options are mutually exclusive. If neither option is specified, **-launch** is assumed. This option cannot be used in summary or skew reports.

-hold

Indicates that only the hold data path is to be used in the report, and that the skews, latencies, or transition times reported must correspond to those used by the **report_timing** command in the verification of hold constraints. The **-setup** and **-hold** options are mutually exclusive. If neither option is specified, **-setup** is assumed. This option cannot be used in summary reports.

DESCRIPTION

This command generates a report of clock timing information for the current design.

Several reporting types allow you to examine skew, latency, and transition time attributes of a specified clock network or subnetwork at various levels of generality. By default, the report displays the values of these attributes only at sink pins (that is, the clock pins of sequential devices) of the clock network. Use the **-verbose** option to display source-to-sink path traces. If you specify several clock domains, **report_clock_timing** generates a separate subreport for each.

At the highest level of abstraction is the summary style report, which provides only a list of maxima and minima of the skew, latency, and transition time attributes over the given networks. At a lower level of abstraction are the transition, latency, and skew type reports, called list style reports, in which you can sort, filter, and display the worst set of sink pins in the given network with respect to a single attribute of interest. For skew reports, each report entry is a pair of sink pins and their relative skew. For transition time or latency reports, each entry corresponds to a single sink pin. The lowest level of abstraction is provided by verbose mode, which replaces every sink pin in a list style report by a corresponding source-to-sink path trace.

In both summary and list style reports, the right column is an attributed column.

Corresponding to each sink pin, the set of character symbols in this column indicates the following:

Symbol	Meaning
-----	-----
w	Worst-case operating condition
b	Best-case operating condition
r	Rising transition
f	Falling transition
p	Propagated clock to this pin
i	Clock inversion to this pin
e	Exception on this pin
-	Launching transition
+	Capturing transition

In verbose mode, back-annotations on path elements in the timing path are indicated using a character symbol. For definitions of these character symbols, see the man page for the **report_timing** command.

Skews reported by **report_clock_timing** are local skews only. Local skew exists from one sink pin to another only as long as their associated sequential devices are connected via a data path in the appropriate from-to sense. Note that even if all data paths between the sequential devices are false because of user-defined exceptions, the skew still qualifies as local skew if the devices are connected topologically.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example shows a typical summary style report:

```
prompt> report_clock_timing -type summary
```

```
*****
Report : clock timing
        -type summary
        -nworst 1
Design : xor_testcase
Version: W-2004.12-GSHELL-LCA1
Date   : Tue Nov 20 06:17:33 2004
*****
```

```
Clock: CK1
```

```
-----
Maximum setup launch latency:
      flop9/CP                                3.08      rp++
```

```
Minimum setup capture latency:
```

or2_3/B	1.15	fpi-+
Minimum hold launch latency: or2_3/B	1.15	fpi-+
Maximum hold capture latency: flop9/CP	3.08	rp-+
Maximum active transition: or2_3/B	0.20	fpi-+
Minimum active transition: or2_3/B	0.09	fpi-+
Maximum setup skew: flop9/CP		rp-+
flop10/CP	0.87	rp-+
Maximum hold skew: flop9/CP		rp-+
flop10/CP	-0.21	rp-+

The following example displays the five worst setup skews in the clock network of CLK1, taking uncertainty into account:

```
prompt> report_clock_timing -clock CLK1 -type skew -setup \
        -nworst 5 -include_uncertainty_in_skew
```

```
*****
Report : clock timing
        -type skew
        -nworst 5
        -setup
        -include_uncertainty_in_skew
Design : multi_domain
Version: W-2004.12-GSHELL-LCA1
Date   : Tue Nov 20 06:49:19 2004
*****
```

Clock: CLK1

Clock Pin	Latency	Uncert	Skew	
f2_2/CP	6.11			rp-+
f2_1/CP	2.01	0.11	4.21	fp-+
13_2/G	4.10			rpi-
f1_2/CP	1.00	0.22	3.32	rpi-+
12_2/G	4.11			rp-
f1_2/CP	1.00	0.12	3.23	rpi-+
f2_2/CP	6.11			rp-+

l3_3/G	3.01	0.11	3.21	rp-
l1_3/G	5.11			rp-
f2_1/CP	2.01	0.11	3.21	fp-+

The following example displays the five worst launching latencies for hold paths in the clock network of CLK2:

```
prompt> report_clock_timing -clock CLK2 -hold -launch -nworst 5 \
        -type latency
```

```
*****
```

```
Report : clock timing
        -type latency
        -launch
        -nworst 5
        -hold
```

```
Design : multi_domain
```

```
Version: W-2004.12-GSHELL-LCA1
```

```
Date   : Tue Nov 20 06:55:56 2004
```

```
*****
```

```
Clock: CLK2
```

Clock Pin	Trans	Source	--- Latency ---		
			Network	Total	
f1_2/CP	0.04	0.00	1.00	1.00	rpi-+
f1_3/CP	0.00	0.01	1.00	1.01	fp-+
f2_1/CP	0.01	0.01	2.00	2.01	rp-+
f1_1/CP	0.00	0.00	3.00	3.00	rpi-+
f3_2/CP	0.00	0.00	3.00	3.00	fpi-+

The following example demonstrates how to request a verbose report showing the worst local skew from f2_2/CP to any other sink pin:

```
prompt> report_clock_timing -type skew -verbose -from f2_2/CP
```

```
*****
```

```
Report : clock timing
        -type skew
        -verbose
        -nworst 1
        -setup
```

```
Design : multi_domain
```

```
Version: W-2004.12-GSHELL-LCA1
```

```
Date   : Tue Nov 20 07:00:56 2004
```

```
*****
```

```
Clock: CLK1
```

Startpoint: f2_2 (rising edge-triggered flip-flop clocked by CLK1)
 Endpoint: f2_1 (rising edge-triggered flip-flop clocked by CLK1)

Point	Trans	Incr	Path

clock source latency		0.11	0.11
clk2 (in)	0.00	0.00	0.11 r
az_1/Z (B1I)	0.09	1.00 H	1.11 r
az_2/Z (B1I)	0.13	1.00 H	2.11 r
bf2_2_1/Z (B1I)	0.02	1.00 H	3.11 r
if2_2_1/Z (IVA)	0.44	1.00 H	4.11 f
bf2_2_2/Z (B1I)	0.01	1.00 H	5.11 f
if2_2_2/Z (IVA)	0.13	1.00 H	6.11 r
f2_2/CP (FD1)	0.13	0.00	6.11 r
startpoint clock latency			6.11
clock source latency		0.01	0.01
clk2 (in)	0.00	0.00	0.01 r
az_1/Z (B1I)	0.02	1.00 H	1.01 r
az_3/Z (B1I)	0.01	1.00 H	2.01 r
f2_1/CP (FD1)	0.01	0.00	2.01 r
endpoint clock latency			2.01

startpoint clock latency			6.11
endpoint clock latency			-2.01

skew			4.10

The following example traces the two worst launch latencies for hold paths in the clock network of CLK1:

```
prompt> report_clock_timing -type latency -hold -verbose \
        -nworst 2 -clock CLK1
```

```
Report : clock timing
        -type latency
        -verbose
        -launch
        -nworst 2
        -hold
```

```
Design : multi_domain
Version: W-2004.12-GSHELL-LCA1
Date   : Tue Nov 20 07:14:28 2004
```

Clock: CLK1

Endpoint: f1_2 (rising edge-triggered flip-flop clocked by CLK1')

Point	Trans	Incr	Path

clock source latency		0.00	0.00
clk1 (in)	0.00	0.00	0.00 f

if1_2_1/Z (IVA)	0.04	1.00 H	1.00 r
f1_2/CP (FD1)	0.04	0.00	1.00 r
total clock latency			1.00

Endpoint: f1_3 (rising edge-triggered flip-flop clocked by CLK1)

Point	Trans	Incr	Path
clock source latency		0.01	0.01
clk1 (in)	0.00	0.00	0.01 r
bf1_3_1/Z (B1I)	0.00	1.00 H	1.01 r
f1_3/CP (FD1)	0.00	0.00	1.01 r
total clock latency			1.01

The following example makes use of a slack filter to display the worst three skews between latches whose latch-to-latch paths are violating:

```
prompt> report_clock_timing -type skew -nworst 3 \
        -slack_lesser_than 0.0
```

```
Report : clock timing
        -type skew
        -slack_lesser_than 0.00
        -nworst 3
        -setup
```

```
Design : multi_domain
Version: W-2004.12-GSHELL-LCA1
Date   : Fri Dec 14 09:11:05 2004
```

Clock: CLKA

Clock Pin	Latency	CRP	Skew	Slack	
f2_2/CP	6.01				rp-+
f2_1/CP	2.11	-0.03	3.87	-1.49	rp-+
l2_2/G	4.01				rp-
f1_2/CP	1.10	-0.21	2.70	-6.64	rpi-+
l1_3/G	5.01				rp-
f2_1/CP	2.11	-0.32	2.58	-1.63	rp-+

The following example requests the two largest setup skews for paths both launched and captured by any clock networks in the list `$my_clocks`:

```
prompt> report_clock_timing -type interclock_skew \
        -from_clock $my_clocks -to_clock $my_clocks \
        -nworst 2 -include_uncertainty_in_skew -show_clocks
```

```

Report : clock timing
        -type interclock_skew
        -nworst 100
        -setup
        -include_uncertainty_in_skew
        -show_clocks
Design : my_design
Version: W-2004.12-GSHELL-LCA1
Date   : Thu May  2 10:55:20 2004
*****

```

```

Number of startpoint pins   : 907
Number of endpoint pins     : 907
Number of startpoint clocks : 5
Number of endpoint clocks   : 5

```

Clock Pin	Latency	Uncert	Skew	
orig_clk				
orig_if/ram_pdp1/FFB35/CK	1016.72			rp-+
dcd_ram/ram_clk				
dcd_ram/dcd_ram/RAM/CKRB	149.60	195.00	1062.12	rp-+
comp_clk				
comp_if/c_port_if/edge_trig_reg/CK	1400.42			rp-+
comp_clk				
comp_if/c_port_if/posi/iq_reg/CK	1159.09	199.00	440.34	rp-+

SEE ALSO

```

report_clock(2)
report_timing(2)
set_clock_uncertainty(2)
report_default_significant_digits(3)
timing_remove_clock_reconvergence_pessimism(3)

```

report_clock_tree

Reports structural and timing characteristics of a compiled clock tree.

SYNTAX

```
status report_clock_tree
[-clock_trees clock_tree_list]
[-summary]
[-structure]
[-drc_violators]
[-settings]
[-exceptions [-show_all_sinks]]
[-from from_list | -to to_list]
[-operating_condition condition]
[-level_info]
[-high_fanout_net net_or_pin_list
  | -premesh
  | -postmesh]
[-nosplit]
[-all_drc_violators]
[-partial_structure_within_exceptions]
[-skew_group skew_groups_string]
[-histogram_transition transition_histogram_file_name
  | -histogram_capacitance capacitance_histogram_file_name
  | -histogram_rcdelay rc_delay_histogram_file_name
  | -histogram_fanout fanout_histogram_file_name]
[-number_of_bins bin_number]]
```

Data Types

<i>clock_tree_list</i>	list
<i>from_list</i>	list
<i>to_list</i>	list
<i>condition</i>	string
<i>net_or_pin_list</i>	list
<i>skew_groups_string</i>	string
<i>transition_histogram_file_name</i>	string
<i>capacitance_histogram_file_name</i>	string
<i>rc_delay_histogram_file_name</i>	string
<i>fanout_histogram_file_name</i>	string
<i>bin_number</i>	integer

ARGUMENTS

-clock_trees *clock_tree_list*
Reports only the clock trees specified in *clock_tree_list*. By default, the command applies to all currently defined clock trees.

-summary
Prints a summary table for the clocks.

-structure
 Prints the complete structure of the clock tree. This report will traverse through exceptions such as explicit float pins and implicit and explicit exclude pins.

-drc_violators
 Lists the design rule checking (DRC) violations that occur in the clock trees up to the endpoints (default sinks or don't touch subtrees), including violations beyond exception on stop pins, exclude pins, and float pins but excluding violations on don't buffer nets, don't touch subtrees, nets inside interface logic models, nets connected to boundary cells or pad cells.

-settings
 Prints the clock tree settings specified by the **set_clock_tree_options** command, and references available for buffering, design rule checking (DRC) constraints, the list of available routing layers, and so on.

-exceptions
 Prints a detailed list of clock tree exceptions.

-show_all_sinks
 Reports all default sinks in the exception report when used with the **-exceptions** option.

-from from_list
 Specifies the list of pin names whose fanout clock tree is reported.

-to to_list
 Specifies the list of pin names to which the clock paths are reported. The specified pins must be clock tree sink pins. This option can be used with the **-operating_condition** and **-clock_trees** options.

-operating_condition condition
 Reports the clock tree based on the specified condition.

-level_info
 Reports information about the clock tree by level.

-high_fanout_net net_or_pin_list
 Reports a tree built by the clock tree synthesis based high fanout net synthesis engine. This option can be used alone or in combination with the following options:

- summary**
- structure**
- level_info**
- drc_violators**
- operating_condition**
- nosplit**

-premesh
 Reports the clock network that has a clock mesh, starting from the clock root to the mesh driver inputs. This option treats the mesh driver inputs as stop pins and ignores the portion of the clock tree beneath the mesh. You can run this option as a stand-alone process or in combination with the following

options:

- clock_tree
- summary
- structure
- exceptions
- drc_violators
- operating_condition
- nosplit

-postmesh

Reports the clock network that has a clock mesh, starting from the inputs of the clock drivers that are the load pins of the clock mesh. This option does not report the clock network from the clock root to the clock mesh. You can run this option as a stand-alone process or in combination with the following options:

- clock_tree
- summary
- structure
- exceptions
- drc_violators
- operating_condition
- nosplit

When you specify this option by itself, the command reports the longest path, the shortest path, and the skew among all subtrees, starting from the load pins of the clock mesh. In addition, the clock mesh is treated as ideal so that every point of the clock mesh would have the same arrival time. Using this option with other options is equivalent to using the -from 'load pins of the clock mesh'.

-nosplit

Prevents linesplitting to allow ease of use to create scripts to extract information from the report output. By default, the design information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

-all_drc_violators

Forces the **report_clock_tree** command to list all design rule checking (DRC) violations in the clock trees, including violations beyond exceptions and violations on don't touch subtrees but excluding don't buffer nets, nets inside interface logic models (ILM), nets connected to boundary cells or pad cells.

-partial_structure_within_exceptions

Prints the part of the clock tree structure that is within explicit exclude and float pin exceptions. The clock structure beyond explicit exclude and float pin exceptions is not printed. Use the **-structure** option to print the complete clock structure.

-skew_group *skew_groups_string*

Reports the clock skew information only for the specified skew groups. You

report_clock_tree

can separate each skew group with a space in the specified *skew_groups_string*. The default skew group name is **default**. For each specified skew group, the skew information will be printed for all clocks that propagate to the skew group. You can only use this option with **-clock_trees** and **-summary**. If you did not specify **-skew_group** but ran **commit_skew_group** successfully, the clock skew information of all skew groups will be reported.

- histogram_transition *transition_histogram_file_name*
Reports the clock pin transition delay information to the file specified by *transition_histogram_file_name* in histogram format.
- histogram_capacitance *capacitance_histogram_file_name*
Reports the clock pin capacitance information to the file specified by *capacitance_histogram_file_name* in histogram format.
- histogram_rcdelay *rc_delay_histogram_file_name*
Reports the RC delays between clock drivers and load pairs to the file specified by *rc_delay_histogram_file_name* in histogram format. RC-delay histogram must be reported after clock tree synthesis gets valid delay data on clock nets.
- histogram_fanout *fanout_histogram_file_name*
Reports clock pin fanout numbers to the file specified by *fanout_histogram_file_name* in histogram format.
- number_of_bins *bin_number*
Specifies the number of bins that are used to generate the histogram report. Use this option when you specify at least one of the histogram reporting options: **-histogram_transition**, **-histogram_capacitance**, **-histogram_rcdelay**, **-histogram_fanout**. By default, the *nubmer_of_bins* is 10.

DESCRIPTION

The **report_clock_tree** command obtains information on clock trees. The command reports the structural and timing characteristics of a compiled clock tree.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

In the following example, the **report_clock_tree** command is run on a previously-compiled clock tree called CLK1, and the reported results are based on a clock route estimator:

```
prompt> report_clock_tree -clock_trees CLK1
```

In the following example, **report_clock_tree** is run on a precompiled clock tree named CLK1. The report includes a listing of all logical design rule checking (DRC)

violations and a report of clock tree settings.

```
prompt> report_clock_tree -clock_trees CLK1 -drc_violators -settings
```

In the following example, **report_clock_tree** is run on a precompiled clock tree named CLK1. The report includes hierarchical level information about the clock tree netlist structure, the distribution clock tree delays, DRC violations, and all exceptions.

```
prompt> report_clock_tree -clock_trees CLK1 -structure -drc_violators -exceptions
```

In the following example, **report_clock_tree** is run on a precompiled clock tree named clk. The report includes a listing of all clock tree exceptions in the clock tree.

```
prompt> report_clock_tree -exceptions
```

```
*****
```

```
Report : clock tree
```

```
Design : top
```

```
Version: Y-2006.06
```

```
Date   : Fri Sep 15 17:58:35 2006
```

```
*****
```

```
===== Clock Tree Exception Pins =====
```

```
Clock Tree Exceptions Summary
```

```
=====
```

```
1. Clock: CLK
```

```
   Clock root: clk
```

```
   Clock net: clk
```

```
   Total sinks: 2
```

```
   Explicit ignore pins: 0
```

```
   Explicit sync pins: 0
```

```
   Implicit ignore pins: 1
```

```
   Default sink pins: 1
```

```
   Explicit nonstop pins: 0
```

```
   Implicit nonstop pins: 0
```

```
   Dont touch subtree pins: 0
```

```
   Dont buffer nets: 0
```

```
   Dont size cells: 0
```

```
=====
```

```
Clock Tree Exceptions
```

```
=====
```

```
Legends Used
```

```
-----
```

```
(P) = Default sink pin
```

```
(F) = Explicit stop/float pin
```

(I) = Implicit ignore pin
(E) = Explicit ignore pin
(J) = Implicit nonstop pin
(T) = Explicit nonstop pin
(D) = Dont touch subtree pin
(B) = Dont buffer net
(S) = Dont size cell

1. Clock: CLK
Clock root: clk
Clock net: clk
Total sinks: 2
Explicit ignore pins: 0
Explicit sync pins: 0
Implicit ignore pins: 1
 (I) s/clk
Default sink pins: 1
Explicit nonstop pins: 0
Implicit nonstop pins: 0
Dont touch subtree pins: 0
Dont buffer nets: 0
Dont size cells: 0

=====

In the following example, the **report_clock_tree** command is run on a previously compiled clock tree called CLK1, and two histogram report files are generated. The histogram report of transition delay of all clock pins in CLK1 is written to file tran_hist.rpt. The histogram report of RC-delay between all the driver-load pin pairs in CLK1 is written to file rc_hist.rpt. The histogram reports are generated, using the default bin value of 10.

```
prompt> report_clock_tree -clock_trees CLK1 \
-histogram_transition tran_hist.rpt -histogram_rcdelay rc_hist.rpt
```

SEE ALSO

compile_clock_tree(2)
set_clock_tree_exceptions(2)
set_clock_tree_options(2)

report_clock_tree_optimization_options

Reports options used by clock tree optimization.

SYNTAX

```
status report_clock_tree_optimization_options  
[-clock_trees clock_name_or_collection]
```

ARGUMENTS

```
-clock_trees clock_name_or_collection  
    Specifies a set of clocks to be processed. Only clock names defined through  
    create_clock or create_generated_clock are accepted. If there is a list of  
    clock names, the options then apply to those clocks only. By default, the  
    options apply for all the clocks.
```

DESCRIPTION

This command is the UI reporting options for clock tree optimization in ICC. Before command optimize_clock_tree or clock_opt, user can use this command to report options to specific clocks if -clock_trees is given, or report options to all clocks that do not have the per clock option.

The options reported by this command are enable_multicorner_optimization, gate_sizing, gate_relocation, preserve_levels, area_recovery.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example disables gate_sizing to clock PCI_CLK. Then report the options for PCI_CLK.

```
prompt> set_clock_tree_optimization_options -clock_trees [get_clock PCI_CLK]\  
-gate_sizing false  
1  
prompt> report_clock_optimization_options -clock_trees PCI_CLK  
====CLOCK: PCI_CLK====  
Enable multicorner optimization: none  
Gate sizing: false  
Gate relocation: true  
Preserve level: false  
Area recovery: true  
  
1
```


SEE ALSO

`optimize_clock_tree(2)`
`clock_opt(2)`
`compile_clock_tree(2)`
`set_clock_tree_options(2)`
`set_clock_tree_optimization_options(2)`
`reset_clock_tree_optimization_options(2)`

report_clock_tree_power

Calculates and reports dynamic and static power for a clock network.

SYNTAX

```
int report_clock_tree_power  
[-net]  
[-cell]  
[-verbose]  
[-include_input_nets]  
[-include_register]  
[-sort_mode mode]  
[-nosplit]  
[clock_list]
```

Data Types

<i>mode</i>	string
<i>clock_list</i>	object_list

ARGUMENTS

-net

Reports the power consumption of nets. Use the **-net** option alone, or use it with the **-cell** option to report on both nets and cells. By default, only the design's summary power information is reported when neither option is specified.

-cell

Reports the power consumption of cells. When using the **-cell** option, some entries in the power report might not apply to certain cells. The report column entry for those cells is annotated with N/A. Use the **-cell** option alone, or use it with the **-net** option to report on cells and nets. By default, only the design's summary power information is reported when neither option is specified.

-verbose

Displays additional detailed information about the power of the cells and/or nets. This option is valid only if the **-net** and/or the **-cell** options are specified.

-include_input_nets

Specifies that the power of the boundary net is included in the power report. The default is to exclude the boundary net.

-include_register

Specifies that the power of the registers is included in the power report. The default is to exclude the registers.

-sort_mode *mode*

Determines the sorting mode for the report order and **-nworst** selection. The valid sorting modes for the **-net** and **-cell** options are as follows:

report_clock_tree_power

-net option	-cell option
-----	-----
name	name
cumulative_fanout	cumulative_fanout
cumulative_fanin	cumulative_fanin
net_static_probability	cell_internal_power
net_switching_power	cell_leakage_power
net_toggle_rate	dynamic_power
total_net_load	
When using -net and -cell and specifying a sorting mode, you must select a sorting mode that is valid for both options. The mode is used for both the cell and the net reports.	
If you do not explicitly set the sorting mode, a default is chosen based on the mode of the report_clock_tree_power command:	
Mode	Implicit default
-----	-----
-net	net_switching_power
-cell	cell_internal_power
-net -cell	dynamic_power

-nosplit

Prevents line-splitting and facilitates writing software to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

clock_list

Specifies a list of clock objects. The list can be the name of the clock used in the **create_clock** command, or a list of clock port names. If a *clock_list* is not specified, all clock objects in the current design are considered.

DESCRIPTION

The **report_clock_tree_power** command calculates and reports power for a clock network. The probabilistic estimation algorithm functions on nets that are not explicitly annotated with switching activity values. During the probabilistic propagation, **report_clock_tree_power** uses the startpoint nets' switching activity values, if available, when computing the switching activity values for internal nets. The switching activity values are retained for any nets that were annotated with the **set_switching_activity** command. The values are not overwritten during the probabilistic propagation.

The options enable you to specify cells and/or nets for reporting. The default operation is to display the summary power values for only the current design.

The **-verbose** option displays more detailed power information.

The **-include_input_nets** and **-sort_mode** options enable filtering of objects that are selected by **report_clock_tree_power**. The **-include_input_nets** option also affects the way that the design's power values are computed by including the clock input net in the design's totals. The **-include_register** option affects the way that the design's power values are computed by including the power consumption of the registers. The -

sort_mode option also affects the formatting of the power reports by modifying the order of nets and/or cells that are displayed by **report_clock_tree_power**.

If they are not specifically annotated with switching activity information, all input ports and black-box cell outputs are assumed to have a toggle rate of

$$2 * sp * (1-sp) * fclk$$

where *sp* is the static probability (the default is 0.5), and *fclk* is the frequency of the object's related clock.

Power analysis uses any back-annotated net loads during the power calculation. For nets that do not have back-annotated capacitance, the net load is estimated from the appropriate wire load model. If any cluster information has been annotated on the design (Floorplan Manager), Power Compiler uses the improved capacitance estimates from the cluster's wire loads.

When invoked from within `dc_shell` (Design Compiler), the **report_clock_tree_power** command first checks out a Power Compiler license. If a license is not available, the command terminates with an error message. Otherwise, the command proceeds normally. At the completion of the command, the Power Compiler license is released. To keep the license at the completion of the **report_clock_tree_power** command, set the variable **power_keep_license_after_power_commands** to **true**.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example shows a **report_clock_tree_power** summary report. It reports all clock nets in the design.

```
prompt> report_clock_tree_power
```

Information: Clock is not specified, all clock nets of design are considered

```
*****
Report : clock_tree_power
Design : cts_test
Version: V-2004.06-DFT-Power-Beta4
Date   : Thu Apr 1 11:27:57 2004
*****
```

Library(s) Used:

```
lsi_10k_wc (File: /remote/test/cts/small/db/lsi_10k_wc.db)
lsi_10k_sup_crnt_wc (File: /remote/test/cts/small/db/lsi_10k_sup_crnt_wc.db)
```

```
Operating Conditions: nom_pvt   Library: lsi_10k_wc
Wire Load Model Mode: top
```

Design	Wire Load Model	Library

```
report_clock_tree_power
1718
```

cts_test zero_load lsi_10k_wc

Global Operating Voltage = -1.049e+06

Power-specific unit information :

Voltage Units = 1V

Capacitance Units = 1.000000pf

Time Units = 1ns

Dynamic Power Units = 1mW (derived from V,C,T units)

Leakage Power Units = 1pW

Report Clock Tree Power on clock net clk.

Information: Updating design information... (UID-85)

Information: Propagating switching activity (super low effort zero delay simulation). (PWR-6)

Cell Internal Power = 293.8016 uW (81%)

Net Switching Power = 69.0751 uW (19%)

Total Dynamic Power = 362.8767 uW (100%)

Cell Leakage Power = 452.1823 nW

The following example shows a net power report for the *clk* clock net, sorted by net name and filtered to display only the 5 nets with the highest switching power. A low effort analysis is performed to estimate the design's power values.

prompt> **report_clock_tree_power -net -sort_mode name clk**

Report : clock_tree_power

-net

-sort_mode name

Design : cts_test

Version: V-2004.06-DFT-Power-Beta4

Date : Thu Apr 1 11:33:24 2004

Library(s) Used:

lsi_10k_wc (File: /remote/test/cts/small/db/lsi_10k_wc.db)

lsi_10k_sup_crnt_wc (File: /remote/test/cts/small/db/lsi_10k_sup_crnt_wc.db)

Operating Conditions: nom_pvt Library: lsi_10k_wc

Wire Load Model Mode: top

Design	Wire Load Model	Library
cts_test	zero_load	lsi_10k_wc

Global Operating Voltage = 1.35

Power-specific unit information :

Voltage Units = 1V

Capacitance Units = 1.000000pf

Time Units = 1ns
 Dynamic Power Units = 1mW (derived from V,C,T units)
 Leakage Power Units = 1pW

Report Clock Tree Power on clock net clk.

Attributes

- a - Switching activity information annotated on net
- d - Default switching activity information on net

Net	Total Net Load	Static Prob.	Toggle Rate	Switching Power	Attrs
clk	0.060	0.500	0.6667	0.0362	d
n2	0.036	0.375	0.4994	0.0164	
n4	0.043	0.315	0.4202	0.0165	
Total (3 nets)				69.0751 uW	

The following example displays both cell and net reports for all clock nets of the design:

prompt> **report_clock_tree_power -net -cell -include_register**

Information: Clock is not specified, all clock nets of design are considered

```
*****
Report : clock_tree_power
      -net
      -cell
      -sort_mode dynamic_power
Design : cts_test
Version: V-2004.06-DFT-Power-Beta4
Date   : Thu Apr 1 11:36:14 2004
*****
```

Library(s) Used:

lsi_10k_wc (File: /remote/test/cts/small/db/lsi_10k_wc.db)
 lsi_10k_sup_crnt_wc (File: /remote/test/cts/small/db/lsi_10k_sup_crnt_wc.db)

Operating Conditions: nom_pvt Library: lsi_10k_wc
 Wire Load Model Mode: top

Design	Wire Load Model	Library
cts_test	zero_load	lsi_10k_wc

Global Operating Voltage = 1.35
 Power-specific unit information :
 Voltage Units = 1V
 Capacitance Units = 1.000000pf
 Time Units = 1ns
 Dynamic Power Units = 1mW (derived from V,C,T units)

report_clock_tree_power
 1720

Leakage Power Units = 1pW

Report Clock Tree Power on clock net clk.

Attributes

- a - Switching activity information annotated on net
- d - Default switching activity information on net

Net	Total Net Load	Static Prob.	Toggle Rate	Switching Power	Attrs

n2	0.036	0.375	0.4994	0.0164	
n4	0.043	0.315	0.4202	0.0165	
clk	0.060	0.500	0.6667	0.0362	d

Total (3 nets)				69.0751 uW	

Attributes

- h - Hierarchical cell

Cell	Cell Internal Power	Driven Net Switching Power	Tot Dynamic Power (% Cell/Tot)	Cell Leakage Power	Attrs

clk_gate_x_reg/main_gate	0.0334	0.0164	4.97e-02 (67%)	0.2902	
clk_gate_y_reg/main_gate	0.0290	0.0165	4.55e-02 (64%)	0.2902	
z_reg[0]	0.0224	9.041e-04	2.33e-02 (96%)	37681.8008	
z_reg[3]	0.0223	9.233e-04	2.32e-02 (96%)	37681.8008	
z_reg[2]	0.0224	6.107e-04	2.30e-02 (97%)	37681.8008	
z_reg[1]	0.0224	4.539e-04	2.29e-02 (98%)	37681.8008	
x_reg[2]	0.0203	6.828e-04	2.10e-02 (97%)	37681.8008	
x_reg[3]	0.0204	6.017e-04	2.10e-02 (97%)	37681.8008	
x_reg[1]	0.0201	2.341e-04	2.03e-02 (99%)	37681.8008	
x_reg[0]	0.0195	3.599e-04	1.99e-02 (98%)	37681.8008	
y_reg[2]	0.0156	2.441e-04	1.59e-02 (98%)	37681.8008	
y_reg[1]	0.0154	4.267e-04	1.58e-02 (97%)	37681.8008	
y_reg[0]	0.0155	1.482e-04	1.56e-02 (99%)	37681.8008	
y_reg[3]	0.0154	2.235e-04	1.56e-02 (99%)	37681.8008	

Totals (14 cells)	293.802uW	38.651uW	332.453uW (88%)	452.182nW	

SEE ALSO

```
create_clock(2)
set_switching_activity(2)
power_keep_license_after_power_commands(3)
```

report_congestion

Reports the congestion statistics.

SYNTAX

```
status report_congestion
[-routing_stage global | track | detail]
[-search_repair | -no_reroute]
[-grc_based]
[-grc_number grc_number]
[-overflow_threshold threshold]
[-by_layer]
[-coords | -polygon {polygon_area}]
```

Data Types

<i>grc_number</i>	int
<i>threshold</i>	int
<i>rectangle</i>	list of points
<i>polygon_area</i>	list of points

ARGUMENTS

-routing_stage global | track | detail

Specifies the routing stage for reporting the congestion map.
Valid values are

- **global** (the default)

Reports a global route congestion map.

By default, the tool generates a new global route congestion map by running global routing without the search-and-repair stage. To also run the search-and-repair stage, use the **-search_repair** option.

To report the existing global route congestion map instead of generating a new one, use the **-no_reroute** option.

- **track**

Reports a track assignment congestion map.

By default, the tool generates a new track assignment congestion map from the current track assignment routing information. Before running this command, you need to ensure that the design is track assigned.

To report the existing track assignment congestion map instead of generating a new one, use the **-no_reroute** option.

The **-search_repair** option is not supported for track assignment congestion maps.

- **detail**

Reports a detail route congestion map.

By default, the tool generates a new detail route congestion map from the current detail routing information. Before running this command, you need to ensure that the design is detail routed.

To report the existing detail route congestion map instead of generating a new one, use the **-no_reroute** option.

The **-search_repair** option is not supported for detail route congestion maps.

-search_repair

Includes the search and repair stage when performing global routing to generate the congestion map. In other words, global routing includes the "rip-up-and-reroute" stages. In this mode, the congestion number is better than that of the default mode, which does not include the "rip-up-and-reroute" stages.

This option applies only when the routing stage is **global**.

This option and the **-no_reroute** option are mutually exclusive; you can specify only one.

If you do not specify either option, the command generates a congestion map based on the routing stage specified in the **-routing_stage** option.

-no_reroute

Reports the existing congestion map for the specified routing stage from the design library.

The command runs faster when you specify this option because routing is not run. However, it is possible that the existing congestion map is not up to date, so you need to ensure that the existing congestion map is valid and up-to-date when using this option.

This option and the **-search_repair** option are mutually exclusive; you can specify only one.

If you do not specify either option, the command generates a congestion map based on the routing stage specified in the **-routing_stage** option.

-grc_based

Generates a GRC-based congestion report that provides information about the 10 worst global routing cells (GRCs) in terms of total congestion, horizontal congestion, and vertical congestion.

-grc_number *grc_number*

Generates a GRC-based congestion report that is similar to the report generated by the **-grc_based** option, except that it reports the specified number of global routing cells for each category, rather than 10.

-overflow_threshold *threshold*

Generates a GRC-based congestion report that is similar to the report generated by the **-grc_based** option, except that it reports only those global routing cells that exceed the specified overflow threshold, up to a maximum of 10 global routing cells per category.

-by_layer

Generates a layer-based congestion report, which includes a congestion summary for each routing layer, as well as the default congestion summary.

-coords *{rectangle}*

Generates a congestion report only for the specified region. The format for specifying a rectangle is *{llx lly urx ury}*. The coordinates of the rectangle are in microns.

This option and the **-polygon** option are mutually exclusive; you can specify only one.

If you do not specify either option, the tool generates a congestion report for the entire design.

`-polygon {polygon_area}`

Generates a congestion report only for the specified region. The format for specifying a polygon is `{{x1 y1} {x2 y2} {x3 y3}...{x1 y1}}`. Note that the first point and the last point in the list must be the same. The coordinates of the polygon are in microns.

This option and the **-coords** option are mutually exclusive; you can specify only one.

If you do not specify either option, the tool generates a congestion report for the entire design.

DESCRIPTION

This command generates a congestion map based on the specified routing stage. By default, it runs global routing, generates a new congestion map, and reports congestion statistics for the current design.

When running global routing, this command uses the default router, either the classic router or Zroute, as set by the Zroute route mode option. To set the Zroute route mode option, use the **set_route_mode_options -zroute** command.

For congestion analysis purposes, the design is divided into small regions called global routing cells (GRCs). A given global routing cell is considered overcongested if the actual number of wires passing through the global routing cell is greater than the number of available tracks in the global routing cell.

The default congestion report provides a summary of the congestion information, which includes the total congestion, the horizontal congestion, and the vertical congestion. You can report additional information by selected a GRC-based report (**-grc_based**, **-grc_number**, or **-overflow_threshold** option) or layer-based (**-by_layer** option) congestion report.

In the congestion report, the Overflow value is the total number of wires in the design global routing cells that do not have a corresponding track available; the Max value corresponds to the highest number of overutilized wires in a single global routing cell; and the GRCs value is the total number of overcongested global routing cells in the design.

This command creates an on-disk file in your Milkyway database.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example runs the **report_congestion** command.

```
prompt> report_congestion
```

```
...
```

```
Both Dirs: Overflow = 39 Max = 8 (1 GRCs) GRCs = 14 (0.26%)
```

```
H routing: Overflow = 6 Max = 2 (2 GRCs) GRCs = 4 (0.07%)
```

```
V routing: Overflow = 33 Max = 8 (1 GRCs) GRCs = 10 (0.18%)
```

report_congestion

1724

1

SEE ALSO

`route_global(2)`

report_congestion_options

Reports congestion options in the design.

SYNTAX

```
integer report_congestion_options  
[-all]  
id_list
```

ARGUMENTS

-all

Specifies to report all congestion options.

id_list

Reports congestion options with the specified ids.

DESCRIPTION

The **report_congestion_options** command reports the congestion options in the design. If **-all** is used, all congestion options in the design will be reported. If a list of ids is specified, those congestion options will be reported. The report will first report the design-wide congestion options. It will then report the regional congestion options, if any. The regional congestion options report includes the id, the coordinates, and the values.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following is an example report of the congestion options:

```
prompt> report_congestion_options -all  
  
Report congestion options:  
*****  
Congestion parameters for the design:  
  Horizontal threshold: 0.8 (default)  
  Vertical threshold: 0.8 (default)  
  Maximum utilization: 0.95 (default)  
Regional congestion options:  
  ID 0: {0 0 20 20}, MAX_UTIL: 0.5  
*****
```

SEE ALSO

`remove_congestion_options(2)`
`report_congestion(2)`
`set_congestion_options(2)`

report_constraint

Displays constraint-related information about a design.

SYNTAX

```
status report_constraint
[-all_violators]
[-verbose]
[-significant_digits digits]
[-max_area]
[-max_delay]
[-critical_range]
[-min_delay]
[-max_capacitance]
[-min_capacitance]
[-max_transition]
[-max_fanout]
[-cell_degradation]
[-min_porosity]
[-max_dynamic_power]
[-max_leakage_power]
[-max_net_length]
[-connection_class]
[-multiport_net]
[-nosplit]
[-max_toggle_rate]
[-max_total_power]
[-scenario scenario_list]
```

Data Types

<i>digits</i>	integer
<i>scenario_list</i>	list

ARGUMENTS

- all_violators
Displays a summary of all of the optimization and design rule constraints with violations in the current design. The **-verbose** option provides detailed information about constraint violations. Multiple violations for a given constraint are listed from the most to least violation.
- verbose
Displays more detail about constraint calculations.
- significant_digits *digits*
Specifies the number of digits to the right of the decimal point that are to be reported. The *digits* value must be between 0 and 13. The default value is 2. This option overrides the value set by the **report_default_significant_digits** variable.

-max_area
 Displays only the max_area constraint information. The default is to display all optimization and design rule constraints.

-max_delay
 Displays only the max_delay and setup information. The default is to display all optimization and design rule constraints.

-critical_range
 Displays only the critical_range information. The critical_range is a design rule used to inform the optimization engine that it should optimize near critical paths along with the most critical path. The default is to display all optimization and design rule constraints.

-min_delay
 Displays only the min_delay and hold information. The default is to display all optimization and design rule constraints.

-max_capacitance
 Displays only the max_capacitance constraint information. The max_capacitance constraint is a design rule used to limit total capacitance on a net. The default is to display all optimization and design rule constraints.

-min_capacitance
 Displays only the min_capacitance constraint information. The min_capacitance constraint is a design rule used to limit total capacitance on a net. The default is to display all optimization and design rule constraints.

-max_transition
 Displays only the max_transition constraint information. The max_transition constraint is a design rule used to limit transition time on a net. The default is to display all optimization and design rule constraints. If the library uses the cmos2 delay model, this option shows the max_edge_rate information.

-max_fanout
 Displays only **max_fanout** constraint information. The **max_fanout** constraint is a design rule used to limit fanout_load on a net. The default is to display all optimization and design rule constraints.

-cell_degradation
 Displays only **cell_degradation** constraint information. The **cell_degradation** is a design rule used to limit total capacitance on a net. This option differs from the max_capacitance rule, in that the total capacitance that can be driven by a cell is a function of the transition times at the inputs of the cell. The default is to display all optimization and design rule constraints.

-min_porosity
 Displays only the min_porosity constraint information. The min_porosity constraint is an optimization constraint for routability. The default is to display all optimization and design rule constraints.

-max_dynamic_power
 Displays only the `max_dynamic_power` constraint information. The default is to display power constraint information. Queries for power constraint information are only valid if a power-related license is available.

-max_leakage_power
 Displays only the **max_leakage_power** constraint information. The default is to display power constraint information. Queries for power constraint information are only valid if a power-related license is available.

-max_net_length
 Displays only `max_net_length` constraint information. The `max_net_length` constraint is a design rule used to limit route length on a net. The default is to display all optimization and design rule constraints.

-connection_class
 Displays only the `connection_class` constraint information. The `connection_class` constraint is displayed only if there is a `connection_class` violation.

-multiport_net
 Displays only the `multiport_net` constraint information. The `multiport_net` constraint is displayed only if the attribute **fix_multiple_port_nets** is set to the default value of **none**.

-nosplit
 Prevents line-splitting and facilitates writing software to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds its column's width, the next field begins on a new line, starting in the correct column.

-max_total_power
 Displays only the `max_total_power` constraint information. The default is to display power constraint information. Queries for power constraint information are only valid if a power-related license is available.

-scenario *scenario_list*
 Reports constraints for given list of scenarios of a multiscenario design. Inactive scenarios will be skipped in the report. Each scenario is reported separately. If this option is not given, `report_constraint` will report constraints on all active scenario except `-all_violators` and `-verbose` option. With these two options and without the `-scenario` option, `report_constraints` will only report constraints on current scenario.

DESCRIPTION

The **report_constraint** command displays the following information for the constraints on the current design:

- Whether the constraint was violated or met
- By how much the constraint value was violated or met

- The design object that was the worst violator.

The maximum delay information shows cost by path group. This includes violations of setup time on registers or ports with output delay as well as violations of **set_max_delay** commands. The total maximum delay cost is the sum of each group's weighted cost. For details on creating path groups, refer to the **group_path** command man page. To see the current path groups in the design, use the **report_path_group** command.

The minimum delay cost includes violation of hold time on registers or ports with output delay as well as violations of **set_min_delay** commands.

Multicorner-Multimode Support

By default, this command uses information from all active scenarios. You can select different scenarios by using the **-scenario** option.

EXAMPLES

The following example shows brief constraint information for the current design:

```
prompt> report_constraint
```

```
*****
Report : constraint
Design : counter
Version: 1998.02
Date   : Fri Dec 26 15:49:46 1997
*****
```

Group (max_delay/setup)	Cost	Weight	Weighted Cost
CLK	0.00	1.00	0.00
default	0.00	1.00	0.00
max_delay/setup			0.00

Group (critical_range)	Total Neg Slack	Critical Endpoints	Cost
CLK	0.00	0	0.00
default	0.00	0	0.00
critical_range			0.00

Constraint	Cost
max_transition	0.00 (MET)
max_fanout	0.00 (MET)
max_delay/setup	0.00 (MET)
critical_range	0.00 (MET)

min_delay/hold	0.40 (VIOLATED)
max_leakage_power	6.00 (VIOLATED)
max_dynamic_power	14.03 (VIOLATED)
max_area	48.00 (VIOLATED)
min_porosity	2.00 (VIOLATED)

The following example displays detailed constraint information for the current design:

prompt> **report_constraint -verbose**

Report : constraint
-verbose

Design : counter

Version: v3.1a

Date : Tue 1992

Startpoint: ffb (rising edge-triggered flip-flop clocked by CLK)

Endpoint: ffd (rising edge-triggered flip-flop clocked by CLK)

Path Group: CLK

Path Type: max

Point	Incr	Path

clock CLK (rise edge)	0.00	0.00
startpoint clock skew (ideal)	0.00	0.00
startpoint clock uncertainty	0.00	0.00
ffb/CP (FD3)	0.00	0.00 r
ffb/QN (FD3)	2.42	2.42 r
w/Z (ND4)	0.59	3.01 f
q/Z (EO)	1.13	4.14 f
j/Z (AO2)	1.08	5.22 r
ffd/D (FDS2)	0.00	5.22 r
data arrival time		5.22
clock CLK (rise edge)	10.00	10.00
endpoint clock skew (ideal)	0.00	10.00
endpoint clock uncertainty	0.00	10.00
ffd/CP (FDS2)	0.00	10.00 r
library setup time	-0.90	9.10
data required time		9.10

data required time		9.10
data arrival time		-5.22

slack (MET)		3.88

Design: counter

max_area	30.00
- Current Area	78.00

Slack	-48.00 (VIOLATED)

report_constraint

1732

Design: counter

max_leakage_power	70.00
- Current Leakage Power	76.00

Slack	-6.00 (VIOLATED)

Design: counter

max_dynamic_power	500.00
- Current Dynamic Power	514.03

Slack	-14.03 (VIOLATED)

The following example displays detailed information on only those constraints that have violations:

prompt> **report_constraint -all_violators -verbose**

Report : constraint
 -all_violators
 -verbose
Design : led
Version: v3.2a
Date : Tue Jan 3 13:00:45 1995

Startpoint: b (input port)
Endpoint: z5 (output port)
Path Group: default
Path Type: max

Point	Incr	Path

input external delay	0.00	0.00 r
b (in)	0.00	0.00 r
U5/Z (IV)	1.32	1.32 f
U3/Z (NR2)	3.35	4.67 r
U18/Z (AO6)	0.73	5.40 f
U22/Z (AO4)	1.42	6.82 r
z5 (out)	0.00	6.82 r
data arrival time		6.82
max_delay	6.50	6.50
output external delay	0.00	6.50
data required time		6.50

data required time		6.50
data arrival time		-6.82

slack (VIOLATED)		-0.32

Startpoint: c (input port)

Endpoint: z3 (output port)
 Path Group: default
 Path Type: max

Point	Incr	Path
input external delay	0.00	0.00 r
c (in)	0.00	0.00 r
U6/Z (IV)	1.34	1.34 f
U2/Z (NR2)	3.35	4.69 r
U15/Z (AO7)	0.87	5.56 f
U24/Z (AO3)	1.02	6.57 r
z3 (out)	0.00	6.57 r
data arrival time		6.57
max_delay	6.50	6.50
output external delay	0.00	6.50
data required time		6.50
data required time		6.50
data arrival time		-6.57
slack (VIOLATED)		-0.07

Net: a

max_transition	1.00
- Transition Time	1.26
Slack	-0.26 (VIOLATED)

Net: a

max_fanout	5.00
- Fanout	7.00
Slack	-2.00 (VIOLATED)

Design: led

max_area	30.00
- Current Area	36.00
Slack	-6.00 (VIOLATED)

Design: led

max_dynamic_power	1000.00
- Current Dynamic Power	1254.81
Slack	-254.81 (VIOLATED)

The following example displays the max_area, max_delay/setup, min_delay/hold, and max_leakage_power constraint information:

```
prompt> report_constraint -max_area -max_delay -min_delay \
-max_leakage_power
```

```
*****
Report : constraint
-max_area
-max_delay
-min_delay
-max_leakage_power
Design : led
Version: v3.2a
Date   : Tue Jan  3 13:00:56 1995
*****
```

Group (max_delay/setup)	Cost	Weight	Weighted Cost
default	0.32	1.00	0.32
max_delay/setup			0.32
Constraint			Cost
max_delay/setup			0.32 (VIOLATED)
max_area			6.00 (VIOLATED)

SEE ALSO

```
create_clock(2)
group_path(2)
report_clock(2)
report_design(2)
report_path_group(2)
report_timing(2)
report_timing_requirements(2)
set_critical_range(2)
set_max_area(2)
set_max_delay(2)
set_max_dynamic_power(2)
set_max_leakage_power(2)
set_max_net_length(2)
set_max_total_power(2)
```

report_critical_area

Calculates the critical area of the design.

SYNTAX

```
status report_critical_area
[-particle_distr_func_file filename]
[-input_layers layerlist]
[-fault_type short | open]
[-tsmc_encr_particle_distr_file]
[-layer_alias_DSD_format layeraliaslist]
[-suppress_zeros_in_report]
[-multiparticle_report_format]
```

Data Types

<i>filename</i>	string
<i>layerlist</i>	list
<i>layeraliaslist</i>	list

ARGUMENTS

- particle_distr_func_file *filename***
Specifies the particle probability function file. It can be in ASCII or Synopsys encrypted format.
If you do not specify this option, the tool uses an internal default particle probability function.
- input_layers *layerlist***
Specifies the metal layers (in the format m1 m2 m3 ... or M1 M2 M3 ... or metal1 metal2 metal3 ...) for which critical area should be calculated. The metal layer can also be specified as in the technology file.
If you do not specify this option, the tool calculates critical area for all metal layers.
- fault_type short | open**
Specifies the fault type.
If you do not specify this option, the tool calculates the short critical area.
- tsmc_encr_particle_distr_file**
Specifies that the particle distribution function is in TSMC encrypted format.
If you do not specify this option, the tool assumes the default format for the particle probability function.
- layer_alias_DSD_format *layeraliaslist***
Specifies the layer alias used for defect size distribution (DSD) of each layer. These numbers indicate the number of layers for each of these metal types. Each type of metal layer will have a DSD curve and D0 specified in the particle-distribution file. The order of these aliases is from bottom to top. M1 is always a separate type. Then above it are Mx layers, My layers, Mz or Mt layers, and Mr layers. Use this option along with the -

multilayer_DSD_format option.

-suppress_zeros_in_report

If critical area for a window is zero, this option suppresses that entry from the output heatmap file. This helps to reduce the size of the heatmap file. The default is to report windows with zero critical area.

-multiparticle_report_format

Specifies that the tool calculates and reports the critical area of each particle separately in the heatmap file. If you specify this option, windows with zero critical area are automatically suppressed to reduce the size of the heatmap file.

The default is to calculate the average critical area for all particles and report a single value.

DESCRIPTION

This command is used to calculate the critical area of a design. It can be used to calculate short critical area or open critical area. This command calculates the average critical area per metal layer. The input can be specified in two formats: a particle probability file or a multilayer defect size distribution (DSD) format. A particle probability file can be specified in text format or in Synopsys encrypted format. Use the **process_particle_probability_file** command to encrypt the particle probability file. The input in multilayer DSD format can be a text file or a TSMC encrypted file. If you do not specify an input particle probability file, a default particle probability function is used. The output is a text format heatmap file that is written to the current working directory. This heatmap is also stored as an attachment and can be displayed. A separate display heatmap is created for each fault type and each metal layer.

The layer aliases are specified in x, y, z, t, r format strictly in this order. This option is used only for input files in TSMC format. You must provide an integer value for each of x, y, z, t, r. The definitions for each of them should be in the TSMC format input file. 'x' layers starting from M2 going up use the 'Mx' particle distribution definition from input file. The next 'y' layers use the 'My' definition and so on. A nonzero value of 'z' is provided for 90 nm technology. A nonzero value of 't' is specified for 65 nm technology. A nonzero value for 'z' and 't' at the same time is treated as an error. The last 'r' layers use the 'Mr' particle distribution definition from input file.

For best results, run this command on the design with zero or minimum DRCs.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example calculates open critical area on metal layers m2 and m3 using the default particle probability function:

```
prompt> report_critical_area -fault_type {open} \  
-input_layers {m2 m3}
```

SEE ALSO

`process_particle_probability_file(2)`

report_crpr

Reports the clock reconvergence pessimism calculated between specified register clock pins or ports.

SYNTAX

```
status report_crpr  
-from from_latch_clock_pin  
-to to_latch_clock_pin  
[-from_clock from_clock]  
[-to_clock to_clock]  
[-setup | -hold]  
[-significant_digits digits]
```

Data Types

<i>from_latch_clock_pin</i>	string
<i>to_latch_clock_pin</i>	string
<i>from_clock</i>	string
<i>to_clock</i>	string
<i>digits</i>	integer

ARGUMENTS

-from *from_latch_clock_pin*
Specifies the clock pin of the launching sequential device or clock gating check to be reported. A constrained input port may also be used.

-to *to_latch_clock_pin*
Specifies the clock pin of the capturing sequential device or clock gating check to be reported. A constrained output port may also be used.

-from_clock *from_clock*
Specifies the clock that fans out to the launching sequential device.

-to_clock *to_clock*
Specifies the clock that fans out to the capturing sequential device.

-setup
Reports the clock reconvergence pessimism for a setup data path. The **-setup** and **-hold** options are mutually exclusive. If neither option is specified, **-setup** is assumed.

-hold
Reports the clock reconvergence pessimism for a hold data path. The **-setup** and **-hold** options are mutually exclusive. If neither option is specified, **-setup** is assumed.

-significant_digits *digits*
Specifies the number of digits to the right of the decimal point that are to be reported. Allowed values are 0-13. If this option is not specified, the number of significant digits is determined by the

report_default_significant_digits variable, which has a default value of 2.

DESCRIPTION

This command reports the clock reconvergence pessimism calculated between specified register clock pins or ports. The command displays the following information about the calculation of the clock reconvergence pessimism between the two specified sequential elements:

- The name of the common pin in the clock network.
- The clock edges (rising or falling) that propagate through the common point to the launching and capturing registers.
- The value of the variable **timing_crpr_threshold_ps**.
- A tabulation of the arrival times for both clock edges at the common point and their associated clock reconvergence pessimism (crp_rise and crp_fall).
- The clock reconvergence pessimism used along with the reasoning behind the selection of crp_rise or crp_fall used for that report.
- The potential range of accuracy of the clock reconvergence pessimism value that will appear in the corresponding timing report. The range of accuracy depends upon the value of the **timing_crpr_threshold_ps** variable.

If the **timing_remove_clock_reconvergence_pessimism** variable is set to false, clock reconvergence pessimism removal is inactive and no report can be generated. To determine the current value of the variable, type the following command:

```
prompt> printvar timing_remove_clock_reconvergence_pessimism
```

The **-from_clock** and **-to_clock** options allow you to generate a report for only the specified clocks. By default, a report is issued for all clocks that fanout to each sequential device.

Two clock reconvergence pessimism values are calculated for each potential common point in the design:

- The crp_rise value is calculated from rising arrival times at the common point.
- The crp_fall value is calculated from falling arrival times at the common point.

The value of clock reconvergence pessimism used in the report depends upon what clock edges propagate through the common point to the clock pins of the launching and capturing devices. For example, if a rising edge through the common point triggers the launching device and also triggers the capturing device, then a rising clock reconvergence pessimism value (crp_rise) is used. Similar reasoning applies for a falling edge propagating through the common point and subsequently launching and capturing data leading to a falling clock reconvergence pessimism value being used (crp_fall). If a rising edge through the common point launches the data and a falling edge through the common point captures it, then a mismatch in sense occurs.

If a mismatch occurs and the **timing_clock_reconvergence_pessimism** variable is set to **normal**, then the minimum of **crp_rise** and **crp_fall** is used. If a mismatch occurs and the variable is set to **same_transition**, then the clock reconvergence pessimism is reported as 0. This selection process is reported in the selection details section of the report. To determine the current value of the **timing_clock_reconvergence_pessimism** variable, type the following command:

```
prompt> printvar timing_clock_reconvergence_pessimism
```

When the capturing sequential device is a level-sensitive latch 2 clock reconvergence pessimism values are reported. The first value corresponds to the opening edge of the latch and appears in the timing report. The second value corresponds to the closing edge of the device. The selection details section also reports the decision making process behind both clock reconvergence pessimism values. In this case, since the opening and closing clock edges at the latch will always be different, there will always be a mismatch in clock edges for one of them.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example reports the clock reconvergence pessimism calculated between the *ffa* and *ffd* sequential elements for a setup data path:

```
prompt> report_crpr -from ffa/CP -to ffd/CP -setup
```

```
*****
Report : CRP Calculation
Design : counter
Version: 2004.12
Date   : Mon Nov 1 15:44:20 2004
*****
```

```
Startpoint: ffa (rising edge-triggered flip-flop clocked by CLK)
Endpoint: ffd (rising edge-triggered flip-flop clocked by CLK)
```

```
Common Point: CLK <inside>
Common Clock: CLK
Launching edge at common point: RISING
Capturing edge at common point: RISING
CRPR threshold: 0.02
```

Arrival Times	Early	Late	CRP
Rise	3.00	19.00	16.00
Fall	5.00	23.00	18.00

Selection Details

Edge Match: Match, using rise CRP

clock reconvergence pessimism 16.00

Range of accuracy of CRP in report_timing, due to value
of timing_crpr_threshold_ps: 15.98 <= CRP <= 16.00

1

The following example displays a report where the launching device is a flip-flop and the capturing device is a latch. In this case, the mismatch in clock edges occurs for the clock reconvergence pessimism corresponding to the closing edge at the latch.

prompt> **report_crpr -from reg1/CP -to lat2/G**

Report : CRP Calculation

Design : borrow

Version: 2004.12

Date : Mon Nov 1 15:44:20 2004

Startpoint: reg1 (rising edge-triggered flip-flop clocked by clk)

Endpoint: lat2 (positive level-sensitive latch clocked by clk)

Common Point: clk_buf/Z

Common Clock: clk

Launching edge at common point: RISING

Latch open edge at common point: RISING

CRPR threshold: 0.01

Arrival Times	Early	Late	CRP
Rise	2.1229	2.6529	0.5300
Fall	2.0868	2.7868	0.7000

Selection Details

Edge Match (opening): Match, using rise CRP

Edge Match (closing): Mismatch, using min(rise CRP, fall CRP)

clock reconvergence pessimism (open edge) 0.5300

clock reconvergence pessimism (close edge) 0.5300

Range of accuracy of CRP in report_timing, due to value
of timing_crpr_threshold_ps: 0.5200 <= CRP <= 0.5300

1

SEE ALSO

`timing_clock_reconvergence_pessimism(3)`
`timing_crpr_threshold_ps(3)`
`timing_remove_clock_reconvergence_pessimism(3)`

report_cts_batch_mode

Reports the clock tree synthesis batch mode status.

SYNTAX

status **report_cts_batch_mode**

ARGUMENTS

None

DESCRIPTION

To reduce runtime in script that call CTS commands many times, in some cases, one clock at a time, the set command turns on the batch mode. It skips CTS pre and post process whenever it can. For example, if 3 compile_clock_tree commands are called, the first call will run the CTS pre-process, but skipped the post process. The second and the third calls will skip both pre and post process. The following command, like reset_cts_batch_mode, save_mw_cel or other trigger commands will make up the CTS post process.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

```
set_cts_batch_mode
```

```
compile_clock_tree -clock_trees clk1
```

```
compile_clock_tree -clock_trees clk2
```

```
report_cts_batch_mode  
optimize_clock_tree  
balance_inter_clkok_delay  
reset_cts_batch_mode
```

SEE ALSO

```
reset_cts_batch_mode(2)  
set_cts_batch_mode(2)
```

report_delay_calculation

Displays the actual calculation of a timing arc delay value for a cell or net.

SYNTAX

```
status report_delay_calculation
-min
-max
-from from_pin
-to to_pin
[-nosplit]
[-crosstalk]
[-from_rise_transition from_rise_value]
[-from_fall_transition from_fall_value]
```

Data Types

<i>from_pin</i>	string
<i>to_pin</i>	string
<i>from_rise_value</i>	float
<i>from_fall_value</i>	float

ARGUMENTS

-min

Specifies that the report must show how a minimum delay value is computed. If you do not specify this option, the report shows how a maximum delay value is computed.

-max

Specifies that the report must show how a maximum delay value is computed. This is the default behavior. This option cannot be used with the **-min** option.

-from *from_pin*

Specifies the starting point of a timing arc within a design. This option must be used along with **-to** *to_pin*. When this option is specified, you cannot specify a collection of multiple pins. For a cell timing arc, the pins must represent a common leaf cell's input and output pins, which have a timing arc specified between them in the library. For a net timing arc, the pins must be a driver and a load on a common net. The specified pins must be associated with library cell instances connected by the net. Port names are allowed in place of a pin name for net arcs. A hierarchical pin is not a valid pin for this option.

-to *to_pin*

Specifies the ending point of a timing arc within a design. This option must be used along with **-from** *from_pin*. When this option is specified, you cannot specify a collection of multiple pins. For a cell timing arc, the pins must represent a common leaf cell's input and output pins, which have a timing arc specified between them in the library. For a net timing arc, the pins must be a driver and a load on a common net. The specified pins must be associated with library cell instances connected by the net. Port names are allowed in

place of a pin name for net arcs. A hierarchical pin is not a valid pin for this option.

-nosplit

Prevents line splitting and facilitates writing software to extract information from the report output. Most of the delay data is listed in fixed-width columns. If the text for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

-crosstalk

Reports the crosstalk information for a net arc. The arc is specified by *from_pin* and *to_pin*.

-from_rise_transition *from_rise_value*

Specifies the from rise transition value used by the delay calculation. The specified value is in main library units.

-from_fall_transition *from_fall_value*

Specifies the from fall transition value used by the delay calculation. The specified value is in main library units.

DESCRIPTION

This command provides detailed timing calculation information about the specified cell or net timing arc. You can use this information for debugging or verifying timing data in a technology library. Before using this command to display details of a cell timing arc, read the technology library file into the system using the **read_lib** command. The **read_lib** command automatically compiles the library and enables the **report_delay_calculation** command for cell timing arcs. Otherwise, the built-in security mechanism terminates the command when issued for a cell timing arc. This restriction does not apply to net timing arcs.

Both operating conditions and wire load models are taken into account when making delay calculations. Timing ranges are not taken into account because they typically apply to an entire path.

The **-crosstalk** option on net arc reports the aggressor and victim information for both rise and fall. The option prints coupling capacitance, the driving library cell, and the clocks reaching the net (both victim and aggressor). The aggressors have "Switching Bump," which is used for prioritizing the aggressors and also have aggressor attributes (active or screened).

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following examples show reports generated using **report_delay_calculation**:

```
prompt> report_delay_calculation -from BLK1/A -to BLK1/Z
```

```
report_delay_calculation
1746
```

Report : delay_calculation
Design : led
Version: v3.1a
Date : Tue Apr 7 16:52:43 1992

From : BLK1/A
To : BLK1/Z

arc type : cell
arc sense : inverting
Input net transition times: Dt_rise = 0, Dt_fall = 0

Rise Delay computation:

rise_intrinsic	0 +
rise_slope * Dt_fall	0 * 0 +
rise_resistance * (pin_cap + wire_cap) / driver_count	
0.2 * (0 + 0.53) / 1	
rise_transition_delay	0.106

Total	0.106

Fall Delay computation:

fall_intrinsic	0 +
fall_slope * Dt_rise	0 * 0 +
fall_resistance * (pin_cap + wire_cap) / driver_count	
0.15 * (0 + 0.53) / 1	
fall_transition_delay	0.0795

Total	0.0795

prompt> **report_delay_calculation -from BLK1/Z -to BLK2/A**

Report : delay_calculation
Design : led
Version: v3.1a
Date : Tue Apr 7 16:28:07 1992

From : BLK1/Z
To : BLK2/A

arc type : net

Operating Conditions: BASIC_WORST Library: basic
Wire Load Model Mode: top

Design	Wire Loading Model	Library

RDC_GENERIC	BASIC_ONE	basic

Balanced case tree

```
equation : (r_wire/load_count) * (c_pins + c_wire/load_count)
(0.53 / 1) * (1 + 0.53 / 1)
delay rise, fall : 0.608175 , 0.608175
```

```
prompt> report_delay_calculation \
        -from exetop0/e_dptop0/e_flag0/I27/Y \
        -to exetop0/e_dptop0/e_flag0/U1/A -crosstalk
```

```
*****
Report : delay_calculation
Design : Xtalk_test
Version: A-2007.12
Date   : Tue Sep 25 23:18:25 2007
*****
```

```
From pin: exetop0/e_dptop0/e_flag0/I27/Y
To pin:   exetop0/e_dptop0/e_flag0/U1/A
```

* Some/all delay information is back-annotated.

Operating Conditions: slow Library: slow

```
Annotated max rise net delta delay: 0.011865 arc delay: 0.014991
Annotated max fall net delta delay: 0.006210 arc delay: 0.009337
```

```
Annotated max rise net delta transition: 0.025998 pin transition: 0.247500
Annotated max fall net delta transition: 0.013779 pin transition: 0.240000
```

```
Reporting for Crosstalk:
Victim net name: exetop0/e_dptop0/e_flag0/N194
Number of aggressors: 5
Number of effective (non-filtered) aggressors: 5
Victim driver rail voltage(VDD): 1.620000
```

```
Attributes:
A - aggressor is Active
S - aggressor is screened
```

Victim is rising:

Victim Net	Coupling Cap	Driver Lib Cell	Clocks
exetop0/e_dptop0/e_flag0/N194	0.013922	MX4X4	CK108M

Aggressor Net	Coupling Cap	Driver Lib Cell	Clocks	Attributes	Switching Bump (ratio of VDD)
exetop0/e_rndm0/n35	0.004685	INVX4	CK108M	A	0.023494
exetop0/e_rndm0/n18454	0.001537	SEDFFX4	CK108M	A	0.018957
exetop0/e_rndm0/n10	0.003462	BUFX2	CK108M	A	0.017911
exetop0/e_rndm0/n19178					

exetop0/s_AEI2_0_	0.001858	INVX2	CK108M	S	-
	0.002379	SEDFFX4	CK108M	S	-
Victim is falling:					
Victim	Coupling	Driver	Clocks		
Net	Cap	Lib Cell			
-----	-----	-----	-----		
exetop0/e_dptop0/e_flag0/N194					
	0.013922	MX4X4	CK108M		
Aggressor	Coupling	Driver	Clocks	Attributes	Switching Bump
Net	Cap	Lib Cell			(ratio of VDD)
-----	-----	-----	-----	-----	-----
exetop0/e_rndm0/n18454					
	0.001537	SEDFFX4	CK108M	A	0.016719
exetop0/e_rndm0/n35					
	0.004685	INVX4	CK108M	A	0.014618
exetop0/e_rndm0/n10					
	0.003462	BUFX2	CK108M	S	-
exetop0/e_rndm0/n19178					
	0.001858	INVX2	CK108M	S	-
exetop0/s_AEI2_0_					

SEE ALSO

```
report_lib(2)
report_timing(2)
rc_driver_model_mode(3)
rc_receiver_model_mode(3)
```

report_delay_estimation_options

Reports the parameters that influence delay estimation.

SYNTAX

status **report_delay_estimation_options**

ARGUMENTS

The **report_delay_estimation_options** command has no arguments.

DESCRIPTION

The **report_delay_estimation_options** command reports the parameters that influence the delay calculation the tool uses in placement and routing estimation.

The scaling factors are scenario aware. When you switch the current scenario, only the scaling factors for the current scenario will be printed out.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example runs the **report_delay_estimation_options** command:

```
prompt>set_delay_estimation_options
-min_unit_horizontal_capacitance 0.0015 -min_unit_vertical_capacitance 0.0010 \
-min_unit_horizontal_resistance 0.15 -min_unit_vertical_resistance 0.10 \
-max_unit_horizontal_capacitance 0.0030 -max_unit_vertical_capacitance 0.0035 \
-max_unit_horizontal_resistance 0.25 -max_unit_vertical_resistance 0.20 \
-min_unit_horizontal_capacitance_scaling_factor 1.0015 \
-min_unit_vertical_capacitance_scaling_factor 1.0010 \
-min_unit_horizontal_resistance_scaling_factor 1.15 \
-min_unit_vertical_resistance_scaling_factor 1.10 \
-max_unit_horizontal_capacitance_scaling_factor 1.0030 \
-max_unit_vertical_capacitance_scaling_factor 1.0035 \
-max_unit_horizontal_resistance_scaling_factor 1.25 \
-max_unit_vertical_resistance_scaling_factor 1.20 \
-max_via_resistance 0.25 -max_via_resistance 0.20 \
-max_via_resistance_scaling_factor 1.25 -max_via_resistance_scaling_factor 1.20
```

```
prompt> report_delay_estimation_options
```

Report delay estimation options:

Delay estimation options for the design:

```
report_delay_estimation_options
1750
```

```
Maximum horizontal unit capacitance: 0.003
Minimum horizontal unit capacitance: 0.0015
Maximum vertical unit capacitance: 0.0035
Minimum vertical unit capacitance: 0.001
Maximum horizontal unit resistance: 0.25
Minimum horizontal unit resistance: 0.15
Maximum vertical unit resistance: 0.2
Minimum vertical unit resistance: 0.1
Maximum horizontal unit capacitance scaling factor: 1.003
Minimum horizontal unit capacitance scaling factor: 1.0015
Maximum vertical unit capacitance scaling factor: 1.0035
Minimum vertical unit capacitance scaling factor: 1.001
Maximum horizontal unit resistance scaling factor: 1.25
Minimum horizontal unit resistance scaling factor: 1.15
Maximum vertical unit resistance scaling factor: 1.2
Minimum vertical unit resistance scaling factor: 1.1
Maximum via resistance: 0.2
Maximum via resistance scaling factor: 1.2
*****
```

SEE ALSO

`set_delay_estimation_options(2)`

report_design

Displays attributes of the current design.

SYNTAX

```
int report_design  
[-nosplit]  
[-physical]
```

ARGUMENTS

-nosplit

Prevents line-splitting and facilitates writing software to extract information from the report output. Design information is listed in fixed-width columns. If information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

-physical

Reports details about the physical state of the design in various sections. Design Statistics Section: number of macro cells, module cells, pins, IO pad cells, IO pins, nets and average pins per net.

Chip Utilization section: Total Area for: macro cells, std cells, blockages, pad cells, core area, pad core area, chip area. As well, reports width and height for chip and pad core. Also core width/height if core is not rectilinear.

Utilization: std cell [(std cell area) / (core - blockage area)] cell/core [(std cell + macro area)/(core area)] cell/chip [(std cell + macro + pad area) / (chip area)]

Other: number of cell rows

Master Instantiation Section: Name of master cells, their type and instantiation count

Timing/Optimization section: Various cell type/slack statistics (if applicable)

Global Routing Information: Details about the routing (if applicable) including Bounding Box, metal layers, track details, etc

Track Assignment Information: wire length estimates/statistics

Other sections: DRC information, Ring Wiring Statistics, Stripe Wiring Statistics, User Wiring Statistics, PG follow-pin Wiring Statistics, Signal Wiring Statistics

DESCRIPTION

The **report_design** command lists information about the attributes of the design.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following is an example of a design report:

prompt> **report_design**

```
*****
Report : design
Design : counter
Version: v3.0
Date   : Tue 1992
*****
```

Library(ies) Used:

tech_lib (File: /usr/synopsys/libraries/tech_lib.db)

Flip-Flop Types:

No flip-flop types specified.

Latch Types:

No latch types specified.

Operating Conditions:

Name	Library	Process	Temp	Volt	Interconnect Model
WCCOM	tech_lib	1.50	70.00	4.75	worst_case_tree

Wire Loading Model:

Selected manually by the user.

Name	Library	Res	Cap	Area	Slope	Fanout	Length
05x05	tech_lib	.0000	1.0000	0.0000	0.1860	1	0.3900

Wire Loading Model Mode: top.

Timing Ranges:

No timing ranges specified.

Pin Input Delays:

Pin	Input Delay				Clock	
	Min		Max			
	Rise	Fall	Rise	Fall		
U1/A	4.50	4.50	4.50	4.50	--	

Pin Output Delays:

None specified.

Disabled Timing Arcs:

No arcs disabled.

Required Licenses:

None Required

Design Parameters:

width => 16

SEE ALSO

report_clock(2)
report_internal_loads(2)
report_port(2)
report_placement_utilization(2)

report_design_lib

Lists the design units contained in the specified libraries.

SYNTAX

```
int report_design_lib  
[-libraries] [-designs] [-architectures] [-packages] [library_list]
```

Data Types

library_list list

ARGUMENTS

-libraries
Indicates that libraries, and not their contents, are to be listed.

-designs
Indicates that designs in libraries (Verilog modules, VHDL entities, or configurations) are to be listed.

-architectures
Indicates that designs in libraries, plus their architectures, are to be listed.

-packages
Indicates that packages in specified libraries are to be listed.

library_list
Specifies a list of libraries whose contents are to be displayed. If no *library_list* is specified, all libraries are displayed.

DESCRIPTION

Lists the contents of specified libraries.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses **report_design_lib** to list the contents of the current design libraries:

```
prompt> report_design_lib  
  
*****  
Report : design libraries
```

```

Version: v3.0
Date   : Tue Oct  1 14:44:53 1991
*****
Contents of current design libraries
DEFAULT (/users/Jane_Doe/design/test/work)
WORK (/users/Jane_Doe/design/test/work)
package:pack1
entity:add
architecture: dfast(add)
architecture: m dsmall(add)
entity: pmult
architecture: mverilog(mult)
TYPES_LIB (/users/Jane_Doe/design/report/types_lib)
package: ntop_pack
FUNCS_LIB (/users/Jane_Doe/design/report/funcs_lib)
package: spack
SYNOPSYS (/users/Jane_Doe/design/top/dc/libraries/syn/packages)
package:ATTRIBUTES
package:TYPES
package:synopsys

p --- This design has parameters.
m --- This architecture is the most recently analyzed.
n --- Can't find the source for this file.
s --- This file is out of date with respect to its source.
d --- This file is out of date with respect to its depends.

```

SEE ALSO

read_file(2)

report_design_physical

Displays design-related information for the current design.

SYNTAX

```
status report_design_physical  
[-design_setup]  
[-netlist]  
[-floorplan]  
[-route]  
[-utilization]  
[-all]  
[-verbose]
```

ARGUMENTS

-design_setup

Reports design setup information, such as the link library; target library; search path; and units in the logic library, Milkyway design library, and the technology file.

The design setup information is reported if you specify this option, specify the **-all** option, or specify no options.

-netlist

Reports logical netlist information, such as information about the cells, nets, pins, and ports.

For cells, the reported information includes the instance type, such as macro, ILM, or physical-only cell; the gate count; and the cell area in both square microns and sites.

The logical netlist information is reported only if you specify this option or specify the **-all** option.

-floorplan

Reports floorplan information, such as information about site rows, layers, voltage areas, groups, bounds, and power domains.

The floorplan information is reported if you specify this option, specify the **-all** option, or specify no options.

-route

Reports routing information, such as information about wires and via counts. The routing information is reported only if you specify this option or the **-all** option.

-utilization

Reports utilization information, such as various site and area utilization numbers and power net options.

The utilization number reported by this command is approximately equal to that reported by the **report_placement_utilization** command.

The utilization information is reported only if you specify this option or the **-all** option.

-all

Reports the design setup, logical netlist, floorplan, routing, and utilization information. The behavior is the same as if you specified the **-design_setup**, **-netlist**, **-floorplan**, **-routing**, and **-utilization** options.

-verbose

Reports additional information when you specify the **-netlist**, **-floorplan**, and **-utilization** options.

If you specify this option with the **-netlist** option, the tool reports reference instantiation information. If you specify this option with the **-floorplan** or **-utilization** option, the tool reports the number of islands in the blockage section.

DESCRIPTION

The **report_design_physical** command reports information about the current design.

If you do not specify any options, the report includes design setup information and floorplan information. For more information about what is reported, see the descriptions for the individual options.

EXAMPLES

The following is an example of a design report with the **-design_setup** and **-floorplan** options:

```
prompt> report_design_physical -design_setup -floorplan
```

```
-----
Report   : report_design_physical
          -floorplan
          -design_setup
```

```
Date      : Thu Mar 12 03:45:40 2009
```

```
Version   : v3.0
-----
```

DESIGN SETUP INFORMATION

```
-----
Host Name       : jxtopt2
Working Directory : /home/snps/myDir
Library Name    : mw_design
Cell Name       : top_placement_1.CEL;12
Library settings :
  Search Path    : .
                  /usr/mylibs/synopsys/libraries/syn
  Target Libraries : slow.db
  Link Libraries  : *
                  slow.db
                  dw01.sldb
                  dw02.sldb
                  dw_foundation.sldb
  Milkyway Ref Libraries : /usr/mylibs/MW/MWRefLibs
```

```
Units          :
                Library   Milkyway TechFile
```

Time	ns	ns	ns
Capacitance	pF	pF	pF
Resistance	kOhm	kOhm	kOhm
Power	-	-	pW
Current	uA	-	mA
Voltage	V	V	V

FLOORPLAN INFORMATION

SITE ROW INFORMATION

Site Type	Hrows	Vrows	Tiles	Width	Height	Row area
unit	57	0	26049	0.46	3.69	44215.57

CHIP AND CORE INFORMATION

	Width	Height	Area
Core	210.220	210.330	44215.573
Chip	210.220	210.330	44215.573

ROUTE GUIDES

No route guides exist

VOLTAGE AREAS

Name	Area	Util	bbox_llx	bbox_lly	bbox_urx	bbox_ury	GB-X	GB-Y
DEFAULT_VA	44215.57	0.6410	0.00	0.00	210.22	210.33	0	0

PLAN GROUPS

PG Name	Area	Util	bbox_llx	bbox_lly	bbox_urx	bbox_ury	
Padding							
							L
B	R	T					
PlanGroup1	40000.00	0.0141	200.00	300.00	400.00	500.00	0.000
0.000	0.000	0.000					

EXCLUSIVE MOVE BOUNDS

No exclusive move bounds exist

MOVE BOUNDS

	count	Area
Hard MoveBounds	2	960.00
Soft MoveBounds	1	484.00
No Of GroupBounds	1	

RELATIVE PLACEMENT GROUPS

No relative placement groups exist

POWER DOMAINS

No power domains exist

LAYERS

Name	Dir	Ign	Pitch	Spacing	Width
METAL1	Hor	NO	0.410	0.180	0.160
METAL2	Ver	NO	0.460	0.210	0.200
METAL3	Hor	NO	0.410	0.210	0.200
METAL4	Ver	NO	0.460	0.210	0.200
METAL5	Hor	NO	0.410	0.210	0.200
METAL6	Ver	NO	0.460	0.210	0.200

SEE ALSO

report_clock(2)
report_design(2)
report_port(2)
report_placement_utilization(2)

report_direct_power_rail_tie

Reports all the library pins on which the **direct_power_rail_tie** attribute is set to true.

SYNTAX

```
int report_direct_power_rail_tie
```

ARGUMENTS

The **report_direct_power_rail_tie** command has no arguments.

DESCRIPTION

Reports all the library pins on which the attribute **direct_power_rail_tie** is set to true.

If the value of **direct_power_rail_tie** attribute on a library pin is true, then all the pins in the design, for which this is the library pin, it won't connect it to a tieoff cell for connecting it to constant signal. Instead, it will keep them connecting to generic constant signals, so that during power routing these pins can be connected directly to power rails.

To remove **direct_power_rail_tie** attribute, use **remove_attribute**.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command reports all the library pins which have the **direct_power_rail_tie** attribute set to true on them.

```
prompt> report_direct_power_rail_tie target_lib/leaf_cell/GND target_lib/  
leaf_cell/GND
```

SEE ALSO

```
set_direct_power_rail_tie(2)
```

report_disable_timing

Reports disabled timing arcs in the current design.

SYNTAX

```
string report_disable_timing  
[-nosplit]
```

ARGUMENTS

-nosplit

Prevents line splitting and facilitates writing software to extract information from the report output. By default, most design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line starting in the correct column.

DESCRIPTION

The **report_disable_timing** command reports disabled timing arcs in the current design. Timing arcs can be disabled in three ways. The first way is by using the **set_disable_timing** command. The second and third ways are both through automatically disabled arcs by the synthesis timing engine. This automatic disabling occurs in order to break timing loops or when propagating constants in the design.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example disables the timing arc of a cell named *U1/U2* from pin *A* to pin *Z*:

```
prompt> set_disable_timing {U1/U2} -from A -to Z
```

```
prompt> report_disable_timing
```

```
*****  
Report : disable_timing  
Design : middle  
*****
```

```
Flags :      c  case-analysis  
          C  Conditional arc  
          l  loop breaking  
          u  user-defined  
          L  stored loop breaking
```


Cell or Port	From	To	Flag
U1/U2	A	Z	u
U1/U2	B	Z	L

The following example disabled the timing arc of a cell named *ff1* from pin *CP* to pin *TE* and *TI* as a result of a case-analysis constant propagation to pin *TE* of cell *ff1*. Note that the L flag appears only when such arcs exist.

```
prompt> set_case_analysis 0 {ff1/TE}
```

```
prompt> report_disable_timing
```

```
*****
Report : disable_timing
Design : middle
*****
```

```
Flags :      c  case-analysis
           C  Conditional arc
           l  loop breaking
           u  user-defined
```

Cell or Port	From	To	Flag
ff4	CP	TI	c
ff4	CP	TE	c

SEE ALSO

```
set_case_analysis(2)
set_disable_timing(2)
```

report_distributed_hosts

Creates a detailed report on user-defined distributed hosts in the farm's pool

SYNTAX

Boolean **report_distributed_hosts**

DESCRIPTION

The **report_distributed_hosts** generates a report on user-defined distributed hosts in the farm's pool. The following information is displayed for each distributed processor.

Farm indicates the type of the farm from which the distributed processor was acquired, one of 'lsf', 'grd', 'generic' or 'now'.

Status indicates whether the distributed processor was launched by issuing 'successfully launched' on success and 'Not yet launched' when the distributed processor failed to launch or has not yet been launched.

Host Name indicates the name of the machine, grid name, or lsf name from which the distributed processor was acquired.

Command is the command used to launch the distributed processor.

EXAMPLES

In the following example a distributed processor is allocated from a workstation called platinum1. The processor is launched using the **create_distributed_farm** command. The **report_distributed_hosts** command then generates a report on the processor.

```
prompt> add_distributed_hosts -farm now -32bit platinum1
1
```

```
prompt> create_distributed_farm
```

```
1] Launched: rsh -n -l user platinum1 pt_shell -multi_scenario -32bit
-__zdp_slave ptsrv2 16874 16859 1 /user/work PrimeTime 4106 4108 53205
      Status: Forking successful
      Stdout: your job
      Stderr: **<<EMPTY>>**
```

```
-----
Waiting for  1 (of  1) distributed hosts
-----
```

```
prompt> report_distributed_hosts
```

```
report_distributed_hosts
1764
```

```
1] Farm      : now
   Status    : Successfully launched
   Host Name: platinum1
   Command   : pt_shell -multi_scenario -32bit
```

SEE ALSO

`add_distributed_hosts(2)`

report_distributed_route

Return status of a specified job and total number of available CPUs.

SYNTAX

```
integer report_distributed_route  
-job jobId
```

Data Types

jobId integer

ARGUMENTS

```
-job jobId  
      Enter a job ID.
```

DESCRIPTION

Return status of a specified job ID and the total number of CPUs available to use on running the job. If the job is existing and still running, then the current status will be returned. The failure will be returned if the specified job was killed or interrupted (or not existing), and no number of CPUs will be returned, either.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example is to report status of job with ID number 100 and current available CPUs.

```
prompt> report_distributed_route -job 100
```

SEE ALSO

```
set_distributed_route(2)  
remove_distributed_route(2)  
close_distributed_route(2)
```

report_drc_error_type

Outputs a formatted report on the specified error types.

SYNTAX

```
status report_drc_error_type  
[-type error_type | -class type_class]  
[-error_view mw_error_view]
```

Data Types

<i>error_type</i>	list
<i>type_class</i>	list
<i>mw_error_view</i>	list or collection of one item

ARGUMENTS

-type *error_type*
Specifies the error type ids or names on which to report.
This option is mutually exclusive with the **-class** option. If you do not specify either the **-type** or **-class** option, the tool reports on all error types in the specified error view.

-class *type_class*
Specifies the type classes on which to report. You must specify one or more of the following type classes: default, open, openlocator, short, and spacing. The type class keywords are case insensitive.
If you specify this option, the report is generated for all error types in the specified classes. This option is mutually exclusive with the **-type** option. If you do not specify either the **-type** or **-class** option, the tool reports on all error types in the specified error view.

-error_view *mw_error_view*
Specifies the error view on which to report on the error types.
If you do not specify this option, the tool reports on the error types in the current top-level design. You can specify only a single error view; specifying more than one error view causes an error.

DESCRIPTION

This command outputs a formatted report on the specified error types.

EXAMPLES

The following example opens an error view named mydesign_idrc.err and outputs a report on the error type named Overlap:

```
prompt> set cellId [open_mw_cel -not_as_current mydesign_idrc.err]  
{mydesign_idrc}  
prompt> report_drc_error_type -name Overlap -error_view $cellId
```

```

*****
Report          : DRC_Error Types
Design cel      : mydesign
Error view cel  : mydesign_idrc
Version         : C-2009.06
Date            : Thu Dec 18 10:27:02 2008
*****

```

Data	Value

Error type ID	1024
Error type name	Overlap
Number of errors	2
Type class	Default
Description	Metal and blockage overlap
Default error status	Error Must-Fix
Bbox coordinates	{{10.000 10.000} {120.000 30.000}}

SEE ALSO

```

create_drc_error_type(2)
list_drc_error_types(2)

```

report_droute_options

Reports the persistent detail route options, their current value, default value, range, type, and a brief description.

SYNTAX

```
status report_droute_options  
[-name name]
```

Data Types

name string

ARGUMENTS

`-name name`
 Specifies the name or name pattern of the option to be reported. Name pattern will report all options that match it. By default, if "-name" is not specified, all droute options are reported.

DESCRIPTION

The **report_droute_options** command shows options that tune detailed router operation. These settings are persistent in the Milkyway design. Cell-persistent options are saved in cell data base by **set_droute_options** command. Before printing options, **report_droute_options** loads all droute options from currently opened cell. All options are optional. This command prints all available detailed router options, their current value, default value, range, type and a brief description. However, if you want to see a specific option or options that match a specified name pattern, use argument "-name". You can set detailed router options with command **set_droute_options**.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

```
prompt> report_droute_options -name shiftVarWidthWire
```

```
(Integer Option for "droute" "shiftVarWidthWire" 1)  
;;        range [0,2], default=0, stored in cell;  
;;        0: put wide wires on routing tracks;  
;;        1: shift wide wires by half-pitch if neccessary to improve routing  
;;        2: shift NDR wires to optimize for minSpace, maxSpace and metal_fill  
;;        (experimental feature).
```

Name pattern will report all options that match it,
for example M9MinEdgeLength matches 3 options:

```

prompt> report_droute_options -name M9MinEdgeLength

(Real Option for "droute" "M9MinEdgeLength4" 0.000)
;;      range [0.000,1.000], default=0.000, stored in cell;
;;      N: minEdgeLength4

(Real Option for "droute" "M9MinEdgeLength5" 0.000)
;;      range [0.000,1.000], default=0.000, stored in cell;
;;      N: minEdgeLength5

(Real Option for "droute" "M9MinEdgeLength6" 0.000)
;;      range [0.000,1.000], default=0.000, stored in cell;
;;      N: minEdgeLength6

prompt> report_droute_options

Integer Option for droute removeObsoleteStub 0
#      range [0,1], default=0, stored in cell;
;;      0: leave obsolete stubs in routing solution
;;      1: try to remove obsolete stubs at the end of each Sbox routing.

Integer Option for droute accessPolyPin 1
#      range [0,1], default=1, stored in cell;
;;      0: ignore Poly pins,
;;      1: access Poly pins if there are any.

etc.

Real Option for droute defaultDiodeProtect 0.000
#      range [0.000,1000000.000], default=0.000, stored in cell;
;;      N: default diodeProtection value for all unspecified output ports
;;      except top-cell ports (used when doAntennaConx == 4)
Real Option for droute defaultGateSize 0.000
#      range [0.000,1000000.000], default=0.000, stored in cell;
;;      N: default gateSize value for all unspecified input ports

etc.

```

SEE ALSO

set_droute_options(2)

report_eco_history

Reports ECO history information and generates a change list file in the Tcl format.

SYNTAX

```
int report_eco_history  
[-output outfile_name]  
[-summary]  
[-id id_number]
```

Data Types

<i>outfile_name</i>	string
<i>id_number</i>	integer

ARGUMENTS

- output *outfile_name***
Specifies the name of the change list file to be created for saving ECO history information. When not specified, the command writes the report to the IC Compiler log file.
- summary**
Summarizes the ECO changes that have been made to the cell. This option cannot be used with the **-id** option.
- id *id_number***
Specifies the ID of the ECO process to be reported. When not specified, the command reports the latest ECO changes that were done through the **eco_netlist** command. This option cannot be used with the **-summary** option.

DESCRIPTION

This command reports either all the ECO changes or the changes of a specific ECO process which have been applied on the current editing cell through the **eco_netlist** command. The report can be saved to an output change list file in Tcl format.

NOTE:

1. Only the changes made by the **eco_netlist** command can be reported.
2. One invocation of **eco_netlist** produces a single ECO ID. Running **eco_netlist** multiple times in one session produces multiple ECO IDs for each run.
3. When running **report_eco_history** right after **eco_netlist**, **report_eco_history** reports the ECO change that was just done, even though it has not yet been saved. But if you close the cell without saving and then reopen the cell, the ECO change that was just done will not be reported.

EXAMPLES

The following are the examples of reporting ECO history information on Milkyway design cells.

Example 1: Run **report_eco_history** to report the latest ECO change list.

```
prompt> report_eco_history
# ECO DATE: Mon Mar 23 03:07:01 2009
# ECO AUTHOR: Synopsys
create_cell {ci2} AND2T
connect_net {inp} {ci2/B}
connect_net {inp} {ci2/A}
connect_net {outp} {ci2/X}
```

Example 2: Run **report_eco_history** to report the ECO change list and save it to the ECO change file.

```
prompt> report_eco_history
               -output eco_change.tcl
```

Example 3: Run **report_eco_history** to summarize all ECO history information in the report.

```
prompt> report_eco_history
               -summary
ID      Date          Author
-----
2       Mon Mar 23 03:07 Synopsys
1       Mon Mar 23 02:07 Synopsys
```

Example 4: Run **report_eco_history** to generate a specific ECO change list by ECO ID and save the list to the ECO change file.

```
prompt> report_eco_history
               -id 2
               -output eco_change_2.tcl
```

SEE ALSO

eco_netlist(2)

report_edit_groups

Reports details about edit groups.

SYNTAX

```
status report_edit_groups
[edit_groups]
```

Data Types

edit_groups collection

ARGUMENTS

edit_groups
Specifies the edit groups to be included in the report. By default, the report lists all of the edit groups in the design.

DESCRIPTION

This command produces a report that lists details about the specified edit groups or all of the edit groups in the design.

The report lists details for each edit group in the following format:

```
Edit Group: <name>
  BBox: {{<xmin> <ymin>} {<xmax> <ymax>}}
  Objects: (#<number of objects>)
    <objects>
```

In this format,

<name> is the name of the edit group.

<xmin> and <ymin> are the coordinates for the lower
left corner of the bounding box.

<xmax> and <ymax> are the coordinates for the upper
right corner of the bounding box.

<number of objects> is the number of child objects
in the edit group.

<objects> lists the types and names of the child objects,
with one line for each object.

EXAMPLES

The following example reports all edit groups in the design:

```
prompt> report_edit_groups
```

SEE ALSO

```
create_edit_group(2)  
get_edit_groups(2)  
remove_edit_groups(2)
```

report_error_coordinates

Displays the coordinates of all error objects in the error cell.

SYNTAX

```
status report_error_coordinates  
error_cell_name
```

Data Types

```
error_cell_name      string
```

ARGUMENTS

```
error_cell_name  
    Specifies the error cell of interest. The error objects in this error cell  
    will be reported.
```

DESCRIPTION

This command will list all the error objects in the given error cell. To run this command, the library containing the error cell must be opened first. Error cells can be created by commands such as DRC and LVS verification commands. The information that is reported includes:

- The type of summary being reported
- The distance unit used in the library
- Each error type along with a summary, the number of error objects, and the coordinates of all the error objects.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows what the output of report_error_coordinates looks like.

```
prompt> report_error_coordinates my_error_cell
```

```
*****  
Report           : Error Coordinates  
Error cell       : stub.err  
Date             : Mon Dec  3 16:39:36 2007  
Summary          : DRC Error Summary  
Distance unit    : micron  
*****
```

Data Names	Value

```

error type           Met1 Spacing
number of errors     5
bbox coordinates     {{413.355 378.240} {413.430 378.280}}
                     {{413.355 377.940} {413.430 377.960}}
                     {{197.555 158.240} {197.630 158.280}}
                     {{201.130 160.520} {201.205 160.560}}
                     {{203.535 93.440} {203.610 93.480}}

```

```

error type           Met1 specEoLSpc
number of errors     188816
bbox coordinates     {{2.340 1.430} {2.570 1.530}}
                     {{2.800 1.230} {2.900 1.440}}
                     {{2.915 1.550} {3.010 1.660}}
                     {{6.245 1.545} {6.410 1.640}}
                     {{6.410 1.360} {6.530 1.635}}
                     {{7.220 1.330} {7.510 1.430}}
                     {{6.815 1.140} {6.910 1.420}}
                     {{9.880 1.430} {10.110 1.530}}
                     {{10.340 1.230} {10.440 1.440}}

```

...

SEE ALSO

```

check_routeability(2)
verify_drc(2)
verify_lvs(2)
verify_pg_nets(2)

```

report_extraction_options

Reports the options that influence postroute extraction.

SYNTAX

```
status report_extraction_options
[-scenario scenario_list]
```

ARGUMENTS

```
-scenario scenario_list
    Reports extraction options given list of scenarios of a multiscenario design.
    Inactive scenarios will be skipped in the report. Each scenario is reported
    separately. If this option is not given, only the current scenario is
    reported.
```

DESCRIPTION

The **report_extraction_options** command reports the parameters that influence the post-route extraction.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example runs the **report_extraction_options** command.

```
prompt>set_extraction_options -max_cap_scale 1.1 -min_cap_scale 0.9 \
      -max_res_scale 1.2 -min_res_scale 0.8 \
      -max_ccap_scale 1.3 -min_ccap_scale 0.7 \
      -max_ccap_thres 0.01 -min_ccap_thres 0.005 \
      -max_ccap_ratio 0.1 -min_ccap_ratio 0.05 \
      -max_process_scale 1.4 -min_process_scale 0.6 \
      -no_obstruction \
      -max_segment_length 100 \
      -fan_out_thres 100
```

```
prompt> report_extraction_options
```

```
*****
Report : extraction options
Design : RISC_CORE
Version: A-2007.12-ICC
Date   : Wed Jul 11 14:20:05 2007
*****
```

```
Max capacitance scaling factor : 1.1
Min capacitance scaling factor : 0.9
```

Max resistance scaling factor : 1.2
Min resistance scaling factor : 0.8
Max coupling capacitance scaling factor : 1.3
Min coupling capacitance scaling factor : 0.7
Max coupling capacitance threshold : 0.01
Min coupling capacitance threshold : 0.005
Max coupling capacitance ratio : 0.1
Min coupling capacitance ratio : 0.05
Max process scaling factor : 1.4
Min process scaling factor : 0.6
No obstruction considered
Max segment length : 100
Fan out threshold : 100

1

SEE ALSO

`set_extraction_options(2)`

report_fast_mode

Reports the fast mode settings.

SYNTAX

report_fast_mode

ARGUMENTS

None

DESCRIPTION

Report current setting of the fast mode by the set_fast_mode command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

```
prompt> report_fast_mode
fast mode : On
no_congestion : false
```

SEE ALSO

set_fast_mode(2)

report_feasibility_options

Reports options set for feasibility using set_feasibility_options command.

SYNTAX

```
int report_feasibility_options  
[-scenarios {scenario_name1, scenario_name2, ...}]
```

DESCRIPTION

The **report_feasibility_options** command reports all the options set for feasibility optimization and reportion using the command **set_feasibility_options**. Use these options to constraint feasibility place_opt and reporting commands.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example shows a report:

```
prompt>set_feasibility_options -group_io false -zero_path_margin 50 -  
zero_wire_load_margin 100  
prompt>set_feasibility_options -design_slack_threshold -3  
prompt>set_feasibility_options -slack_threshold_path_group "Clk1 Clk2" -  
path_group_slack_threshold -2.5  
prompt>set_feasibility_options -  
slack_threshold_path_group "feasibility_io_path_group" -path_group_slack_threshold -  
3.5  
prompt>report_feasibility_options  
  
Feasibility Mode is enabled  
  
zp_violations           : true  
zwl_violations          : true  
group_io                : false  
reporting               : true  
zp_margin               : 50%  
zwl_margin              : 100%  
io_margin               : 0%  
design slack threshold   : -3  
path groups slack threshold : Clk1:-2.5 Clk2:-2.5 feasibility_io_path_group:-  
3.5  
verbose_level           : 1
```

SEE ALSO

place_opt(2)
set_feasibility_options(2)

```
get_adjusted_endpoints(2)
report_adjusted_endpoints(2)
```

report_filler_placement

Reports on the standard cell fillers contained in a design.

SYNTAX

```
int report_filler_placement  
[-abut]  
-lib_cell libcells
```

ARGUMENTS

-abut

Reports only the adjacent filler cells that form a consecutive pair in a cell row.

DESCRIPTION

The **report_filler_placement** command reports the types of fillers used in the design and their respective locations, based on specified list of filler cells.

If the **-abut** option is specified, the **report_filler_placement** command reports only the occurrence of two consecutive filler cells in the same cell row and their locations.

EXAMPLES

The following example reports the consecutive filler cells with the given reference library names:

```
prompt> report_filler_placement -lib_cells {OK0NA02M1 OK0BF00M3 OK0SPACER}  
-abut
```

The following are examples with and without the **-abut** option:

```
prompt> report_filler_placement -lib_cell FILL1BWP  
Information: xofiller!FILL1BWP!1 standard-  
cell filler is located at (311.920000, 285.040000). (APL-054)  
Information: xofiller!FILL1BWP!2 standard-  
cell filler is located at (312.060000, 285.040000). (APL-054)  
Information: xofiller!FILL1BWP!3 standard-  
cell filler is located at (15.400000, 358.120000). (APL-054)  
Information: xofiller!FILL1BWP!4 standard-  
cell filler is located at (15.540000, 358.120000). (APL-054)
```

```
prompt> report_filler_placement -lib_cell FILL1BWP -abut  
Warning: xofiller!FILL1BWP!1 filler abuts with xofiller!FILL1BWP!2 and they are  
located at (311.920000, 283.665000) and (312.060000, 283.665000). (APL-053)  
Warning: xofiller!FILL1BWP!3 filler abuts with xofiller!FILL1BWP!4 and they are  
located at (15.400000, 356.745000) and (15.540000, 356.745000). (APL-053)  
Warning: xofiller!FILL1BWP!4 filler abuts with xofiller!FILL1BWP!5 and they are  
located at (15.540000, 356.745000) and (15.680000, 356.745000). (APL-053)
```

SEE ALSO

`insert_stdcell_filler(2)`
`remove_stdcell_filler(2)`

report_flip_chip_bump_attributes

Reports the attributes set on the flip-chip bump cells for flip-chip routing.

SYNTAX

```
status report_flip_chip_bump_attributes  
-bump_cells collection_of_bump_cells  
[-output bool]
```

Data Types

collection_of_bump_cells collection

ARGUMENTS

```
-bump_cells collection_of_bump_cells  
           Specifies the names of the bump cells to report the attributes on them.  
  
-output bool  
           Outputs the report in a tcl script format.
```

DESCRIPTION

This command reports attributes set on the bump cells for the flip-chip router to honor.

EXAMPLES

```
The following example reports attributes on selected bump cells:  
prompt> report_flip_chip_bump_attributes -bump_cells [get_selection]  
Flip-chip bump pin target access edges:  
-----  
SNPS_BUMP_1      N NE E  
SNPS_BUMP_2      S  
1
```

SEE ALSO

set_flip_chip_bump_attributes(2)

report_flip_chip_driver_bump

Issues a report of flip chip drivers and bumps matching results set with assign_flip_chip_nets command.

SYNTAX

status_value **report_flip_chip_driver_bump**

ARGUMENTS

None.

DESCRIPTION

This command issues a report of the driver-bump matching results. The report is in table format. Each line represents a flip chip net connection. The first column specifies the bump cell; the second column specifies the net that connects the bump and the driver; and the third column specifies the driver cell. The driver-bump connections can be set with assign_flip_chip_nets command. The following is an example of the report:

```
/***** Bump / Driver Matching Net *****/
"BUMP_X_503" "REG_DATA_4[21]" "U__4_PAD_92"
"BUMP_X_502" "REG_DATA_1[10]" "U__1_PAD_81"
"BUMP_X_501" "REG_DATA_4[6]" "U__4_PAD_77"
"BUMP_X_500" "COL_ON_2" "U__2_PAD_15"
"BUMP_X_499" "REG_ADDR_4[9]" "U__4_PAD_68"
"BUMP_X_498" "REG_DATA_4[4]" "U__4_PAD_75"
"BUMP_X_497" "REG_ADDR_4[0]" "U__4_PAD_59"
"BUMP_X_496" "REG_DATA_2[29]" "U__2_PAD_100"
"BUMP_X_495" "W_FBI_ON_4" "U__4_PAD_12"
"BUMP_X_494" "REG_DATA_4[8]" "U__4_PAD_79"
"BUMP_X_493" "RE_RDY_4" "U__4_PAD_58"
"BUMP_X_492" "RE_BUSY_4" "U__4_PAD_8"
"BUMP_X_491" "REG_DATA_4[1]" "U__4_PAD_72"
"BUMP_X_490" "REG_DATA_4[16]" "U__4_PAD_87"
"BUMP_X_489" "REG_DATA_4[10]" "U__4_PAD_81"
```

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports the results of driver-bump matching and dumps the report to a file named fc_match using redirect operator >.

```
prompt> report_flip_chip_driver_bump > fc_match
```

SEE ALSO

`assign_flip_chip_nets(2)`

report_flip_chip_type

Report the personality type of specified nets, flip chip bumps or drivers

SYNTAX

```
status_value report_flip_chip_type  
net_or_cell_list
```

Data Types

net_or_cell_list collection

ARGUMENTS

net_or_cell_list
Specifies a collection of nets or cells whose personality type will be reported. The collection must contain nets, flip chip drivers or flip chip bumps.

DESCRIPTION

This command reports personality types of specified nets, bumps or drivers.

In a flip chip design, a bump is a piece of metal at the topmost layer of the chip that forms an electrical contact to the chip's packaging. A flip chip driver is an I/O circuitry that drives or receives signals from or to the chip thru the bump. A bump and its associated driver may be placed at different locations but need to be electrically connected.

A personality type is a string associated to a net or a flip chip driver or a flip chip bump. Only drivers and bumps of identical personality type can be electrically connected to form a complete I/O circuit. Drivers and bumps without a personality type setting will have a default personality type, which is an empty string. The personality type setting stored in the design and is Milkyway persistent. This command will list the personality type for each cell.

The following is an example of the report: /*****Flip Chip Bump/Driver/Net Personality Type Report*****/

```
"BUMP_X_66" "red" "BUMP_X_252" "green" "BUMP_X_144" "red" "BUMP_X_352" "red"  
"BUMP_X_287" "BUMP_X_87" "red" "BUMP_X_57" "red" ...
```

The first column is the bump cell name, the second column is its personality type, empty is personality type not specified.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports the personality type of all cells whose name start with "BUMP".

```
prompt> report_flip_chip_type [get_cells "BUMP*" -all]
```

SEE ALSO

```
report_flip_chip_type(2)  
assign_flip_chip_nets(2)
```

report_floorplan_data

Reports the floorplan data on one or more modules.

SYNTAX

```
status report_floorplan_data  
-module module_objects  
[-view view_name]  
[-quiet]
```

Data Types

<i>module_objects</i>	list
<i>view_name</i>	string

ARGUMENTS

-module *module_objects*
Specifies a list of module objects on which to report the floorplan data value.
Each element in the list must be a collection, in which valid objects are current Milkyway design, hierarchical cells, soft macros, hard macros and plan groups.

-view *view_name*
Specifies the view in which floorplan data will be reported.
Valid views are *one_level*, *physical* and *logical*.
The *one_level* view indicates data value is calculated by traversing only one level inside the module.
Both *physical* view and *logical* view indicate data value is calculated by traversing through all levels of the hierarchical tree in the module. The difference between *physical* view and *logical* view is that in *physical* view soft macros inside the module will be traversed down.
Plan groups only have *logical* view so if you specify a plan group in the option **-module**, this option will be ignored.
The default value of this option is *one_level*.

-quiet
Indicates that all error and warning messages are not to be reported.

DESCRIPTION

The **report_floorplan_data** command generates a report of floorplan data on the specified objects.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports floorplan data on cell I_ORCA_TOP in one_level view:

```
prompt> report_floorplan_data -module [get_cell I_ORCA_TOP] -view one_level
```

```
*****
Report : Floorplan Data
LIBRARY: lib_ORCA
DESIGN  : ORCA
CELVIEW : ORCA
Date    : Mon Dec 17 05:49:04 2007
*****
```

Object	Module Type	View	FP Data Name	Value
I_ORCA_TOP	child_cell	one_level	hard_macro_area_percentage	0.000000
I_ORCA_TOP	child_cell	one_level	macro_area_percentage	0.501060
I_ORCA_TOP	child_cell	one_level	number_of_black_box	0
I_ORCA_TOP	child_cell	one_level	number_of_hard_macro	0
I_ORCA_TOP	child_cell	one_level	number_of_io_cell	0
I_ORCA_TOP	child_cell	one_level	number_of_macro	3
I_ORCA_TOP	child_cell	one_level	number_of_standard_cell	127
I_ORCA_TOP	child_cell	one_level	physical_area	1045885.445100

The following example reports floorplan data on current Milkyway design in physical view:

```
prompt> report_floorplan_data -module [current_mw_cel] -view physical
```

```
*****
Report : Floorplan Data
LIBRARY: lib_ORCA
DESIGN  : ORCA
CELVIEW : ORCA
Date    : Mon Dec 17 05:50:38 2007
*****
```

Object	Module Type	View	FP Data Name	Value
ORCA	top	physical	hard_macro_area_percentage	0.151072
ORCA	top	physical	macro_area_percentage	0.535885
ORCA	top	physical	number_of_black_box	0
ORCA	top	physical	number_of_hard_macro	41
ORCA	top	physical	number_of_io_cell	924
ORCA	top	physical	number_of_macro	41
ORCA	top	physical	number_of_standard_cell	54092
ORCA	top	physical	physical_area	1690459.156600

SEE ALSO

`floorplan_data_attributes(3)`
`get_floorplan_data(2)`
`list_floorplan_data(2)`

report_fp_clock_plan_options

Reports options for the clock planning clock tree synthesis engine.

SYNTAX

```
status report_fp_clock_plan_options
```

ARGUMENTS

The **report_fp_clock_plan_options** command has no arguments.

DESCRIPTION

This command reports the options set by the **set_fp_clock_plan_options** command. If no clock plan options have been set, the command reports the default values. If you do not specify any options with this command, all settings are reported. The generated report can be redirected to an output file.

Multicorner-Multimode Support

This command is not supported in a multi-scenario design flow.

EXAMPLE

The following example reports the directory used for clock planning constraint, log, and report files.

```
prompt> report_fp_clock_plan_options -output_directory
```

The following example redirects the report to the cp_options.log file.

```
prompt> report_fp_clock_plan_options > cp_options.log
```

SEE ALSO

```
compile_fp_clock_plan(2)  
set_fp_clock_plan_options(2)
```

report_fp_macro_array

Report user defined macro array constraints.

SYNTAX

```
status report_fp_macro_array  
[-output_file file_name]  
[array_name]
```

Data Types

<i>file_name</i>	string
<i>array_name</i>	string

ARGUMENTS

-output_file *file_name*
Specifies name of the file to write command output to. The default is to output to xterm and gui log window.

array_name
This option specifies the name of the macro array to be reported. The default is to report all the macro array constraints.

DESCRIPTION

The **report_fp_macro_array** command reports user defined macro array constraints set by **set_fp_macro_array** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows how to report the array constraint named arr1.

```
prompt>report_fp_macro_array arr1
```

SEE ALSO

```
set_fp_macro_array(2)  
create_fp_placement(2)
```

report_fp_macro_options

Issues a report about the floorplan macro options.

SYNTAX

```
status report_fp_macro_options
[-output_file file_name]
[macro_cells]
```

Data Types

<i>file_name</i>	string
<i>macro_cells</i>	collection

ARGUMENTS

-output_file *file_name*
Specifies name of the file to write command output to. The default is to output to xterm and gui log window.

macro_cells
Reports floorplan macro options for the specified macros. The default is to report macro options for all macros.

DESCRIPTION

This command reports various options set by the **set_fp_macro_options** command to constrain macros during the automatic generation of the floorplan, using virtual flat placer in design planning.

For more information about these floorplan options, read the man page description of the **set_fp_macro_options** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows a sample report:

```
prompt> report_fp_macro_options
*****
Report macro options
Design: my_top_design
Version: Z-2007.03-ICC-SP3-INTERNAL
Date   : Mon Jun 18 14:08:55 2007
*****
Pin Align:
```

```
report_fp_macro_options
1794
```


Pin D[0] on cell A1 to align with port PD[0] of the design

Anchor Bound:

Anchor bound to 1 with x_offset=10.000 on cell(s):

A1

A2

A3

Info: 2 macro options reported.

1

SEE ALSO

set_fp_macro_options(2)

create_fp_placement(2)

report_fp_pin_constraints

Displays the pin assignment options that have been set for the specified soft macros, plan groups or ports.

SYNTAX

```
int report_fp_pin_constraints  
[blocks]  
[-block_level]
```

Data Types

blocks collection or list

ARGUMENTS

blocks

Specifies the soft macros and or plan groups for which the pin constraints are shown. If you do not specify this argument, the pin constraints are shown for all soft macros and plan groups in the current design.

-block_level

Reports the block-level pin assignment constraints on the currently open cell instead of the pin constraints for blocks contained in the cell.

DESCRIPTION

Displays the pin assignment options controlling both pin assignment and pin cutting.

Regarding side number, the left most edge is numbered 1 (one). Other edges are numbered according to their orders of traversal in the clockwise direction starting from edge 1. If there are multiple left most edges then edge 1 is the lowest leftmost edge.

Use **set_fp_pin_constraints** to set these pin constraints.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLE

The following example shows the pin constraints for all soft macros and plan groups in the current design.

```
prompt> report_fp_pin_constraints
```

```
report_fp_pin_constraints  
1796
```

SEE ALSO

`analyze_fp_routing(2)`

report_fp_placement

Generates a quality of results (QoR) report for virtual flat placement.

SYNTAX

```
int report_fp_placement  
[-check_abutment]  
[-verbose integer]
```

ARGUMENTS

-check_abutment
Select this option for channel-less design when you want to check for the abutment of each soft/hard macro with surrounding macro cell instances. The abutment check reports gaps detected between the macro cell instances. (No warning is issued if an edge of a soft/hard macro touches a cell boundary.)

-verbose *integer*
Enables you to define the amount of information to print. The default is 5.

DESCRIPTION

Generates a placement report to indicate the quality of placement. In this report the abbreviation "TL" means top level, the abbreviation "PG" means plan group. Note that in this context, plan group actually means any exclusive placement constraint so it includes voltage area constraints.

By default, this command will check for placement density and overlaps. When "-check_abutment" is specified, it will only check for the abutment of each soft and hard macro.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLE

The following example shows a basic output of the report_fp_placement command.

```
prompt> report_fp_placement
```

```
*****  
Report      : Virtual Flat Placement  
Design      : ORCA_routed  
Version     : B-2008.09-ICC-SP3  
Date        : Fri Nov 21 11:37:24 2008  
*****
```

```
report_fp_placement  
1798
```

Total wirelength: 35153.93
Number of 100x100 tracks cell density regions: 1
Number of low (< 10%) cell density regions: 0 (0.000%)
Number of high (> 200%) cell density regions: 0 (0.000%)
Maximum cell density: 71.37% (at 45 223 223 401)
Checking hard macro to hard macro overlaps...
Number of hard macro to hard macro overlaps: 0
Checking hard macro to std cell overlaps...
Number of hard macro to std cell overlaps: 0
Checking plan group to plan group overlaps...
Number of plan group to plan group overlaps: 0
Number of TL cells overlapping PG: 0
Number of cells violating core area: 0
Total number of cells violating plan group or core area: 0

SEE ALSO

report_fp_placement_strategy

Prints the current values and default values for the parameters controlled by **set_fp_placement_strategy**.

SYNTAX

```
status report_fp_placement_strategy
```

ARGUMENTS

The **report_fp_placement_strategy** command has no arguments.

DESCRIPTION

Prints the current values and default values for the parameters controlled by **set_fp_placement_strategy**. These parameters are used to control the virtual flat placer and legalizer (**create_fp_placement** and **legalize_fp_placement**). These parameters are not applicable to other ICC placement commands or in other parts of the flow.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLE

The following example shows a sample output of the **report_fp_placement_strategy** command.

```
prompt> report_fp_placement_strategy

*****
Report      : report_fp_placement_strategy
Design CEL  : top.CEL;1
Date       : Wed Oct 18 17:29:45 2006
*****
*** Macro related parameters ***
set_fp_placement_strategy -macro_orientation automatic | all | N
    current setting: N
    default setting: automatic
set_fp_placement_strategy -auto_grouping none | user_only | low | high
    current setting: low
    default setting: low
set_fp_placement_strategy -array_macros_only on | off
    current setting: off
    default setting: off
set_fp_placement_strategy -macros_on_edge on | off
    current setting: off
    default setting: off
set_fp_placement_strategy -sliver_size <distance>

report_fp_placement_strategy
1800
```

```

    current setting: 0.00
    default setting: 0.00
set_fp_placement_strategy -snap_macros_to_user_grid on | off
    current setting: off
    default setting: off
set_fp_placement_strategy -fix_macros none | soft_macros_only | all
    current setting: none
    default setting: none

*** Congestion driven placement related parameters ***
set_fp_placement_strategy -congestion_effort low | medium | high
    current setting: low
    default setting: low

*** Net weighting related parameters ***
set_fp_placement_strategy -IO_net_weight <float>
    current setting: 1.00
    default setting: 1.00
set_fp_placement_strategy -plan_group_interface_net_weight <float>
    current setting: 1.00
    default setting: 1.00
set_fp_placement_strategy -voltage_area_interface_net_weight <float>
    current setting: 1.00
    default setting: 1.00
set_fp_placement_strategy -voltage_area_net_weight_LS_only on | off
    current setting: off
    default setting: off

*** Miscellaneous parameters ***
set_fp_placement_strategy -name legalizer_effort low | high
    current setting: high
    default setting: high
set_fp_placement_strategy -spread_spare_cells on | off
    current setting: on
    default setting: on
set_fp_placement_strategy -virtual_IPO on | off
    current setting: off
    default setting: off
1
prompt>

```

SEE ALSO

```

set_fp_placement_strategy(2)
create_fp_placement(2)

```

report_fp_rail_constraints

Reports power network synthesis (PNS) constraints defined by the user or by default.

SYNTAX

```
status report_fp_rail_constraints  
[-all]  
[-layer layer]  
[-ring]  
[-global]  
[-block_ring block]
```

Data Types

<i>layer</i>	string
<i>block</i>	string

ARGUMENTS

- all
Report all the PNS constraints specified by the user or by default. This option is on by default.
- layer *layer*
Report the PNS layer constraints on the specified metal layer. The keyword "all" can be used as layer name to refers to all the metal layers. By default this option is off.
- ring
Report the core ring constraints specified by the user or by default. By default, this option is off.
- global
Report the global constraints specified by the users or by default. By default, this option is off.
- block_ring *block*
Report the block ring constraints on the specified block. The keyword "all" can be used as block name to refer to all blocks in the design. By default, this option is off.

DESCRIPTION

This command reports the power network synthesis (PNS) constraints for power planning including layer constraints, core ring constraints, block ring constraints and global constraints.

Multicorner-Multimode Support

This command is not supported in a multi-scenario design flow.

EXAMPLES

The following example shows how to report the constraints of the METAL5 layer.

```
prompt> report_fp_rail_constraints -layer METAL5
```

SEE ALSO

```
set_fp_rail_constraints(2)  
set_fp_block_ring_constraints(2)
```

report_fp_rail_strategy

Reports the user-defined strategy in power network synthesis (PNS) and power network analysis (PNA) strategies.

SYNTAX

```
status report_fp_rail_strategy
[-all]
[-use_lm_view]
[-use_tluplus]
[-pna_ultra_solver]
[-virtual_pad_wire_width]
[-define_pad_connection]
[-pad_resistance_file]
[-pns_skip_ir]
[-honor_macro_strap_config]
[-pns_hor_relative_offset]
[-pns_ver_relative_offset]
[-pns_ignore_via_cut_to_edge]
[-pns_ignore_soft_macro_blockage]
[-pns_clip_top_boundaries]
[-cut_plangroup_edge_layers]
[-pna_via_cut_row_column]
[-honor_macro_route_constraints]
[-align_strap_with_top_pin true]
[-align_strap_with_mtcmos_cells]
[-align_strap_with_bump_cells]
[-align_strap_with_m1_rail]
[-put_strap_in_std_cell_row]
```

ARGUMENTS

-use_lm_view

Reports strategy value for the LM view set by using the **-refdb_lm** option of the **set_fp_rail_strategy** command. Please note that LM views are optional aspects of a Milkyway database. All ICC related Milkyway libraries may not have LM views present.

-use_tluplus

Reports the strategy value for the TLU+ model set by using the **-use_tluplus** option of the **set_fp_rail_strategy** command.

-pna_ultra_solver

Reports the strategy value for the ultra-solver set by using the **-pna_ultra_solver** option of the **set_fp_rail_strategy** command.

-virtual_pad_wire_width

Reports the strategy value for the virtual wire width set by using the **-virtual_pad_wire_width** option of the **set_fp_rail_strategy** command.

-pad_resistance_file

Reports the strategy value for the pad resistance file set by using the **-**

report_fp_rail_strategy

pad_resistance_file option of the **set_fp_rail_strategy** command.

-pns_skip_ir

Reports strategy value for the IR drop simulation set by using the **-pns_skip_ir** option of the **set_fp_rail_strategy** command.

-honor_macro_strap_config

Reports the strategy value for the configuration file set by using the **-honor_macro_strap_config** option of the **set_fp_rail_strategy** command.

-pns_hor_relative_offset

Reports the strategy value for the relative offsets from the top-most horizontal layer to the specified layer set by using the **-pns_hor_relative_offset** option of the **set_fp_rail_strategy** command.

-pns_ver_relative_offset

Reports the strategy value for the relative offsets from the top-most vertical layer to the specified layer set by using the **-pns_ver_relative_offset** option of the **set_fp_rail_strategy** command.

-pns_ignore_via_cut_to_edge

Reports the strategy value set by using the **-pns_ignore_via_cut_to_edge** option of the **set_fp_rail_strategy** command.

-pns_ignore_soft_macro_blockage

Reports the strategy value for soft macro blockages set by using the **-pns_ignore_soft_macro_blockage** option of the **set_fp_rail_strategy** command.

-pns_clip_top_boundaries

Reports the strategy value for clipping at top-cell boundaries set by using the **-pns_clip_top_boundaries** option of the **set_fp_rail_strategy** command.

-cut_plangroup_edge_layers

Reports the strategy value for the synthesized strap overlap with plangroup edges set by using the **-cut_plangroup_edge_layers** option of the **set_fp_rail_strategy** command.

-pna_via_cut_row_column

Reports the strategy value for breaking a large square or a rectangular via array into smaller square via arrays based on the value set by using the **-pna_via_cut_row_column** option of the **set_fp_rail_strategy** command.

-honor_macro_route_constraints

Reports the strategy value for routing constraints for macros in an input file set by using the **-honor_macro_route_constraints** option of the **set_fp_rail_strategy** command.

-align_strap_with_top_pin true

Reports the strategy value for PNS aligning synthesized P/G straps with P/G pins on block boundaries set by using the **-align_strap_with_top_pin** option of the **set_fp_rail_strategy** command.

-align_strap_with_mtcmos_cells

Reports the strategy value for PNS generating P/G straps to align MTCMOS cells and connect the straps to corresponding pins in the MTCMOS cells based set

by using the **-align_strap_with_mtcmos_cells** option of the **set_fp_rail_strategy** command.

-align_strap_with_bump_cells

Reports the strategy value for specifying the direction and metal layer of straps to be aligned with bump cells in flip-chip design set by using the **-align_strap_with_bump_cells** option of the **set_fp_rail_strategy** command.

-align_strap_with_m1_rail

Reports the strategy value for PNS generating the lowest horizontal P/G straps to align with standard cell rails set by using the **-align_strap_with_m1_rail** option of the **set_fp_rail_strategy** command.

-put_strap_in_std_cell_row

Reports the strategy value for PNS generating the lowest horizontal P/G straps inside standard cell rows and between standard cell rails set by using the **-put_strap_in_std_cell_row** option of the **set_fp_rail_strategy** command.

DESCRIPTION

This command prints the current values of the strategies set by the **set_fp_rail_strategy** command. These strategies are used for power network synthesis and analysis.

Multicorner-Multimode Support

This command is not supported in a multi-scenario design flow.

EXAMPLES

The following example prints the current values of **-use_lm_view**.

```
prompt> report_fp_rail_strategy -use_lm_view\  
-use_lm_view = true
```

SEE ALSO

set_fp_rail_strategy(2)

report_fp_rail_voltage_area_constraints

Reports multi-voltage power network synthesis (MVDD PNS) constraints defined by the user or by default.

SYNTAX

```
status report_fp_rail_voltage_area_constraints  
-all  
-voltage_area voltage_area  
[-synthesis]  
[-layer layer]  
[-ring]  
[-global]
```

Data Types

<i>voltage_area</i>	string
<i>layer</i>	string

ARGUMENTS

-all
Reports all voltage-area PNS constraints in all voltage areas. The **-all** and **-voltage_area** options are mutually exclusive; use only one.

-voltage_area *voltage_area*
Reports all voltage-area PNS constraints in the specified voltage area. The **-all** and **-voltage_area** options are mutually exclusive; use only one.

-synthesis
Reports synthesis PNS constraints in the voltage area. This includes net name(s), supply voltage, power budget, target voltage drop, and the number of power straps extended out from the voltage area.

-layer *layer*
Reports layer PNS constraints in the voltage area. You can substitute the layer name you want for *layer* or use the keyword **all** as a *layer* value to refer to all the metal layers.

-ring
Reports voltage-area ring PNS constraints in the voltage area.

-global
Reports global PNS constraints in the voltage area.

DESCRIPTION

This command reports the multi-voltage power network synthesis (MVDD PNS) constraints for power planning, including voltage area synthesis constraints, layer constraints, ring constraints, and global constraints.

Multicorner-Multimode Support

This command is not supported in a multi-scenario design flow.

EXAMPLES

The following example shows how to report all PNS constraints in the *MULTIPLIER* voltage area.

```
prompt> report_fp_rail_voltage_area_constraints\  
-voltage_area MULTIPLIER
```

SEE ALSO

```
set_fp_rail_voltage_area_constraints(2)
```

report_fp_relative_location

Reports macro relative location constraints.

SYNTAX

```
status report_fp_relative_location  
[-output_file file_name]  
[cells]
```

Data Types

<i>file_name</i>	string
<i>cells</i>	collection

ARGUMENTS

-output_file *file_name*
Specifies the name of the file to output the macro relative location constraints. The default is to output to the xterm and the IC Compiler GUI log windows.

cells
Specifies a collection of macro cells. The tool reports the relative location constraints of the macro cells in this collection only. The default is to report relative location constraints on all macro cells.

DESCRIPTION

The **report_fp_relative_location** command report the macro relative location constraints set by the **set_fp_relative_location** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows how to report the relative location constraint on cell iRAM1.

```
prompt>report_fp_relative_location \  
[get_cells "iRAM1"]
```

SEE ALSO

```
set_fp_relative_location(2)  
remove_fp_relative_location(2)  
create_fp_placement(2)
```

report_fp_shaping_strategy

Prints the current values and default values of the parameters controlled by the **set_fp_shaping_strategy** command.

SYNTAX

status **report_fp_shaping_strategy**

ARGUMENTS

The **report_fp_shaping_strategy** command has no arguments.

DESCRIPTION

Prints the current values and default values for the parameters controlled by the **set_fp_shaping_strategy** command. These parameters are used to control the **shape_fp_blocks** command. These parameters are not applicable to other IC Compiler commands.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLE

The following example shows a sample output of the **report_fp_shaping_strategy** command.

```
prompt> report_fp_shaping_strategy

*****
Report      : report_fp_shaping_strategy
Design CEL  : none found
Version     : C-2009.06-IC Compiler
Date        : Thu May 14 15:02:53 2009
*****
set_fp_shaping_strategy -avoid_power_grid on | off
  current setting: off
  default setting: off
set_fp_shaping_strategy -distance_to_power_grid <float>
  current setting: -1.000
  default setting: -1.000
set_fp_shaping_strategy -keep_top_level_together on | off
  current setting: on
  default setting: off
set_fp_shaping_strategy -min_channel_size <float>
  current setting: 15.000
  default setting: 0.000
set_fp_shaping_strategy -util_slack <float>
  current setting: 0.150
```

```
report_fp_shaping_strategy
1810
```



```
    default setting: 0.100
set_fp_shaping_strategy -add_channel_blockages none | partial | soft | hard
    current setting: soft
    default setting: soft
set_fp_shaping_strategy -adjust_macro_locations on | off
    current setting: on
    default setting: off
set_fp_shaping_strategy -preserve_abutment on | off
    current setting: off
    default setting: off
set_fp_shaping_strategy -max_shape_complexity <integer>
    current setting: 0
    default setting: 0
```

SEE ALSO

```
set_fp_shaping_strategy(2)
shape_fp_blocks(2)
```

report_fp_voltage_area_constraints

Displays the feedthrough options that have been set for the specified voltage areas.

SYNTAX

```
int report_fp_voltage_area_constraints
```

ARGUMENTS

The **report_fp_voltage_area_constraints** command has no arguments.

DESCRIPTION

Displays the options controlling feedthrough creation on voltage areas.

Use **set_fp_voltage_area_constraints** to set these feedthrough constraints.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLE

The following example shows the feedthrough constraints for all voltage areas in the current design.

```
prompt> report_fp_voltage_area_constraints
```

SEE ALSO

```
analyze_fp_routing(2)  
remove_fp_voltage_area_constraints(2)  
route_global(2)  
set_fp_voltage_area_constraints(2)
```

report_groute_options

Reports global router cell-persistent options, their current value, default value, range, type and a brief description.

SYNTAX

```
status report_groute_options  
[-name name]
```

Data Types

name string

ARGUMENTS

`-name name`
 Specifies the name or name pattern of the option to be reported. Name pattern will report all options that match it. By default, if "-name" is not specified, all groute options are reported.

DESCRIPTION

The **report_groute_options** command shows options that tune global router operation. These settings are persistent in the Milkyway design. Cell-persistent options are saved in cell data base by **set_groute_options** command. Before printing options, **report_groute_options** loads all groute options from currently opened cell. All options are optional. This command prints all available global router options, their current value, default value, range, type and a brief description. However, if you want to see a specific option or options that match a specified name pattern, use argument "-name". You can set global router options with command **set_groute_options**.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

```
prompt> report_groute_options -name accessPolyPin  
  
(Integer Option for "groute" "accessPolyPin" 1)  
;;        range [0,1], default=1, stored in cell;  
;;        0: don't access poly pins,  
;;        1: access poly pins.
```

SEE ALSO

set_groute_options(2)

report_hierarchy

Displays the reference hierarchy of the current instance or the current design.

SYNTAX

```
integer report_hierarchy  
[-nosplit]  
[-full]  
[-noleaf]
```

ARGUMENTS

-nosplit

Prevents line-splitting and facilitates writing software to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

-full

Displays the full hierarchy. By default, components of submodules in multiple locations in a hierarchy are listed only once. An ellipsis (...) indicates the contents of a previously-displayed module.

-noleaf

Indicates that the leaf library cells are to be excluded from the reference hierarchy report.

DESCRIPTION

This command displays the indented reference hierarchy of the current instance or the current design. If the current instance has been set, the report is generated for the design of that instance. Otherwise, the report is generated for the current design.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following is an example of a hierarchy report:

```
prompt> report_hierarchy  
  
*****  
Report : hierarchy  
Design : CONTROL  
Version: v2.0
```

Date : Fri Mar 20 14:09:24 1991

```
CONTROL
  CTLX
    AO1      tech_lib
    AO2      tech_lib
    AO3      tech_lib
    AO4      tech_lib
    AO6      tech_lib
    AO7      tech_lib
    EON1     tech_lib
    INVA     tech_lib
    NAND2    tech_lib
    NAND3    tech_lib
    NAND4    tech_lib
    NR2      tech_lib
    NR3      tech_lib
    OR2      tech_lib
    OR3      tech_lib
  INV       tech_lib
  JPMP
  MX1P      tech_lib
  NAND2     tech_lib
  NAND3     tech_lib
```

SEE ALSO

```
group(2)
report_cell(2)
report_design(2)
report_lib(2)
report_reference(2)
ungroup(2)
```

report_host_options

Prints a report of hosts, pools, and their options as defined by the *set_host_options* command.

SYNTAX

```
int report_host_options
```

DESCRIPTION

The **report_host_options** command is used to print a report of the hosts, pools, and their options defined by the *set_host_options* command.

SEE ALSO

```
set_host_options(2)  
remove_host_options(2)  
set_mcmm_job_options(2)
```

report_ideal_network

Displays information about ports, pins, nets, and cells on ideal networks in the current design.

SYNTAX

```
int report_ideal_network  
[-net]  
[-cell]  
[-load_pin]  
[-timing]  
[object_list]
```

Data Types

object_list list

ARGUMENTS

-net

Causes **report_ideal_network** to list all nets in the specified ideal network(s). By default, nets are displayed for all ideal networks.

-cell

Indicates to display all cells in the specified ideal network(s). By default, cells are displayed for all ideal networks.

-load_pin

Specifies to display load pins (boundary pins) of the specified ideal network(s), along with any ideal timing set on them using the **set_ideal_latency** and **set_ideal_transition** commands. By default, load pins are displayed for all ideal networks.

-timing

Specifies to display all internal pins (for example, non-source and non-boundary pins) in the specified ideal network(s) that have ideal timing set on them using the **set_ideal_latency** and **set_ideal_transition** commands. By default, internal pins with timing are displayed for all ideal networks.

object_list

Defines a list of source ports or source pins of the specified ideal network(s) to be displayed. By default, the source pins and source ports for all ideal networks in the current design are displayed.

DESCRIPTION

Displays information about the ideal networks in the current design. If no arguments are specified, all ideal network sources in the current design are displayed. If you specify a list of source pins or ports, the command displays information about the ideal networks at these objects. By default, source pins, internal pins with ideal timing, load (boundary) pins, nets, and cells are displayed. You can force a subset

of this information to be displayed using the arguments listed above.

The "Latency" and "Transition" columns show the minimum and maximum values set for both rising and falling edges. You set these values with the **set_ideal_latency** and **set_ideal_transition** commands.

Ideal networks are an extension of ideal nets that incorporate automatic propagation of the **ideal** attribute. You specify only the source ports or pins of the network; all the nets, cells, and pins on the transitive fanin of these objects are treated as ideal by **compile**. The **ideal_network** attribute is automatically spread by Design Compiler, and respread as needed during **compile** optimizations.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows an ideal network report.

```
prompt> set_ideal_network {reset1 reset2}
prompt> set_ideal_latency 5.2 reset1
prompt> set_ideal_transition 0.6 reset1
prompt> report_ideal_network
```

```
*****
Report : ideal_network
Design : TEST
Version: 2001.08
Date   : Mon May  7 10:36:21 2001
*****
```

Source ports and pins	Latency				Transition			
	Rise		Fall		Rise		Fall	
	min	max	min	max	min	max	min	max
top/reset1	5.20	5.20	5.20	5.20	0.60	0.60	0.60	0.60
top/reset2	--	--	--	--	--	--	--	--

The following example shows an ideal network report for the network starting at the pin named *source1*.

```
prompt> report_ideal_network source1
*****
```

```
Report : ideal_network
Design : idn_test
Version: 2001.08
Date   : Wed May  9 17:09:02 2001
*****
```

Source ports and pins	Latency				Transition			
	Rise		Fall		Rise		Fall	
	min	max	min	max	min	max	min	max

```
idn_test/source1    0.10  0.10      0.10  0.10      0.10  0.10      0.10  0.10
```

Boundary pins	Latency				Transition			
	Rise		Fall		Rise		Fall	
	min	max	min	max	min	max	min	max
ff/TE	--	--	--	--	--	--	--	--
U1/B	--	--	--	--	--	--	--	--
ff/CD	--	--						

Nets

```
-----
a
data
rst1
n7
n8
```

Cells

```
-----
C11
U9
U3
U10
```

The following example reports the cells and nets on the ideal network sourced at pins named *source1* and *source2*.

```
prompt> report_ideal_network -cell -net {source1 source2}
```

```
*****
```

```
Report : ideal_network
```

```
Design : idn_test
```

```
Version: 2001.08
```

```
Date   : Wed May  9 17:09:02 2001
```

```
*****
```

Source ports and pins	Latency				Transition			
	Rise		Fall		Rise		Fall	
	min	max	min	max	min	max	min	max
idn_test/source1	7.10	7.50	6.20	6.20	0.20	0.30	0.20	0.40
idn_test/source2	--	--	--	--	--	--	--	--

Nets

```
-----
source1
source2
data
rst1
n7
```

Cells

```
-----
C11
U9
```

U3
U8
C15
U11

SEE ALSO

`current_design(2)`
`remove_ideal_network(2)`
`report_cell(2)`
`report_constraint(2)`
`report_design(2)`
`report_net(2)`
`set_ideal_network(2)`
`set_ideal_latency(2)`
`set_ideal_transition(2)`

report_ignored_layers

Reports the routing layers that are ignored during congestion analysis and RC estimation. If you have set the minimum and maximum routing layers for the design, they are also reported.

SYNTAX

```
status report_ignored_layers
```

ARGUMENTS

The **report_ignored_layers** command has no arguments.

DESCRIPTION

During congestion analysis and RC estimation, the tool can ignore some routing layers. Use this command to report the ignored layers. If you have specified the minimum and maximum routing layers for the design, these are also reported.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows the **report_ignored_layers** command:

```
prompt> report_ignored_layers
```

SEE ALSO

```
remove_ignored_layers(2)  
report_lib(2)  
set_ignored_layers(2)
```

report_ilm

Reports information about the specified ILM instance.

SYNTAX

```
status report_ilm
[ilm_list]
```

Data Types

```
ilm_list      list
```

ARGUMENTS

```
ilm_list
    Specify the ILM instances to be reported. The option value can be a collection
    of ILM instances or name patterns.
    The command fails if it cannot find any specified ILM instance.
    If this option is not specified, the command reports all the ILM instances
    in the current design.
```

DESCRIPTION

The `report_ilm` command enables you to get detailed information about existing ILMs. If you specify the ILM, only those ILMs are reported; otherwise all ILMs in the current design are reported.

The information reported for each ILM includes: ILM reference name, ILM instance name, filepath of the library from which the ILM was loaded, date and time of ILM creation, options used for ILM creation, cell count and compression statistics for each of the ILM, cell count and compression statistics for the top design with ILMs, tlu plus settings, parasitics data of the ILM, UPF power domains present in the ILM.

This command returns a 1 if successful and returns a 0 otherwise.

EXAMPLES

The following example reports about all ILMs in the current design.

```
prompt> report_ilm
```

```
*****
Report   : ILM
Design   : top
Version  : B-2008.09-ICC-SP5
Date     : Mon Apr  6 02:28:42 2009
```

Top-level details
#####

ILM Instance	Reference	Orient	Location
I_BLK1	BLK1	0	(50.00, 52.52)

Top design leaf cell count statistics:

ILM instance count : 1
Top only cell count : 50000
Top + ILM cell count : 54000
Top + Block cell count : 70000
Compression percentage : 22.85

Reference ILM : BLK1
#####

Library name: /remote/disk1/design1/design_1

Date & Time of ILM creation(mm/dd/yyyy hr:min:sec): 4/6/2009 2:28:36

Options used in ILM creation:

Option	Value used*
verbose	on
identify_only	off
extract_only	off
no_auto_ignore	on
keep_macros	off
keep_boundary_cells	off
keep_full_clock_tree	off
keep_parasitics	on
include_xtalk	off
latch_level	not specified
include_side_load	boundary
ignore_ports	not specified
compact	all
traverse_disabled_arcs	off
case_controlled_ports	not specified
must_connect_ports	not specified

* Some of the options may have been automatically set/reset by the tool

Leaf cell count statistics:

ILM cell count : 4000
Block cell count : 20000

Compression percentage : 80.00

TLU+ files used:

Scenario #0: s1

Min TLU+ file : /remote/disk1/design1/tlu/1.tlu_0606
Max TLU+ file : /remote/disk1/design1/tlu/2.tlu_0606
Emulation Min TLU+ file : /remote/disk1/design1/tlu/3.tlu_0606
Emulation Max TLU+ file : /remote/disk1/design1/tlu/4.tlu_0606
Tech2ITF mapping file : /remote/disk1/design1/tlu/t2itf.map

Scenario #1: s2

Min TLU+ file : /remote/disk1/design1/tlu/5.tlu_0606
Max TLU+ file : /remote/disk1/design1/tlu/6.tlu_0606
Tech2ITF mapping file : /remote/disk1/design1/tlu/t2itf.map

Scenario #2: default

Min TLU+ file : /remote/disk1/design1/tlu/1.tlu_0606
Max TLU+ file : /remote/disk1/design1/tlu/2.tlu_0606
Emulation Min TLU+ file : /remote/disk1/design1/tlu/3.tlu_0606
Emulation Max TLU+ file : /remote/disk1/design1/tlu/4.tlu_0606
Tech2ITF mapping file : /remote/disk1/design1/tlu/t2itf.map

Parasitics data present in the ILM:

RC data : Available
CC data : Not available

TLU+ file used	Temperature(s)
/remote/disk1/design1/tlu/1.tlu_0606	125.00
/remote/disk1/design1/tlu/2.tlu_0606	125.00
/remote/disk1/design1/tlu/3.tlu_0606	125.00
/remote/disk1/design1/tlu/4.tlu_0606	125.00
/remote/disk1/design1/tlu/5.tlu_0606	100.00
/remote/disk1/design1/tlu/6.tlu_0606	100.00

SI aggressor data : Not available

UPF power domain present in the ILM:

Power Domain : I_BLK1/PD_BLK
Current Scope : top
Elements : I_BLK1
Available Supply Nets : SN_A_1, SN_A_2, SN_A_3, SN_A_4, SN_A_5, SN_A_6

Connections	-- Power --	-- Ground --
Primary: (Max Op. Voltage)	I_BLK1/VDD (1.88)	I_BLK1/VSS (0.00)
Isolation: ISO_STR_1	I_BLK1/ISO_VDD (1.88)	I_BLK1/ISO_VSS (0.00)
Retention: RET_STR_1	I_BLK1/R_VDD (1.68)	I_BLK1/R_VSS (0.00)
Retention: RET_STR_2	I_BLK1/R_VDD_1 (1.55)	I_BLK1/R_VSS_1 (0.00)

Switches	-- Input --	-- Output --
Switch: I_BLK1/SW1	I_BLK1/SW_NET_1	I_BLK1/SW_NET_2

Switch: I_BLK1/SW2

I_BLK1/SW_NET_3

I_BLK1/SW_NET_4

1

SEE ALSO

`create_ilm(2)`

`get_ilms(2)`

report_interclock_relation

Displays common multiple clock period between clocks of different periods.

SYNTAX

```
status report_interclock_relation  
[-from clock_name]  
[-to clock_name]  
[-nosplit]  
[-significant_digits digits]
```

Data Types

clock_name string

ARGUMENTS

-from *clock_name*
 Restricts reporting to only transfers that are launched by these clocks.

-to *clock_name*
 Restricts reporting to only transfers that are captured by these clocks.

-nosplit
 Prevents line-splitting and facilitates writing software to extract information from the report output.

-significant_digits *digits*
 Specifies the number of digits to the right of the decimal point to report. Allowed values are 0 through 13. The default value is 2. Using this option overrides the value set by the **report_default_significant_digits** variable.

DESCRIPTION

This command displays details of the calculations used to determine the timing for transfers between registers that are driven by different clocks. In cases where the periods are not the same, the least common multiple of these clock periods becomes the common clock period by adding cycles until a whole number of each clock's period has been reached, or internal limits on the expansion have been reached.

Each line of this report indicates a pair of clocks, one that launches a timing path and one that captures the data at the end of the path. It displays the period of each clock and the common multiple of each period that is used in the timing analysis. If no common period was reached between those two clocks, a dash appears in the Common column.

The command also displays the factor for each clock that was used in determining the common clock period. In each case the factor will be an integer under the column Count1 and Count2. The product of the factor and that clock's period is the common clock period if one was found.

If the **-from** option is specified, the output will be restricted to connections where this clock serves as the launching clock. If the **-to** option is used, the output will be restricted to clock pairs where this clock serves as a capturing clock. If both **-to** and **-from** are specified, there will be a single line if there is such a path.

If there are no connections that satisfy the report, an informational message will be issued.

EXAMPLES

The following example shows the output when there are two clocks; ck1 has a period of 10 and ck2 has a period of 12.

```
prompt> report_interclock_relation
```

```
*****
```

```
Report : Interclock Relation
```

```
  A common period of '-' means a full cycle was not expanded.
```

```
*****
```

From	Period1	Count1	To	Period2	Count2	Common
ck1	10.00	6	ck2	12.00	5	60.00
ck2	12.00	5	ck1	10.00	6	60.00

SEE ALSO

```
report_clock(2)
```

report_internal_loads

Displays internal loads on the nets in the current design.

SYNTAX

```
int report_internal_loads
[-nosplit]
```

ARGUMENTS

-nosplit

Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column. This option prevents line-splitting and facilitates writing software to extract information from the report output.

DESCRIPTION

Displays all internal loads specified on nets with the **set_load** command.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following is an example of the internal-loads report:

```
prompt> report_internal_loads

*****
Report : internal_loads
Design : comb_internal
Version: v2.0
Date   : Fri Nov 30 12:44:23 1990
*****

Attributes:
  p - includes pin load
```

Net	Load	Attributes
n1	3.0000	p
n2	4.5000	

SEE ALSO

`report_design(2)`
`set_load(2)`

report_io_antenna_properties

Reports external antenna properties.

SYNTAX

```
status report_io_antenna_properties  
[-output file_name]
```

Data Types

file_name string

ARGUMENTS

```
-output file_name  
      Specifies the name of the output file.
```

DESCRIPTION

This command reports all the external antenna properties of all ports in the current design to the specified file or log file.

EXAMPLES

```
prompt> report_io_antenna_properties  
  
*****  
Report : io_antenna_properties  
Design : test2_fixed  
Version: C-2009.06-ICC-SP1  
Date   : Sun May 31 23:52:27 2009  
*****  
  
define_io_gate_size -pin_name pad_in[3] -values {1.2 1.7 2.1}  
define_io_diode_protection -pin_name pad_in[3] -values {1.1 1.4 3.3 4 4.2}  
define_io_antenna_area -pin_name pad_in[3] -  
values {{METAL 100.5 false} {METAL 200 true}}
```

SEE ALSO

```
define_io_gate_size(2)  
define_io_diode_protection(2)  
define_io_antenna_area(2)  
remove_io_antenna_properties(2)  
extract_hier_antenna_property(2)
```

report_io_antenna_properties
1830

```
verify_zrt_route(2)
report_antenna_ratio(2)
```

report_io_constraints

Displays a report of the pin and pad physical constraints.

SYNTAX

```
status report_io_constraints  
[-cell name]  
[-pin_only]  
[-pad_only]  
[-chiplinelevel_pad_only]  
[objects]
```

Data Types

<i>name</i>	string
<i>objects</i>	collection

ARGUMENTS

- cell *name*
Specifies the Milkyway cell name or plan group whose physical constraints are reported. The default behavior is to report the current cell's physical constraints.
- pin_only
Reports only pin physical constraints. The **-pin_only**, **-pad_only** and **chiplinelevel_pad_only** options are mutually exclusive. Use only one.
- pad_only
Reports only pad physical constraints. The **-pin_only**, **-pad_only** and **chiplinelevel_pad_only** options are mutually exclusive. Use only one.
- chiplinelevel_pad_only
Reports only chip-level pad physical constraints. The **-pin_only**, **-pad_only** and **chiplinelevel_pad_only** options are mutually exclusive. Use only one.
- objects*
Specifies the pin or pad for which the physical constraints are displayed. If this option is not specified, the physical constraints for all pins, pads and chip level pads are reported.

DESCRIPTION

This command allows you to display a report of the pin and pad physical constraints that were set by using the **set_pin_physical_constraints**, **set_pad_physical_constraints**, **set_chiplinelevel_pad_physical_constraints** or **read_io_constraints** commands.

If no option is specified, the command displays a report of all the physical constraints for all pins, pads and chip level pads.

For the layer constraint, the **report_io_constraints** command displays the layer mask name.

EXAMPLES

The following example displays the physical constraints for all pins, pads and chip level pads in the current design.

```
prompt> report_io_constraints
```

The following example displays the physical constraints for all pins.

```
prompt> report_io_constraints -pin_only
```

SEE ALSO

report_isolate_ports

Displays the status of port isolation on ports on which isolation was requested.

SYNTAX

```
int report_isolate_ports  
[-nosplit]
```

ARGUMENTS

-nosplit

Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line in the same column. This option prevents line-splitting and facilitates writing software to extract information from the report output.

DESCRIPTION

The **report_isolate_ports** command displays all the ports on which isolation was requested and on which isolation cells were inserted.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following is an example of the report that is generated when you execute the **report_isolate_ports** command:

```
prompt> report_isolate_ports
```

```
report_isolate_ports
```

```
***** Report : isolate_ports Design : pil  
Version: 2001.08 Date : Mon Apr 23 15:05:21 2001  
*****
```

```
===== out  
Port Name Cell Name Inst.Name Type Forced Insertion  
===== 1  
- - buffer no qout IVDA U3 buffer no
```

SEE ALSO

```
remove_isolate_ports(2)  
set_isolate_ports(2)
```

```
report_isolate_ports  
1834
```


report_isolated_via

Reports isolated via violations.

SYNTAX

```
integer report_isolated_via  
[-isolated_via_spacing distance]  
[-isolated_via_quadrant_spacing distance]
```

Data Types

distance distance

ARGUMENTS

-isolated_via_spacing *distance*

Determines the adjacent via spacing, in microns, that triggers isolated via violation. If there is no adjacent via located less than *n* microns from the via, the violation of type "Isolated Via" is reported. If set, this argument will be stored in a "droute" cell parameter "isolatedViaSpacing". The range is 0.0 to 20.0. The default is the value, set in a "droute" cell parameter "isolatedViaSpacing", which in turn has a default of 1.0.

To set a distance in a "droute" cell parameter "isolatedViaSpacing", use
set_parameter -type real -module droute \
 -name isolatedViaSpacing -value *n*

-isolated_via_quadrant_spacing *distance*

Determines the adjacent via spacing, in microns, in four quadrants. If adjacent vias that cover all four quadrants are not located less than *n* microns from the via, the violation of type "Isolated Via" is reported. If set, this argument will be stored in a "droute" cell parameter "isolatedViaQuadrantSpacing". The range is 0.0 to 50.0. The default is the value, set in a "droute" cell parameter "isolatedViaQuadrantSpacing", which in turn has a default of 10.0.

To set a distance in a "droute" cell parameter "isolatedViaQuadrantSpacing", use
set_parameter -type real -module droute \
 -name isolatedViaQuadrantSpacing -value *n*

DESCRIPTION

This command is used to report isolated vias rule violations. If two vias are not aligned, corner-to-corner spacing is measured for *isolated_via_spacing* and *isolated_via_quadrant_spacing*. If two vias are aligned, edge-to-edge spacing is measured for *isolated_via_spacing* and *isolated_via_quadrant_spacing*. This command will check *isolated_via_spacing* first, and then *isolated_via_quadrant_spacing*. The value of *isolated_via_quadrant_spacing* should be greater than *isolated_via_spacing*. Please be advised that greater values affect the speed of command execution.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports isolated vias rule violations for adjacent vias that are located less than 5.0 microns from the via (isolated_via_quadrant_spacing is 10.0 by default):

```
prompt> report_isolated_via -isolated_via_spacing 5.0
```

APPENDIX

The following settings stored in "droute" cell parameters, and set in command **set_parameter**, will control defaults and operation of **report_isolated_via** command:

Common for all via layers parameters:

"isolatedViaSpacing" and "isolatedViaQuadrantSpacing".

The range for **"isolatedViaSpacing"** is 0.0 to 20.0. The default is 1.0. The range for **"isolatedViaQuadrantSpacing"** is 0.0 to 50.0. The default is 10.0.

To set these parameters, use:

```
set_parameter -type real -module droute \  
              -name isolatedViaSpacing -value n  
set_parameter -type real -module droute \  
              -name isolatedViaQuadrantSpacing -value n
```

To report isolated via violations on specified via layers only, the following settings should be set with command **set_parameter**:

"checkIsolatedViaPerLayer" - if this parameter is set, the rule is applied only to cut layers, which have "per-layer" isolated via spacing set, and if it is not set, the rule is applied to all cut layers. Valid values are 0 (which is the default) and 1. If it is set to 1, the values of common for all via layers parameters **"isolatedViaSpacing"** and **"isolatedViaQuadrantSpacing"** are ignored. The arguments to **report_isolated_via** command that also can set these common parameters, -isolated_via_spacing and -isolated_via_quadrant_spacing, are also ignored.

To set or reset this run-time integer parameter, use:

```
set_parameter -type int -module droute \  
              -name checkIsolatedViaPerLayer -value n
```

To set "per-layer" isolated via spacing and isolated via quadrant spacing, use the following real parameters:

```
set_parameter -type real -module droute \  
              -name polyContIsolatedViaSpacing -value n  
set_parameter -type real -module droute \  
              -name polyContIsolatedViaQuadrantSpacing -value n
```

```

        -name via1IsolatedViaSpacing -value n
set_parameter -type real -module droute \
        -name via2IsolatedViaSpacing -value n
set_parameter -type real -module droute \
        -name via3IsolatedViaSpacing -value n
set_parameter -type real -module droute \
        -name via4IsolatedViaSpacing -value n
set_parameter -type real -module droute \
        -name via5IsolatedViaSpacing -value n
set_parameter -type real -module droute \
        -name via6IsolatedViaSpacing -value n
set_parameter -type real -module droute \
        -name via7IsolatedViaSpacing -value n
set_parameter -type real -module droute \
        -name via8IsolatedViaSpacing -value n
set_parameter -type real -module droute \
        -name via9IsolatedViaSpacing -value n
set_parameter -type real -module droute \
        -name via10IsolatedViaSpacing -value n
set_parameter -type real -module droute \
        -name via11IsolatedViaSpacing -value n

set_parameter -type real -module droute \
        -name polyContIsolatedViaQuadrantSpacing -value n
set_parameter -type real -module droute \
        -name via1IsolatedViaQuadrantSpacing -value n
set_parameter -type real -module droute \
        -name via2IsolatedViaQuadrantSpacing -value n
set_parameter -type real -module droute \
        -name via3IsolatedViaQuadrantSpacing -value n
set_parameter -type real -module droute \
        -name via4IsolatedViaQuadrantSpacing -value n
set_parameter -type real -module droute \
        -name via5IsolatedViaQuadrantSpacing -value n
set_parameter -type real -module droute \
        -name via6IsolatedViaQuadrantSpacing -value n
set_parameter -type real -module droute \
        -name via7IsolatedViaQuadrantSpacing -value n
set_parameter -type real -module droute \
        -name via8IsolatedViaQuadrantSpacing -value n
set_parameter -type real -module droute \
        -name via9IsolatedViaQuadrantSpacing -value n
set_parameter -type real -module droute \
        -name via10IsolatedViaQuadrantSpacing -value n
set_parameter -type real -module droute \
        -name via11IsolatedViaQuadrantSpacing -value n

```

The range for "per-layer" isolated via spacing is 0, 20.0; the default is 0.0, which disables processing isolated via rule on the specified via layer. The range for "per-layer" isolated via quadrant spacing is 0, 50.0; the default is 0.0, which disables processing isolated via quadrant rule on the specified via layer.

SEE ALSO

`fix_isolated_via(2)`

```
set_parameter(2)
report_parameter(2)
```

report_isolation_cell

Displays information about isolation cells in the current scope. This command is supported only in UPF mode.

SYNTAX

```
status report_isolation_cell
[isolation_cells]
[-domain power_domains]
[-strategy isolation_strategy_names]
[-port pins_ports]
[-verbose]
```

Data Types

<i>isolation_cells</i>	list
<i>power_domains</i>	list
<i>isolation_strategy_names</i>	string

ARGUMENTS

isolation_cells
Specifies the isolation cells that are to be reported. If the specified supply cells do not exist in the current scope, the command fails.
By default, all isolation cells in the current scope are reported.

-domain *power_domains*
Specifies the power domains on which to report all isolation cells. If the power domains specified do not exist in the current scope, the command fails.

-strategy *isolation_strategy_names*
Specifies the names of the isolation strategies to report all isolation cells having the specified isolation strategy. When specifying the isolation strategy, a single power domain must be specified. If the specified isolation strategy is not a part of the specified power domain, the command fails.

-port *pins_ports*
Reports isolation cells that are reporting the specified pins and ports.

-verbose
Reports the isolation strategy information in addition to the isolation cells information.

DESCRIPTION

The *report_isolation_cell* command enables you to get detailed information about all isolation cells within the current scope. If the **-domain** option is specified, all isolation cells in the specified domains are reported. If the **-strategy** option is specified, all isolation cells under the specified strategy for the given power domain are reported.

By default, the report displays the following information:

- The name of the port and pin isolated by the isolation cell
- The direction of the port
- The instance name of the isolation cell
- The library reference cell name of the isolation cell.

If you specify the **-verbose** option, the report also displays details of the isolation strategy:

- The name of the isolation strategy
- The location of the cell dictated by the strategy
- The enable signal
- The sense value
- The clamp value
- The elements to which the strategy applies
- The No Isolation information
- The power and ground nets of the strategy.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports on all of the isolation cells in the current scope:

```
prompt> report_isolation_cell

-----

Power Domain   :  bot

=====
| Port | Port Dir.| ISO Cell Name  | ISO Lib Cell      | ISO Strategy      |
=====
| DI   | In       | DI_UPF_ISO    | GTECH_ISO0_EN1    | iso               |
=====

-----

1
```

The following example reports isolation strategy details in addition to the isolation cell information:

```
prompt> report_isolation_cell -verbose
```

```
-----

Power Domain   :   bot
```

Port	Port Dir.	ISO Cell Name	ISO Lib Cell	ISO Strategy
DI	In	DI_UPF_ISO	GTECH_ISO0_EN1	iso

ISO Strategy	Location	Enable Signal	Enable Sense	Clamp Value	Applies to/Elements	No Isolation	ISO Power net	ISO Ground net
iso	parent	EN	low	0	-	-	iso	-

```
1
```

SEE ALSO

```
report_level_shifter(2)
report_retention_cell(2)
set_isolation(2)
set_isolation_control(2)
```

report_keepout_margin

Reports keepout margins of a specified type for the specified cells in the design.

SYNTAX

```
int report_keepout_margin  
[-type hard | soft]  
[-original]  
[-parameters]  
[-all_derivable]  
[object_list]
```

Data Types

object_list list

ARGUMENTS

- type hard | soft
Specifies the type of keepout to be reported. The valid values are **hard** or **soft**. By default, keepouts of both soft and hard types are reported for the cells or library cells.
- original
Reports the original keepout margin values which are set using set_keepout_margin command and orientation of the cell when the keepout margin is set on the cell.
- parameters
Reports the following pin count based parameter values. 1. tracks_per_macro_pin 2. min_padding_per_macro 3. max_padding_per_macro
- all_derivable
Reports all the derivable keepout margin values for the cells specified in the object list. Keepout margin values set on the cells, derivable from the lib cells and pin count based values in case of macros.
- object_list
Describes the list of cells or library cells for which keepouts are to be reported. Keepouts are reported for all cells with FIXED_PLACEMENT or library cells in specified in *object_list*.

DESCRIPTION

This command reports keepouts for the specified cells or library cells. If no hard or soft keepout exists the specified cell or library cell, and this command is called to report on it, a warning message is printed out and no action is taken. This command prints out the margin parameters for a keepout. These are of the form {left_x, bottom_y, right_x, top_y}. All units are in microns.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports hard keepouts for cells in an object list named *MY_CELL*.

```
prompt> report_keepout_margin -type hard MY_CELL
```

The following example reports hard keepouts for library cells in an object list named *MY_LIB_CELL*.

```
prompt> report_keepout_margin -type hard MY_LIB_CELL
```

The following example reports the original hard keepout margin values set on the cell in object list named *MY_CELL*.

```
prompt> report_keepout_margin -type hard MY_CELL -original
```

The following example reports all the pin count based parameters which will be used during the calculation of pin count based keepout margins for macro cells

```
prompt> report_keepout_margin -parameters
```

The following example reports all the derivable keepout margin values for the cells in object_list named *MY_CELL*

```
prompt> report_keepout_margin MY_CELL -all_derivable
```

SEE ALSO

`remove_keepout_margin(2)`

report_latency_adjustment_options

Reports the options for I/O latency adjustment, defined by using the **set_latency_adjustment_options** command.

SYNTAX

```
status report_latency_adjustment_options
```

DESCRIPTION

This command reports the I/O latency adjustment options set earlier by using the **set_latency_adjustment_options** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports I/O latency adjustment options.

```
prompt> report_latency_adjustment_options
```

SEE ALSO

```
set_latency_adjustment_options(2)
```

report_lcc_hotspot

Reports lithography compliance check (LCC) hotspots.

SYNTAX

```
status report_lcc_hotspot  
[-sort_by_attribute layer | level | type]
```

ARGUMENTS

```
-sort_by_attribute layer | level | type  
    Specifies by which attribute the hotspots will be sorted and output.  
    By default, the hotspots are sorted by layer.
```

DESCRIPTION

This command reports the detected hotspots, which are sorted by the specified attribute (layer, level, or type).

You can use this command to output the hotspots in the order that you would like to browse or check.

For each of the reported hotspots, the report includes the following information: type, level, layer, and coordinates.

EXAMPLES

The following example reports the hotspots sorted by level:

```
prompt> report_lcc_hotspot -sort_by_attribute level
```

```
*****  
Report : lcc hotspot  
Design : top  
Version: B-2008.09-SP3  
Date   : Nov. 11, 2008  
*****
```

```
***** LCC Hotspot Report *****
```

HS#	HS Level	HS Layer	HS Type	Bbox/Coordinate
1	1	34	SlotEnd	(1380.585, 713.600)- (1380.985, 714.000)
2	2	34	Line	(685.802, 33.400)- (686.203, 33.800)
3	2	36	Line	(347.402, 1392.600)- (347.802, 1393.000)
4	3	36	Space	(517.700, 1347.402)- (518.100, 1347.802)
5	3	34	Space	(1391.300, 311.440)-

```
(1391.700, 311.840)
6      5      36      Line      (24.600, 0.102)-
(25.000, 0.502)
```

```
***** LCC Hotspot Report End *****
```

SEE ALSO

```
detect_lcc_hotspot(2)
fix_lcc_hotspot(2)
```

report_left_right_filler_rule

Prints out the left and right filler cell insertion rules.

SYNTAX

status **report_left_right_filler_rule**

DESCRIPTION

This command prints out the left and right filler rules of the current design. A left filler rule inserts a left filler cell to the immediate left side of a standard cell. A right filler rule inserts a right filler cell to the immediate right side of a standard cell. A left or right filler cell is often used to avoid certain design rule spacing problems.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

Here is an example of the output.

```
;; Left/Right filler rules set_left_right_filler_rule -left "fillAL" -right "fillAR"
-lib_cell {nd02d0 nr02d0} set_left_right_filler_rule -left "fillBL" -right "fillBR"
-lib_cell {bufbd1 dfnrq1}
```

SEE ALSO

```
insert_stdcell_filler(2)
set_left_right_filler_rule(2)
remove_left_right_filler_rule(2)
```

report_level_shifter

Displays information about level shifter cells in the current scope. This command is supported only in UPF mode.

SYNTAX

```
status report_level_shifter
[level_shifter_cells]
[-domain power_domains]
[-verbose]
[-nosplit]
```

Data Types

<i>level_shifter_cells</i>	list
<i>power_domains</i>	list

ARGUMENTS

level_shifter_cells
Specifies the level shifter cells that are to be reported. If the specified supply cells do not exist in the current scope, the command fails. By default, all level shifter cells in the current scope are reported.

-domain power_domains
Specifies the power domains to report all level shifter cells in the design belonging to the power domains. If the power domain specified does not exist in the current scope, the command fails. This argument is optional and can be used to report level shifter cells selectively.

-verbose
Reports the level shifter strategies along with the level shifter cells.

-nosplit
Prevents line-splitting and facilitates writing software to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

DESCRIPTION

The **report_level_shifter** command enables you to get detailed information about all level shifter cells in the current scope. If the **-domain** argument is specified, all level shifter cells in the specified domains are reported. The report consists of two or three parts, depending on whether or not the **-verbose** option is specified.

- The first part contains the level shifter summary check for a power domain. The summary reports the number of violating level shifter cells that are marked `dont_touch` and those that are not marked `dont_touch`. It also reports the total number of level shifter cells belonging to the power domain.

- If the **-verbose** option is specified, the second part of the report provides the details of the level shifter strategy. The details include the name, type, applies_to/elements, threshold voltage, location, and no shift of the strategy.
- The third part of the report shows the following information about the level shifter cells:
 - Name of the level shifter cell
 - Reference name
 - Input and output power net and ground net information, including name and voltage
 - Main power and ground net name and direction
 - Whether or not there is a violation and the reason for the violation
 - Input and output ranges of the level shifter. Only one range is reported if the input and output ranges are the same.
 - Type of level shifter.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example reports the information about all level shifter cells in the current scope:

```
prompt> report_level_shifter
```

```
-----
-----
Power Domain : PDT
```

```
*****
```

```
Level shifter check summary
```

```
*****
```

```
No. of violating level shifters - dont_touch      : 0
```

```
No. of violating level shifters - no dont_touch   : 0
```

```
No. of level shifters in domain                   : 4
```

```
*****
```

```
=====
```

```
=====
```

Level Shifter	Reference	Input P/G	Output P/G	Main P/
G Violation	Reason	Voltage	Type	
		Net Info	Net Info	Net Info
		(Volt)	(Volt)	(Dir.)

```
=====
```

```
=====
```

sum_outi[3]_UPF_LS	lvl_shifter_2	PNC(0.86)	PNT(1.08)	PNT(Output)	FALSE	-
---	LH					

		PGT(0.00)	PGT(0.00)	VSS(Output)		
sum_outi[2]_UPF_LS	lvl_shifter_2	PNC(0.86)	PNT(1.08)	PNT(Output)	FALSE	-

```

        ---      LH
sum_outi[1]_UPF_LS  lvl_shifter_2  PGT(0.00)  PGT(0.00)  VSS(Output)
        ---      LH
sum_outi[0]_UPF_LS  lvl_shifter_2  PNC(0.86)  PNT(1.08)  PNT(Output) FALSE -
        ---      LH
                                PGT(0.00)  PGT(0.00)  VSS(Output)
=====
=====
1

```

The following example reports information about all level shifter cells in the current scope along with information about the level shifter strategies:

```
prompt> report_level_shifter -verbose
```

```

-----

Power Domain : PD_BOT

*****
Level shifter check summary
*****
No. of violating level shifters - dont_touch      : 0
No. of violating level shifters - no dont_touch   : 3
No. of level shifters in domain                   : 3
*****

=====
Level shifter      Strategy    Applies To/      Threshold    Location    No shift
Strategy name      Elements          Voltage
=====
levshi_bot_1      Always          Both              NA              Inside      FALSE
=====

=====
Level      Reference  Input P/G      Output P/G      Main P/
G          Violation  Reason  Voltage  Type
Shifter                Net Info    Net Info    Net Info
                   (Volt)      (Volt)      (Dir.)
=====
=====
A_UPF_LS  LVLLHX8      MID_VDD(0.90) BOT_VDD(1.08)  MID_VDD(Input) TRUE      main
0.50-1.50 LH
                   VSS(0.00)      VSS(0.00)      VSS(Output)
                                     power
                                     mismatch
B_UPF_LS  LVLLHX8      MID_VDD(0.90) BOT_VDD(1.08)  MID_VDD(Input) TRUE      main
0.50-1.50 LH
                   VSS(0.00)      VSS(0.00)      VSS(Input)
                                     power
                                     mismatch
Y_UPF_LS  LVLHLX8      BOT_VDD(1.08) MID_VDD(0.90)  BOT_VDD(Output) TRUE      main
0.50-1.50 HL

```


	VSS (0.00)	VSS (0.00)	VSS (Both)	power mismatch
=====				
=====				
1				

SEE ALSO

report_isolation_cell(2)
report_retention_cell(2)
set_level_shifter(2)

report_lib

Displays information about the specified logic library, physical library, or symbol library.

SYNTAX

```
status report_lib
[-all]
[-ccs_recv]
[-em]
[-fpga]
[-k_factors]
[-power]
[-power_label]
[-table]
[-full_table]
[-timing]
[-timing_arcs]
[-timing_label]
[-vhdl_name]
[-yield]
[-switch]
[-pg_pin]
[-char]
[-operating_condition]
[-op_cond_name op_cond_name]
[-routing_rule]
[-rwm]
[-user_defined_data]
library_name
[cell_list]
```

Data Types

<i>op_cond_name</i>	string
<i>library_name</i>	string
<i>cell_list</i>	string

ARGUMENTS

-all
Lists all timing-related, power-related, electromigration-related, and FPGA-related information.
This option applies only to logic libraries.

-ccs_recv
Lists CCS and receiver information for library cells.
This option applies only to logic libraries.

-em
Lists all detailed electromigration-related information.
This option applies only to logic libraries.

-fpga
Lists all detailed FPGA-related information, such as parts and I/O cell attributes.
This option applies only to FPGA libraries.

-k_factors
Lists scaling information for library cells.
This option applies only to logic libraries.

-power
Lists all detailed power-related information.
This option applies only to logic libraries.

-power_label
Lists the power labels for all power arcs, if specified, in a tabular form.
This option applies only to logic libraries.

-table
Lists each cell's state table description in compact form.
This option applies only to logic libraries.

-full_table
Lists each cell's state table description in tabular, expanded form.
This option applies only to logic libraries.

-timing
Lists all detailed timing-related information.
This option applies only to logic libraries.

-timing_arcs
Lists all cell timing arcs.
This option applies only to logic libraries.

-timing_label
Lists the timing labels for all timing arcs, if specified, in a tabular form.
This option applies only to logic libraries.

-vhdl_name
Lists the cells and ports whose database (.db) names are different from their VHDL names.
This option applies only to logic libraries.

-yield
Lists failure rate information for library cells.
This option applies only to logic libraries.

-switch
Lists pin switch function information and cell-level steady state current information.
Use the **-pg_pin** option to display the switch function of a power or ground pin.
This option applies only to logic libraries.

-pg_pin
Lists power and ground pin information, such as the PG pin definition and the

voltage name to which a signal pin has been linked to and the PG type.
This option applies only to logic libraries.

-char

Lists cell characterization information, such as sensitization and pre-driver model.
This option applies only to logic libraries.

-operating_condition

Lists all library operating condition information.
When this is the only option selected, the **report_lib** command reports only operating condition information; no other information is reported.

-op_cond_name *op_cond_name*

Lists operating condition information by name.
When this is the only option selected, the **report_lib** command reports only operating condition information, no other information is reported.

-routing_rule

Lists nondefault routing rule information from physical libraries.
This option applies only to physical libraries.

-rwm

Lists routing wire model information from physical libraries.
In a routing wire model report, the original routing wire model is printed out, which includes overlap wire ratio, adjacent wire ratio, wire ratio and wire length of both directions. Also reported is capacitance (in database units) per micron for each layer and for each direction (based on each layer's wire ratio), based on the default operating condition. The report shows resistance per square (in database units) for each direction based on the default operating condition. These derived numbers provide useful information before applying the routing wire model using the **set_routing_wire_model** command.
This option applies only to physical libraries.

-user_defined_data

Lists detailed information about the user-defined groups and attributes.
This option applies to both logic libraries and physical libraries.

library_name

Specifies the name of the library to report.
This is a required argument.

cell_list

Specifies a list of cells about which information is to be reported. The default is to report information about all cells in the logic library or physical library.
This option applies to both logic libraries and physical libraries.

DESCRIPTION

The **report_lib** command displays information about the specified logic library, physical library, or symbol library.

By default, a logic library report displays a list of operating conditions, timing ranges, and wire load models, as well as a listing of all the cells in the library annotated with their attributes. You can restrict the set of cells by using the *cell_list* argument.

Any library object added by using the **update_lib** command is marked with an asterisk (*) following its name, to identify those objects that have been added since the initial library was created with the **read_lib** command.

The **-timing**, **-noise**, **-table**, **-full_table**, **-timing_label**, **-power_label**, **-em**, **-power**, **-yield**, **-pg_pin**, and **-switch** options can be used only with a logic library that was read in from a library source (.lib) file. If the library is read in from a database (.db) file, an error message is issued.

A physical library report displays a list of available layers, a list of vias, a list of sites and a list of cells. If a *cell_list* is specified with **report_lib**, the report includes only the specified cells; otherwise all cells in the given physical library are listed.

A symbol library report displays a list of the names of symbol definitions contained in the library *library_name*. It also reports the route grid and meter scale for the library.

To generate a report, the specified library must be loaded into memory, unless it is found in the search path. The **list_libs** command displays the libraries that are currently loaded.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command generates a report for the mylib logic library. This type of report includes names of the library cells, wire load models, timing ranges, and operating conditions, the date the library was created, and the date the library was generated.

```
prompt> report_lib mylib
```

```
*****
Report : library
Library: mylib
Version: C-2009.06
*****
```

```
Library Type           : Technology
Tool Created           : 2003.06
Date Created           : 21-Jun-101 (INF CREATED ON 1-JUL-2001)
Library Version         : 1.000000
Time Unit               : 1ns

Capacitive Load Unit   : 1.000000pf
```

Pulling Resistance Unit : 1kilo-ohm
Voltage Unit : 1V
Current Unit : 1uA
Dynamic Energy Unit : 1.000000pJ (derived from V,C units)
Leakage Power Unit : 1pW
Bus Naming Style : %s[%d] (default)

Operating Conditions:

Operating Condition Name : mylib
Library : mylib
Process : 1.20
Temperature : 125.00
Voltage : 1.08
Interconnect Model : worst_case_tree

Input Voltages:

No input_voltage groups specified.

Output Voltages:

No output_voltage groups specified.

default_wire_load_capacitance: 0.000096
default_wire_load_resistance: 0.000380
default_wire_load_area: 0.010000

Wire Loading Model:

Name : 05x05
Location : mylib
Resistance : 0.00038
Capacitance : 9.6e-05
Area : 0.01
Slope : 15.9634

Fanout	Length	Points	Average	Cap	Std	Deviation

1	9.80					
2	22.01					
3	35.15					
4	49.29					
5	64.54					
6	80.97					
7	98.68					
8	117.77					
9	138.32					
10	160.41					
11	184.15					
12	209.61					
13	236.90					
14	266.09					
15	297.29					
16	330.57					
17	366.04					

18 403.77
19 443.86
20 486.41

Wire Loading Model Selection Group:

Name : a

Selection		Wire load name
min area	max area	
0.00	5.00	05x05
5.00	10.00	10x10
10.00	20.00	20x20
20.00	30.00	30x30

Wire Loading Model Mode: enclosed.

Wire Loading Model Selection Group: a.

Porosity information:

No porosity information specified.

Routing Layers:

"metal1"	"metal2"	"metal3"	"metal4"	"metal5"
"metal6"	"metal7"	"metal8"		

default_min_porosity: 0.000000

In_place optimization mode: match_footprint

Timing Ranges:

No timing ranges specified.

Delay Threshold Trip-Points:

input_threshold_pct_rise:	50
output_threshold_pct_rise:	50
input_threshold_pct_fall:	50
output_threshold_pct_fall:	50

Slew Threshold Trip-Points:

slew_lower_threshold_pct_rise:	20
slew_upper_threshold_pct_rise:	80
slew_lower_threshold_pct_fall:	20
slew_upper_threshold_pct_fall:	80
slew_derate_from_library:	0.6

Components:

Attributes:

- af - active falling
- ah - active high
- al - active low
- ar - active rising
- b - black box (function unknown)
- ce - clock enable
- cg - clock gating integrated cell
- ub - unbuffered
- pgt - pass gate
- ctl - has CTL test model
- d - dont_touch
- iso - isolation cell
- ls - level shifter
- mo - map_only
- 1p0g - power only, no ground
- 0p1g - ground only, no power
- 0p0g - no power/ground
- p - preferred
- hp - hold_preferred
- pwr - power gating cell
- r - removable
- s - statetable
- sa0 - dont_fault stuck-at-0
- sa1 - dont_fault stuck-at-1
- sa01 - dont_fault both stuck-at-0 and stuck-at-1
- sz - use_for_size_only
- t - test cell. t(scan_type) as
 - t(mux_ff) - muxed flip-flop
 - t(mux_ld) - muxed latch
 - t(clk_scan) - clocked scan
 - t(clk_scan_ld) - clocked scan latch
 - t(lssd_dld) - double latch LSSD
 - t(lssd_sld) - single latch LSSD
 - t(lssd_clk) - clocked LSSD
 - t(lssd_auxclk) - auxiliary clock LSSD
- u - dont_use
- udp - usable for datapath generators
- ufc - user_function_class. ufc(type) as
 - ufc(u) - user-defined
 - ufc(a) - automatically generated
- imc - macro cell
- switch_cg - switch cell (coarse grain)
- ret - retention cell
- ao - always on cell
- gicg - generic integrated clock gating
 - va - variation-aware ccs timing modeling
 - val - variation-aware ccs leakage power modeling
- fil - filler cell
- tap - tap cell
- decap - decap cell
- pll - pll cell

Pin Attributes:

- ao - always on pin
- lse - level shifter enable pin
- isoe - isolation cell enable pin
- scmr - standard cell main rail

Cell	Footprint	Attributes

AN2		
OR2		

1

The following command generates a logic library report that lists all CCS and receiver information for the library cells:

prompt> **report_lib mylib -ccs_recv**

```
*****
Report : library
Library: mylib
Version: C-2009.06
*****
...
```

Components:

```
Attributes:
  af - active falling
  ah - active high
  al - active low
  ...
  ccs - composite current source
  c_ccs - compact ccs timing modeling
  ccsn - ccs noise data modeling
  recv - receiver model
  ...
  tap - tap cell
  decap - decap cell
  pll - pll cell
```

Pin Attributes:

- ao - always on pin
- lse - level shifter enable pin
- isoe - isolation cell enable pin
- scmr - standard cell main rail

Cell	Footprint	Attributes

AN2		ccs, recv
OR2		ccs

1

The following command generates a report on library timing in the AND2 cell of the tech_lib logic library:

prompt> **report_lib tech_lib -timing AND2**

```
*****
Report : library
Library: tech_lib
Version: C-2009.06
*****
```

```
Library Type      : Technology
Tool Created      : 2003.06
Date Created      : 21-Jun-101 (INF CREATED ON 1-JUL-2001)
Library Version   : 1.000000
```

```
Delay Model       : table_lookup
Time Unit         : 1ns
```

Timing Attributes Defaults:

Attribute	Default
max_transition	
max_fanout	
fanout_load	1
max_capacitance	

Lookup Table Template:

```
Template_name
-----
del_0_5_7_w
  VARIABLE_1: input_net_transition
  VARIABLE_2: total_output_net_capacitance
  INDEX_1:   0.0150  0.2500  0.6500  1.4000  3.0000
  INDEX_2:   0.0000  0.0035  0.0070  0.0192  0.0402  0.0752  0.1750
  ...
```

CELL(AND2): 1.25;

```
PIN(A): in, 0.0035, 1, 3, , ;
END_PIN A;
```

```
PIN(B): in, 0.00375, 1, 3, , ;
END_PIN B;
```

```
PIN(Z): out, 0, 18, 2, 0.175, , ;
DELAY: A, Z, prop, pos_unate, '', ( , ), ( , ), ( , );
  cell_rise ( del_0_5_7_w ) :
    VALUES :    0.0710  0.1000  0.1240  0.2050  0.3440  0.5750
                1.2330  0.1120  0.1410  0.1650  0.2470  0.3860
                0.6170  1.2750  0.1520  0.1840  0.2100  0.2920
                0.4320  0.6630  1.3210  0.1990  0.2360  0.2630
                0.3470  0.4870  0.7190  1.3770  0.2630  0.3080
                0.3400  0.4280  0.5700  0.8030  1.4630

  cell_fall ( del_0_5_7_w ) :
    VALUES :    0.0620  0.0900  0.1160  0.2020  0.3500  0.5950
                1.2940  0.1050  0.1360  0.1620  0.2480  0.3960
```

0.6420	1.3410	0.1440	0.1770	0.2040	0.2930
0.4410	0.6860	1.3860	0.1870	0.2240	0.2530
0.3430	0.4920	0.7390	1.4380	0.2410	0.2850
0.3180	0.4130	0.5650	0.8140	1.5150	

```

rise_transition ( del_0_5_7_w ) :
  VALUES :    0.0450  0.0960  0.1480  0.3400  0.6720  1.2270
               2.8080  0.0490  0.0980  0.1500  0.3400  0.6720
               1.2270  2.8080  0.0580  0.1060  0.1560  0.3440
               0.6740  1.2270  2.8080  0.0720  0.1200  0.1660
               0.3480  0.6770  1.2300  2.8080  0.0930  0.1460
               0.1910  0.3630  0.6860  1.2370  2.8120

```

```

fall_transition ( del_0_5_7_w ) :
  VALUES :    0.0370  0.0900  0.1440  0.3380  0.6720  1.2300
               2.8190  0.0430  0.0930  0.1460  0.3380  0.6720
               1.2300  2.8200  0.0510  0.1020  0.1530  0.3420
               0.6740  1.2300  2.8190  0.0650  0.1150  0.1640
               0.3480  0.6780  1.2330  2.8200  0.0870  0.1400
               0.1870  0.3660  0.6920  1.2430  2.8250

```

```

DELAY: B, Z, prop, pos_unate, '', ( , ), ( , ), ( , );
...

```

```

END_PIN Z;
END_CELL AND2;

```

```

Number of library cells: 1
Number of reported cells: 1
1

```

The following command generates a report on library noise in the AND2 cell of the tech_lib logic library:

```

prompt> report_lib tech_lib -noise AND2
CELL(AND2): 4, ;

```

```

PIN(A): in, 2, , , ;
END_PIN A;

```

```

PIN(B): in, 2, , , ;
END_PIN B;

```

```

PIN(Z): out, , , , , , ( , ), ( , );
noise_immunity_high ( noise5x5 ) :
  INDEX_2 :    0.3620  0.7250  1.0870  1.4490  1.8120
  VALUES :    0.7330  1.0730  1.4120  1.7520  2.0920  0.8730
               1.2140  1.5540  1.8940  2.2340  1.0950  1.4420
               1.7870  2.1320  2.4770  1.2980  1.6480  1.9960
               2.3430  2.6910  1.7030  2.0600  2.4140  2.7660
               3.1190

```

```

END_PIN Z;
END_CELL AND2;

```

For details about library timing report contents and syntax, refer to the *Library*

The following example generates a report on power information in the tech_lib logic library:

```
prompt> report_lib tech_lib -power
```

Power Information:

Attributes:

- a - average power specification
- i - internal power
- l - leakage power
- rf - rise and fall power specification

Power					
Cell	#	Attr	Toggling pin	source of path	When
AN2	0	l			
	1	i,a	Z	A	
	2	i,a	Z	B	
INV	0	l			
	1	i,a	Z	A	
NO2	0	l			
	1	i,a	Z	A	
	2	i,a	Z	B	
flop1	0	l			
	1	i,a	CP		
	2	i,a	Q	CP	
	3	i,a	Q	D	
	4	i,a	QN	CP	
	5	i,a	QN	D	
latch1	0	l			
	1	i,a	G		
	2	i,a	Q	G	
	3	i,a	Q	D	
	4	i,a	QN	G	
	5	i,a	QN	D	

The following command generates a report on library electromigration in the ad3 cell of the tech_lib logic library:

```
prompt> report_lib tech_lib -em ad3
```

CELL(ad3): 2;

PIN(y): out, 0, , , 1.812, , ;

ELECTROMIGRATION: a;

em_max_toggle_rate (output_by_cap_and_trans) :

VALUES : 2.0000 1.0000 0.5000 1.5000 0.7500 0.3300
1.0000 0.5000 0.1500

ELECTROMIGRATION: b;

em_max_toggle_rate (output_by_cap_and_trans) :

VALUES : 2.0000 1.0000 0.5000 1.5000 0.7500 0.3300
1.0000 0.5000 0.1500

```

ELECTROMIGRATION: c;
  em_max_toggle_rate ( output_by_cap_and_trans ) :
    VALUES :    2.0000  1.0000  0.5000  1.5000  0.7500  0.3300
               1.0000  0.5000  0.1500

```

```
END_PIN y;
```

```

PIN(a): in, 1, , ;
END_PIN a;

```

```

PIN(b): in, 1, , ;
END_PIN b;

```

```

PIN(c): in, 1, , ;
ELECTROMIGRATION:
  em_max_toggle_rate ( input_by_trans ) :
    VALUES :    1.5000  1.0000  0.5000

```

```

END_PIN c;
END_CELL ad3;

```

```

Number of library cells: 1
Number of reported cells for em: 1
1

```

For details about library electromigration report contents and syntax, see the *Library Compiler Modeling Timing, Signal Integrity, and Power in Technology Libraries User Guide*.

The following command generates a report on timing arcs in the tech_lib logic library. The INV_NEW cell is identified as having been added using the **update_lib** command:

```
prompt> report_lib tech_lib -timing_arcs
```

Cell	Attributes	Arc #	Sense/type	Arc Pins From	To	When
AND		0	pos unate	A	Z	
		1	pos unate	B	Z	
INVET		0	neg unate	A	Z	
INV_NEW		0	neg unate	A	Z	

If the **-timing_arcs** option is not used, the last five columns do not appear.

The following command generates a report for a symbol library. The report includes names of all symbol definitions, the route_grid, and meter_scale:

```
prompt> report_lib sym_lib.sdb
```

The following command generates a report that displays timing arcs for the inverter cell from the tech_lib logic library:

```
prompt> report_lib tech_lib -timing_arcs inverter
```

The following example generates a report that displays noise arcs for the inverter cell from the tech_lib logic library:

```
prompt> report_lib tech_lib -noise_arcs inverter
```

The following is an example of a compact report on state table information in the tech_lib logic library:

```
prompt> report_lib tech_lib -table
```

State table descriptions:

CELL(D_LATCH) :

```
PIN(Q) : CD, D, G
TABLE: HNLNLLLL
END_PIN Q;
```

```
PIN(QN) :
STATE_FUNCTION: Q'
END_PIN QN;
END_CELL D_LATCH;
```

The following is an example of a tabular report on state table information in the tech_lib logic library:

```
prompt> report_lib tech_lib -full_table
```

State table descriptions:

CELL(D_LATCH) :

```
PIN(Q) : CD, D, G
TABLE:
000    L
001    L
010    L
011    L
100    N
101    L
110    N
111    H
END_PIN Q;
```

```
PIN(QN) :
STATE_FUNCTION: Q'
END_PIN QN;
```

END_CELL D_LATCH;

The following is an example of a pg_pin report in the tech_lib logic library:

```
prompt> report_lib tech_lib -pg_pin
```

CELL(LVLHLEHX2) :

```

PG_PIN(VDDI) :
    VOLTAGE_NAME: VDDH
    PG_TYPE: primary_power
END_PIN VDDI;

PG_PIN(VSS) :
    VOLTAGE: VSS
    PG_TYPE: primary_ground
END_PIN VSS;

PIN(QN) :
    RELATED_POWER_PIN : VDDI
    RELATED_GROUND_PIN : VSS
END_PIN QN;

```

```
END_CELL LVLHLEHX2;
```

The following is an example of a characterization report in the tech_lib logic library:

```
prompt> report_lib tech_lib -char
```

```

*****
Report : library
Library: tech_lib
Version: A-2007.12
*****

Library Type           : Technology
Tool Created           : A-2007.12-BETA3
Date Created           : Not Specified
Library Version        : Not Specified

```

```

Sensitization : 2IN_1OUT
  pin_names : (A, B, C)
  vector(0) : (0, 0, 0)
  vector(1) : (0, 0, 1)
  vector(2) : (0, 1, 0)
  vector(3) : (0, 1, 1)
  vector(4) : (1, 0, 0)
  vector(5) : (1, 0, 1)
  vector(6) : (1, 1, 0)
  vector(7) : (1, 1, 1)
  vector(8) : (1, 1, 1)

```

```

CELL(AN2): 2;
  SENSITIZATION_MASTER: 2IN_1OUT
  PIN_NAME_MAP : (A, B, Z)

  PIN(A): in, 1, , , , ;
  END_PIN A;

  PIN(B): in, 1, , , , ;
  END_PIN B;

```

```

PIN(Z): out, 0, , , , , ;
  DELAY: A, Z, prop, pos_unate, '', (1, 1), (0, 0), (0.1443, 0.0523);
    sensitization_master: 2IN_1OUT
    pin_name_map : (A, B, Z)
    wave_rise : (3, 4, 5)
    wave_fall : (0, 3, 7)
    wave_rise_sampling_index: 2
    wave_fall_sampling_index: 2
    wave_rise_timing_interval : (0, 0.5)
    wave_fall_timing_interval : (0, 0.45)
  DELAY: B, Z, prop, pos_unate, '', (0.48, 0.77), (0, 0), (0.1443, 0.0523);
END_PIN Z;
END_CELL AN2;

```

The following is an example of a switch cell report in the tech_lib logic library:

```

prompt> report_lib tech_lib -switch
CELL(FD1): 2;

```

```

DC_CURRENT:
  dc_current ( ccsn_dc_29x29 ) :
    RELATED_SWITCH_PIN : D
    RELATED_PG_PIN : PWR
    RELATED_INTERNAL_PG_PIN : VVDD
    INDEX_1 :
      -1.2000 -0.6000 -0.2400 -0.1200 0.0000 0.0600
      0.1200 0.1800 0.2400 0.3000 0.3600 0.4200
      0.4800 0.5400 0.6000 0.6600 0.7200 0.7800
      0.8400 0.9000 0.9600 1.0200 1.0800 1.1400
      1.2000 1.3200 1.4400 1.8000 2.4000

```

```

...
END_CELL AN2;

```

SEE ALSO

```

compare_lib(2)
read_lib(2)
update_lib(2)
write_lib(2)

```


report_milkyway_version

Reports information for the specified cell, including the Milkyway data model version and revision information.

SYNTAX

```
status report_milkyway_version  
[-cell cell_name | -all]
```

Data Types

cell_name string

ARGUMENTS

```
-cell cell_name  
    Generates a report for the specified cell.  
  
-all  
    Generates a report for all cells in the current Milkyway design library.
```

DESCRIPTION

This command reports the following for each reported cell:

- The cell data model version
- The product version that is compatible with the cell data model version
- The product release version that created the initial design
- The cell's initial creation time
- The product release version that modified the design last
- The cell's last modification time

All cells must be closed when running this command.

Either the -cell or -all option must be specified. If neither option is specified, this command will display an error message and fail. If both options are specified, the -all option will take precedence and a report is generated for all cells.

EXAMPLES

The following example displays a report for the cell "top" in the current Milkyway design library.

```
prompt> open_mw_lib design  
prompt> report_milkyway_version -cell top  
  
Cell Name                    top.CEL;2  
Data Model                  1.2  
Compatible Product          A-2007.12 And Older Versions  
Creator                     Z-2007.03-ICC-SP2-1
```

```
Creation Time      Fri Jul 27 11:10:04 2007
Modifier          Z-2007.03-ICC-SP2-1
Modification Time  Mon Jul 30 17:34:34 2007
1
```

The following example displays a report for all cells in the current Milkyway design library.

```
prompt> open_mw_lib design
prompt> report_milkyway_version -all
```

```
Cell Name          top.CEL;1
Data Model         1.2
Compatible Product A-2007.12 And Older Versions
Creator            Z-2007.03-ICC-SP2-1
Creation Time      Mon Jul 23 16:00:45 2007
Modifier           Z-2007.03-ICC-SP2-1
Modification Time  Mon Jul 23 16:01:08 2007
```

```
Cell Name          top.CEL;2
Data Model         1.2
Compatible Product A-2007.12 And Older Versions
Creator            Z-2007.03-ICC-SP2-1
Creation Time      Fri Jul 27 11:10:04 2007
Modifier           Z-2007.03-ICC-SP2-1
Modification Time  Mon Jul 30 17:34:34 2007
```

```
Cell Name          macro1.CEL;1
Data Model         1.2
Compatible Product A-2007.12 And Older Versions
Creator            Z-2007.03-ICC-SP2-1
Creation Time      Mon Jul 23 16:00:46 2007
Modifier           Z-2007.03-ICC-SP2-1
Modification Time  Mon Jul 23 16:01:08 2007
```

```
Cell Name          macro1.CEL;2
Data Model         3.2
Compatible Product B-2008.09
Creator            Z-2007.03-ICC-SP2-1
Creation Time      Mon Jul 23 16:00:46 2007
Modifier           B-2008.09-ICC-SP-INTERNAL
Modification Time  Tue Sep  9 14:30:54 2008
1
```

SEE ALSO

```
convert_mw_lib(2)
open_mw_lib(2)
```

report_mim

Generates a report on the multiple instantiated modules (MIMs) in the design.

SYNTAX

status **report_mim**

DESCRIPTION

This command prints a report on all the multiple instantiated modules (MIMs) in the design. This report describes which plan groups or soft macros are MIMs plus additional information on their status. The additional information includes which MIM instance was selected as the master. Also the additional information tells you what is the status of the copied placement data (from **copy_mim**) and also a suggested MIM plan group flipping grid. An MIM plan group flipping grid is a grid on which you can place an MIM plan group such that when it is flipped, the standard cells placement remains legal.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows sample output of the **report_mim** command.

```
prompt> report_mim
*****
Report : MIM
Design : orca.CEL;1
Version: B-2008.09-ICC
Date   : Fri Feb 15 10:32:44 2008
*****

Reference tarang is instantiated to:
    Plan Group: tarang1 (orientation N)
    Plan Group: tarang2 (orientation N) (is_master)
    Plan Group: tarang3 (orientation N)
    Plan Group: tarang4 (orientation N)
    Reference tarang is instantiated 4 times.
Total 1 MIM reference and 4 MIM instances.
MIM plan group flipping grid: x offset=255.000, x step=0.560
                             y offset=255.000, y step=8.960
Copy_mim stored placement data...
MIM group #0: The copy source is tarang1, stored orientation: N
MIM group #0: A copy target is   tarang2, stored orientation: N
MIM group #0: A copy target is   tarang3, stored orientation: N
MIM group #0: A copy target is   tarang4, stored orientation: N
MIM Report End
1
```

SEE ALSO

`copy_mim(2)`
`flip_mim(2)`

report_mode

Prints a report of the instance modes.

SYNTAX

```
string report_mode  
[-nosplit]  
[instance_list]
```

Data Types

instance_list list

ARGUMENTS

-nosplit

Prevents line-splitting and facilitates writing software to extract information from the report output. By default, most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

instance_list

Indicates that the mode report is to include only the specified cells. By default, all cells that have modes are included.

DESCRIPTION

Reports which cells have modes and, for each mode, specifies that the mode is currently enabled or disabled. This reports reflects the mode specifications of the **set_mode** command.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example reports that 2 cells in the design have modes: cell Uram1, and cell Uram2.

```
prompt> report_mode  
  
*****  
Report : mode  
Design : top_design  
*****
```

Cell	Mode (Group)	Status

-		
Uram1/core (RAM2_core)	read(rw)	ENABLED
	write(rw)	ENABLED

-		
Uram2/core (RAM2_core)	read(rw)	ENABLED
	write(rw)	ENABLED

-		

The following example reports that modes are specified for the 2 RAMs that have mode in the design. Ram Uram1 is set in mode read, and Ram Uram2 is set in mode write. This means all timing arcs of RAM Uram1 associated with mode read are enabled, and all timing arcs associated with mode write are disabled.

```
prompt> set_mode read Uram1/core
prompt> set_mode write Uram2/core
prompt> report_mode
```

```
*****
```

```
Report : mode
```

```
Design : top_design
```

```
*****
```

Cell	Mode (Group)	Status

-		
Uram1/core (RAM2_core)	read(rw)	ENABLED
	write(rw)	disabl

-		
Uram2/ core (RAM2_core)	read(rw)	disabled
	write(rw)	ENABLED

-		

SEE ALSO

```
set_mode(2)
reset_mode(2)
```

report_mpc_macro_array

Issues a report on the floorplan macro array set by using the **set_mpc_macro_array** command.

SYNTAX

```
status report_mpc_macro_array
[macro_array]
```

Data Types

macro_array string

ARGUMENTS

macro_array
Reports floorplan macro options for all macro arrays or the macro arrays listed in *macro_array*.

DESCRIPTION

This command reports various options set by the **set_mpc_macro_array** command.

For more information about these floorplan options, read the man page description of the **set_mpc_macro_array** command.

Multicorner-Multimode Support

This command is not supported in a multi-scenario design flow.

EXAMPLES

The following example shows a sample report.

```
prompt> report_mpc_macro_array
*****
Report floorplan macro array
Design : h2anlb_translator
Version: 2002.05-PCE1-beta2-dev
Date   : Fri Aug 30 10:27:17 2002
*****
Name      Type      Offset      Align      Elements
          X        Y        Edge
-----
TT        4 x 1      0.00   30.00   left
  Row 1: HOST2LDT_I_H2LW_BUF_MANAGER_H2LW_DATA_BUF_R8X128M1
_BLOCK_0_INSTANCE_1_R8X128M1_INSTANCE_2
  Row 2: HOST2LDT_I_H2LW_BUF_MANAGER_H2LW_BE_BUF_
```

```
R8X16M1_BLOCK_0_INSTANCE_1_R8X16M1_INSTANCE_2
  Row 3: LDT2HOST_L2HR_BUF_R8X128M1_BLOCK_0_INSTANCE_1
_R8X128M1_INSTANCE_2
  Row 4: HOST2LDT_I_H2LR_BUF_MANAGER_H2LR_DATA_BUF_
R16X128M2_BW_BLOCK_0_INSTANCE_1_R16X128M2_BW_INSTANCE_2
*****
1
```

SEE ALSO

```
set_mpc_macro_array(2)
set_mpc_macro_options(2)
```


report_mpc_macro_options

Issues a report about the floorplan macro options.

SYNTAX

```
int report_mpc_macro_options  
macro_list
```

ARGUMENTS

macro_list
Reports floorplan macro options for the macros listed.

DESCRIPTION

This command reports various options set by the **set_mpc_macro_options** command to constrain macros during the automatic generation of the floorplan, using the minimal physical constraints method.

For more information about these floorplan options, read the man page description of the **set_mpc_macro_options** command.

Multicorner-Multimode Support

This command is not supported in a multi-scenario design flow.

EXAMPLES

The following example shows a sample report:

```
prompt> report_mpc_macro_options  
  
*****  
Report floorplan macro options  
Design : h2anlb_translator  
Version: 2002.05-PCE1-beta2-dev  
Date   : Fri Aug 30 10:27:17 2002  
*****  
Macro          Anchor      Offset      Legal  
Name           Bound       X      Y      Orientation  
-----  
LDT2HOST_L2HR_BUF_R8X128M1_BLOCK_0_INSTANCE_1_R8X128M1_INSTANCE_2  
                -          -      -          N  
HOST2LDT_I_H2LW_BUF_MANAGER_H2LW_BE_BUF_R8X16M1_BLOCK_0_INSTANCE_1_R8X16M1_INST  
ANCE_2  
                -          -      -          N  
HOST2LDT_I_H2LW_BUF_MANAGER_H2LW_DATA_BUF_R8X128M1_BLOCK_0_INSTANCE_1_R8X128M1_  
INSTANCE_2  
                -          -      -          N  
HOST2LDT_I_H2LR_BUF_MANAGER_H2LR_DATA_BUF_R16X128M2_BW_BLOCK_0_INSTANCE_1_R16X1
```

```

28M2_BW_INSTANCE_2
                bottom-left  10.00 10.00      N
TT              bottom-left  10.00 10.00      -
*****

```

1

SEE ALSO

```

set_mpc_macro_options(2)
set_mpc_options(2)
set_mpc_port_options(2)
set_mpc_macro_array(2)

```

report_mpc_options

Issues a report about the floorplan options.

SYNTAX

```
int report_mpc_options
```

ARGUMENTS

The **report_mpc_options** command has no arguments.

DESCRIPTION

This command reports all the floorplan options set for running a minimal physical constraints flow. Use these options to generate a floorplan automatically, using the **set_mpc_options** command. Also, see the **set_mpc_options** command man page to learn the default values of the options and a description of how each option relates to the automatically generated floorplan.

Multicorner-Multimode Support

This command is not supported in a multi-scenario design flow.

EXAMPLES

The following example shows a report:

```
prompt> report_mpc_options
      Report of floorplan options
      -----
Utilization : 0.300000.
Aspect ratio : 1.000000.
Top port limit : 2.
Left port limit : 1.
First row orientation : NORTH.
Ports snapping : enabled.
Default horizontal port layer : Metall1.
Default vertical port layer : Metal2.
Left io margin : 20.000000.
Right io margin : 20.000000.
Top io margin : 20.000000.
Bottom io margin : 20.000000.
Core origin : 0, 0.
No site names specified.
Core row direction : horizontal.
```

1

SEE ALSO

`set_mpc_macro_options(2)`
`set_mpc_options(2)`
`set_mpc_port_options(2)`

report_mpc_pnet_options

Issues a report about the floorplan pnet options.

SYNTAX

```
int report_mpc_pnet_options  
-name
```

ARGUMENTS

-name
Display the pnet options for pnet options matching name.

DESCRIPTION

This command report all the options set using the **set_mpc_pnet_options** command.

Multicorner-Multimode Support

This command is not supported in a multi-scenario design flow.

EXAMPLES

The following example shows a report:

```
prompt> report_mpc_pnet_options  
report_mpc_pnet_options  
*****  
Report floorplan pnet options  
Design : fir_filter  
Version: 2003.03-PSYN-ALPHA2  
Date   : Wed Oct 16 19:51:38 2002  
*****  
Name          Layer  Type Width Pitch Offset Dir      Model Mul_X M  
ul_Y  
-----  
---  
trunk1        MET6   PWR  20.000 100.000 0.000 Horiz  TRUE  -   -  
trunk2        MET4   GND  20.000 100.000 60.00 Horiz  TRUE  -   -  
*****  
1
```

SEE ALSO

```
set_mpc_macro_options(2)  
set_mpc_options(2)  
set_mpc_pnet_options(2)  
set_mpc_port_options(2)
```


o5	bottom	2.000000	2.000000	2.000000	4.
000000					
o6	bottom	2.000000	2.000000	2.000000	4.
000000					
o7	bottom	2.000000	2.000000	2.000000	4.
000000					
o8	bottom	2.000000	2.000000	2.000000	4.
000000					
1					

SEE ALSO

[set_mpc_options\(2\)](#)
[set_mpc_port_options\(2\)](#)

report_mpc_rectilinear_outline

Issues a report about the rectilinear outline of a floorplan.

SYNTAX

status **report_mpc_rectilinear_outline**

ARGUMENTS

The **report_mpc_rectilinear_outline** command has no arguments.

DESCRIPTION

This command reports the rectilinear outline set by the **set_mpc_rectilinear_outline** command. See the man page of the **set_mpc_rectilinear_outline** command for more detail.

Multicorner-Multimode Support

This command is not supported in a multi-scenario design flow.

EXAMPLES

The following example shows a sample report using this command.

```
report_mpc_rectilinear_outline
*****
Report mpc core outline
Design : MRISC
Version: W-2004.12
Date   : Fri Sep 24 18:02:05 2004
*****
      {0.000 0.000} {1000.000 0.000} {1000.000 1500.000}
      {500.000 1500.000} {500.000 1000.000} {0.000 1000.000}
1
```

SEE ALSO

set_mpc_rectilinear_outline(2)

report_mpc_ring_options

Issues a report about the mpc ring options.

SYNTAX

```
int report_mpc_ring_options
```

ARGUMENTS

The **report_mpc_ring_options** command has no arguments.

DESCRIPTION

This command reports all the options set by using the **set_mpc_ring_options** command.

Multicorner-Multimode Support

This command is not supported in a multi-scenario design flow.

EXAMPLES

The following example shows a report:

```
prompt> report_mpc_ring_options
report_mpc_ring_options
*****
Report floorplan Ring options
Design : MRISC
Version: U-2003.06-CTGbeta2_PSYNalpha
Date   : Thu Feb 20 19:16:26 2003
*****
  Name                Layer  Type Width Offset      Sides Object
-----
  mpc_ring_GND_m1_15_0 m1      GND  2.000 -2.000    [lr]bt] MRISC
  mpc_ring_PWR_m2_15_0 m2      PWR  2.000 -5.000    [lr]bt] MRISC
*****
1
```

SEE ALSO

```
set_mpc_macro_options(2)
set_mpc_options(2)
set_mpc_pnet_options(2)
set_mpc_port_options(2)
set_mpc_ring_options(2)
```

report_mtcmos_pna_strategy

Prints the current values and default values for the parameters controlled by **set_mtcmos_pna_strategy**.

SYNTAX

status_value **report_mtcmos_pna_strategy**

ARGUMENTS

The **report_mtcmos_pna_strategy** command has no arguments.

DESCRIPTION

Prints the current values and default values for the parameters controlled by **set_mtcmos_pna_strategy**. These parameters are used to control power network synthesis (PNS) and power network analysis (PNA), when exploring, replacing, optimizing MTCMOS cell size (**optimize_header_footer**, **explore_header_footer** and **replace_header_footer**).

Multicorner-Multimode Support

This command is not supported in a multi-scenario design flow.

EXAMPLE

The following example shows a sample output of the **report_mtcmos_pna_strategy** command.

```
prompt> report_mtcmos_pna_strategy

Power Budget:                default
Voltage Supply:              default
Lowest Voltage Drop:         false
Target Voltage Drop:         default
Pad Lib Cell:                null
Read Pad Cell File:          null
Read Pad Lib Cell File:      null
Use Strap Ends As Pads:      false
Use Pins As Pads:            false
Create Virtual Rails:        null
Synthesize Voltage Areas:    null
Disable Snap To Row&Tile:    false
Relative To Voltage Area:    false
Pattern for inserting MTCMOS: normal
1
prompt>
```

SEE ALSO

`set_mtcmos_pna_strategy(2)`
`analyze_fp_rail(2)`
`synthesize_fp_rail(2)`

report_mw_design_ecos

Report ECO history information and output specific ECO changelist

SYNTAX

```
int report_mw_design_ecos  
[design_name]  
[-output outfile_name]  
[-history]  
[-id id_number]
```

Data Types

<i>design_name</i>	string
<i>outfile_name</i>	string
<i>id_number</i>	integer

ARGUMENTS

design_name
The name of Milkyway design cell whose history will be reported

-output *outfile_name*
The file name to which the command output change list or history information;
if not specified, stdout will be used to output.

-history
This option will make the command to report ECO history of all changes,
including flat ECO changes and hierarchical ECO changes in the design cell.

-id *id_number*
This option specifies the ECO Id to be reported; if not specified, the latest
ECO change will be reported.
NOTE: The hierarchy preservation data and the flat design data of the cells
must be consistent. Both cells will be opened in read-only mode.

DESCRIPTION

This command will report all ECO change history of given design or output specific ECO change list of given id number.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

Following are examples of reporting ECO history information on Milkyway design cells.

```
report_mw_design_ecos  
1886
```

The following example is to get latest ECO change list

```
prompt> report_mw_design_ecos
```

The following example is to get latest ECO change list and save to ECO change file

```
prompt> report_mw_design_ecos  
        -output eco_changefile
```

The following example is to report all ECO history information

```
prompt> report_mw_design_ecos  
        -history
```

The following example is to get specific ECO change list by ECO id of given design cell and save the list to ECO change file

```
prompt> report_mw_design_ecos  
        -id 3  
        -output eco_changefile_3
```

SEE ALSO

`read_mw_eco_list(2)`

report_mw_lib

Displays information about a Milkyway library.

SYNTAX

```
status_value report_mw_lib
[-unit_range]
[-mw_reference_library]
mw_lib
```

Data Types

mw_lib string

ARGUMENTS

-unit_range
Indicates to list the information of the unit. This option requires the library to be reported on to be specified.

-mw_reference_library
Prints the list of reference libraries for the Milkyway library. If the library is not be specified, the current Milkyway library is used.

mw_lib
Specifies the Milkyway library to be reported. The default is the current Milkyway library.

DESCRIPTION

This command displays information about a Milkyway library.

A status indicating success or failure is returned.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example displays Reference Library information about a Milkyway library design:

```
prompt> report_mw_lib -mw_reference_library design
./libs/mw/nand_macro_reflib_018
1
```

The following example displays unit information about a Milkyway library design:

```
report_mw_lib
1888
```

prompt> **report_mw_lib -unit_range design**

Library: design

Tech. Attr.	Unit	Resolution	Min. Value	Max. Value
=====	=====	=====	=====	=====
length	micron	1000	0.001	2147483.647
time	ns	1000	0.001	2147483.647
capacitance	pf	10000000	0.0000001	214.7483647
resistance	kohm	10000000	0.0000001	214.7483647

Layer: POLY

Mask name: poly

Attribute	Minimum	Maximum
=====	=====	=====
thickness	0.0e+00	0.0e+00
unit resistance	0.0e+00	0.0e+00
unit capacitance	0.0e+00	0.0e+00

.....

1

SEE ALSO

close_mw_lib(2)

open_mw_lib(2)

report_name_rules

Reports the values of name rules.

SYNTAX

```
status report_name_rules
[name_rules]
```

Data Types

```
name_rules      string
```

ARGUMENTS

```
name_rules
    Specifies the name of the rules to be reported. If the name_rules option is
    not specified, all currently defined name rules are reported.
```

DESCRIPTION

Reports the values of a set of name rules. Name rules are created or modified by the **define_name_rules** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses the **report_name_rules** command to report the values of the name rules called EXAMPLE.

```
prompt> report_name_rules EXAMPLE
```

```
*****
Report : name_rules
Name Rules : EXAMPLE
Version: v3.0
Date    : Fri Sep 27 11:10:20 1991
*****
```

```
Rules Name: EXAMPLE
    Equal port and net names: false
    Collapse name space: false
    Case insensitive: true
    Special rules: none
    Reserved words: none
```

```
Max  Repl Rem
```

```
report_name_rules
1890
```


Rules	Type	Len	Char	Chrs	Prefix	Allowed Chars
Port Rules	16	'*'	no	P	No restrictions	
Cell Rules	16	'*'	no	U	No restrictions	
Net Rules	16	'*'	no	N	No restrictions	

The following example reports all currently defined name rules. Three sets of name rules are defined as EXAMPLE, MAPPING, and UPPER_ONLY.

prompt> **report_name_rules**

```
*****
Report : name_rules
Name Rules : All
Version: v3.1
Date   : Thu Jan 21 11:03:57 1993
*****
```

Rules Name: EXAMPLE

```
Equal port and net names: false
Collapse name space: false
Case insensitive: true
Special rules: none
Reserved words: none
```

	Max	Repl	Rem		
Rules	Type	Len	Char	Chrs	Prefix
Port Rules	16	'*'	no	P	No restrictions
Cell Rules	16	'*'	no	U	No restrictions
Net Rules	16	'*'	no	N	No restrictions

Rules Name: MAPPING

```
Equal port and net names: false
Collapse name space: false
Case insensitive: false
Special rules: none
Reserved words: none
```

	Max	Repl	Rem		
Rules	Type	Len	Char	Chrs	Prefix
Port Rules	none	'_'	no	P	No restrictions
					Mapping String: "{{"_reg","reg"}}"
Cell Rules	none	'_'	no	U	No restrictions
					Mapping String: "{{"_reg","reg"}}"
Net Rules	none	'_'	no	N	No restrictions
					Mapping String: "{{"_reg","reg"}}"

Rules Name: UPPER_ONLY

```
Equal port and net names: false
Collapse name space: true
Case insensitive: false
Special rules: none
```

Reserved words: none

		Max	Repl	Rem			
Rules	Type	Len	Char	Chrs	Prefix	Allowed	Chars

Port	Rules	none	'_'	no	P	Use	"A-Z_"
Cell	Rules	none	'_'	no	U	Use	"A-Z_"
Net	Rules	none	'_'	no	N	Use	"A-Z_"

SEE ALSO

change_names(2)
define_name_rules(2)
report_names(2)

report_names

Reports potential name changes of ports, cells, and nets in a design.

SYNTAX

```
int report_names
[-rules name_rules]
[-hierarchy]
[-dont_touch designs_list]
[-nosplit]
[-original]
```

Data Types

<i>name_rules</i>	string
<i>designs_list</i>	list

ARGUMENTS

- rules *name_rules*
Specifies a set of name rules to which names must conform. The *name_rules* must be defined with the **define_name_rules** command. By default, the name rules file specified in **default_name_rules** is used.
- hierarchy
Reports all names in the design hierarchy. By default, only objects within the current design are reported.
- dont_touch *designs_list*
Specifies a list of designs to which the **changes_names** command is not applied.
- nosplit
Prevents line splitting when column fields overflow.
- original
Reports original names only, and does not report changes.

DESCRIPTION

The **report_names** command reports names of ports, cells, and nets that would be changed to conform to the specified name rules.

This command may not report all of the name changes that would occur by the **change_names** command.

The **report_names** is normally used before running the **change_names** command. The output of **report_names** can be redirected to a names file and used as input to **change_names**.

The *name_rules* file specifies the rules for modifying names, and must be defined with **define_name_rules**.

Use **report_name_rules** to display a list of name rules currently available. Refer to the **define_name_rules** man page for information about naming rules that can be affected when running **report_names**.

With no options specified, **report_names** operates on ports, cells, and nets in the current design. When **-hierarchy** is specified, the report is expanded to include all design objects within the current design object hierarchy.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports port, cell, and net names in the current design that do not conform to the rules defined in **default_name_rules**:

```
prompt> list default_name_rules
default_name_rules = "EXAMPLE"
```

```
prompt> report_names
```

```
*****
Report : names
        -rules EXAMPLE
Design : TOP
Version: v3.0
Date   : Tue Aug 13 14:24:23 1991
*****
```

Design	Type	Object	New Name
TOP	cell	U\$1	U_1
TOP	net	NET_NAME_IS_WAY_TOO_LONG	NET_NAME_IS_WAY_TOO
TOP	net	12345	N12345

```
prompt> change_names -names_file TOP.names
```

Information: 15 names changed using names file 'TOP.names'.

SEE ALSO

```
change_names(2)
define_name_rules(2)
report_name_rules(2)
```

report_net

Reports net information for the design of the current instance or for the current design.

SYNTAX

```
status report_net
[-nosplit]
[-noflat]
[-transition_times]
[-only_physical]
[-verbose]
[-cell_degradation]
[-min]
[-connections]
[-physical]
[net_list]
[-significant_digits digits]
[-max_toggle_rate]
[-scenario scenario_list]
```

Data Types

<i>net_list</i>	list
<i>digits</i>	integer
<i>scenario_list</i>	list

ARGUMENTS

-nosplit

Prevents line splitting and facilitates writing software to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds its column's width, the next field begins on a new line, starting in the correct column.

-noflat

Allows backward compatibility to produce net reports that only trace fanin and fanout for the current level of hierarchy in a design. The **report_net** command, by default, displays fanin and fanout information throughout the hierarchy for each net, as if the net were flattened. Earlier versions of **report_net** report the fanin and fanout for a net only at the current level of hierarchy.

-transition_times

Reports the rise and fall transition times for each net at the end of the report.

-only_physical

Reports the preroutes of the physical net.

-verbose
 Reports verbose connection information when used with the **-connections** option. When used with the **-physical** or **-only_physical** options, this option displays detailed physical information.

-cell_degradation
 Reports the capacitance on the net, the limit for the capacitance on the net (calculated from the cell_degradation table for the drivers of the net), and violations on the net at the end of the report.

-min
 Reports the minimum value of capacitance, resistance, or transition time instead of the maximum. By default, **report_net** displays only the maximum values.

-connections
 Reports information about pins connected to the nets.

-physical
 Reports the total wire length of the net and preroutes of the net, if preroute information exists.

net_list
 Reports only those nets specified in *net_list*. If you do not specify *net_list*, all nets in the current instance are displayed.

-significant_digits *digits*
 Specifies the number of digits to the right of the decimal point to report. Allowed values are 0 through 13. The default value is 2. Using this option overrides the value set by the **report_default_significant_digits** variable.

-max_toggle_rate
 Shows the maximum toggle rate values for each net.

-scenario *scenario_list*
 Reports the timing derate for the specified list of scenarios in a multi-scenario design. Inactive scenarios are skipped in the report. Each scenario is reported separately. If this option is not specified, only the current scenario is reported.

DESCRIPTION

The **report_net** command displays information about the nets in the design of the current instance or in the current design. If **current_instance** is set, the report is generated for the design of that instance. Otherwise, the report is generated for the current design.

The Load column is calculated as $\text{Load} = \text{Cell\_input\_cap} + \text{Wireload\_cap}$. Attributes such as **dont_touch** and **annotated_capacitance** are displayed for each net. Note that **dont_touch** can be present on a net as an implicit attribute. This may occur when the **set_dont_touch_network** command is used. All of the nets in the transitive fanout of a port are affected, but **dont_touch** cannot be removed independently from these nets. The annotated capacitance is the value set by the **set_load** command on a net.

If you specify the **-connections** option, the leaf cell pins connected to each net are listed.

Multicorner-Multimode Support

By default, this command uses information from the current scenario. You can select different scenarios by using the **-scenario** option.

EXAMPLES

The following is an example of a net report:

```
prompt> report_net
```

```
*****
Report : net
Design : CONTROL
Version: v3.1a
Date   : Fri 1993
*****
```

```
Operating Conditions: WCCOM
Wire Loading Model:   05x05
```

Attributes:

```
  d - dont_touch
  c - annotated capacitance
```

Net	Fanout	Fanin	Load	Pins	Attributes
ACCUMCL	1	1	0.39	2	
ACSEL0	1	1	0.39	2	
ACSEL1	1	1	0.39	2	
ACSEL2	1	1	0.39	2	d
BL0	0	2	0.39	2	
BL1	0	2	0.39	2	
BL2	0	2	0.39	2	
BL3	0	2	0.39	2	d
BOTCHAIN	1	1	1.39	2	
CHAININ	2	1	7.82	3	
CHAINOUT	1	1	0.39	2	
CLK	3	1	2.76	4	
DIN0	1	1	1.39	2	c
DIN2	1	1	0.00	2	
DIN3	1	1	0.00	2	
TOPCHAIN	1	1	1.39	2	
WEABLE	1	1	0.39	2	
ZEROIN	1	1	0.39	2	
n	2	2	2.32	4	

Total 50 nets	66	52	100.71	118	
Maximum	9	2	11.12	10	

Average	1.32	1.04	2.01	2.36

The following example shows a net connection report for the net named *t*:

```
prompt> report_net -connections {t}

*****
Report : net
        -connections
Design : counter
Version: v3.1a
Date   : Fri 1993
*****
```

Connections for net 't':

Driver Pins	Type
-----	-----
t/Z	Output Pin (EO)
Load Pins	Type
-----	-----
t_bar/A	Input Pin (IVA)
zero/A	Input Pin (OR2)
a/C	Input Pin (AO2)

The following example shows a prerouted physical net report for the *t* net. It has one segment on metal m2 and a via at (900,281.70).

```
prompt> report_net -only_physical

*****
Report : net
Design : address
Version: 1999.10-PSYN1.2
Date   : Tue Apr 18 22:24:57 2000
*****
```

Net	Layer/Via	Special_Width	Start_X	Start_Y	End_X	End_Y
-----	-----	-----	-----	-----	-----	-----
t	5(m2)	10.00	202.40	-279.90	202.40	281.70
t	4(via45)	*	900.00	281.70	*	*

SEE ALSO

```
current_design(2)
current_instance(2)
help(2)
report_cell(2)
report_constraint(2)
report_design(2)
report_internal_loads(2)
set_dont_touch_network(2)
```


report_default_significant_digits(3)

report_net_changes

Reports net changes that occurred during some IC Compiler optimizations, such as `place_opt`, `create_buffer_tree`, and `route_opt`.

SYNTAX

```
int report_net_changes  
[-verbose]
```

DESCRIPTION

The IC Compiler **report_net_changes** command provides a methodology for the user to identify net changes between the post-synthesis netlist (PSL) and the post-layout netlist (PLN) for the purpose of performing ECO changes and other tasks.

The PSL is usually the output of Design Compiler and input to IC Compiler. The PLN is the final or an intermediate result of IC Compiler.

The environment variable **icc_track_net_changes** is FALSE by default. When the user sets it true, net names will be preserved during IC Compiler optimization processes as much as possible: **clock_opt**, **route_opt**, **place_opt**, **balance_inter_clock_delay**, **create_buffer_tree**, etc. Net name creations during these optimizations will be "tracked" so that the user can report the created nets later. Deleted nets will not be tracked or reported.

Nets derived by optimization processes (such as buffering) will be given names with the same prefix as the original net. Nets touching hierarchical boundaries will not be renamed, as they must match the boundary name while exporting the design.

report_net_changes will report on the "tracked" net name changes to help the user find the collections of nets that are derived from an original net. Nets that are reported together are logically equivalent to the original net.

LIMITATIONS

Setting **icc_track_net_changes** TRUE does not ensure that net names in the IC Compiler result are necessarily equivalent to the nets in the original PSL (input to IC Compiler), because various commands and flows can change the names and functions of nets in ways that cannot be tracked.

Examples of changes that cannot be tracked are:

-- Netlist editing and other commands, such as **change_names**: these commands may seriously disrupt the name correspondence between nets in the post-synthesis netlist and nets in the post-layout netlist. After re-connecting nets with **disconnect_net** and **connect_net**, the net names may have different logical functions: thus net name equivalence does not (in general) imply functional equivalence between the PSN and the PLN.

-- 3rd-party flows with unknown transformations and name changes are not tracked. The **icc_track_net_changes** methodology does not ensure that net names have the same function over time. If a netlist is imported to IC Compiler from a 3rd party tool,

it is a new starting point and net name tracking begins from that point. There is no way currently to re-introduce net tracking information if the design is exported to a 3rd-party tool and then re-imported to IC Compiler. Verilog files do not contain any tracking information. No name tracking will be present immediately after reading a Verilog file into ICC. If the user outputs Verilog and then rereads the Verilog file, the net name tracking will be lost.

-- There is currently no command to re-compute or verify functional correspondence based on net names.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

report_net_changes reports on the nets that are derived from one original net. The report only mentions nets that have been changed. Nets with no changes are not mentioned. Some comments (after <==) are included to explain the behavior during this example where the only changes to the netlist are due to a hierarchical net being buffered.

```
prompt> create_buffer_tree TGT_RESETJ
```

```
1
```

```
prompt> report_net_changes
```

```
Original net:  TGT_RESETJ
```

```
<== High fanout synthesis net
```

```
Changed to:
```

```
  TGT_RESETJ
```

```
  TGT_RESETJ_1
```

```
<== Maintains prefix of root net
```

```
  TGT_RESETJ_2  (inverted)
```

```
<== Indicates inversion of original net.
```

```
  ASYNC/TGTRLEN/RESETJ
```

```
  ASYNC/TGT_RESETJ_hfs_netlink_8
```

```
  ASYNC/S2T/RESETJ
```

```
<== Original hierarchical port net reuses name
```

```
  ASYNC/S2T/RESETJ_hfs_netlink_38
```

```
<== New hierarchical port net unique
```

```
  ASYNC/S2T/RESETJ_hfs_netlink_39
```

```
1
```

SEE ALSO

change_names(2)

report_net_fanout

Displays net fanout or buffer tree information for the current design.

SYNTAX

```
status report_net_fanout
[-nosplit]
[-high_fanout]
[-threshold lower]
[-bound upper]
[-verbose]
[-connections]
[-physical]
[-min]
[-tree [-depth level]]
[net_list]
```

Data Types

<i>lower</i>	integer
<i>upper</i>	integer
<i>level</i>	integer
<i>net_list</i>	list

ARGUMENTS

-nosplit

Prevents line-splitting and facilitates writing software to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds its column's width, the next field begins on a new line in the same column.

-high_fanout

Specifies to show high fanout nets only. A high fanout net is a net with more fanouts than **500**. A high fanout buffer tree is a buffer tree with more leaf loads than the specified value. The fanouts cross hierarchies. This option is mutually exclusive with the **-threshold** option.

-threshold *lower*

Specifies to only show nets with more fanout than *lower*. The fanouts cross hierarchies. This option is mutually exclusive with the **-high_fanout** option.

-bound *upper*

Specifies to only show nets with fanout less than or equal to *upper*. The fanouts cross hierarchies.

-verbose

Displays verbose information.

-connections

Displays information about pins connected to the nets.

report_net_fanout

-physical
Displays location information when applicable.

-min
Shows the minimum conditions. By default, only maximum conditions are shown.

-tree
Indicates to treat a buffer tree as transparent. The leaf loads of the buffer tree are treated as the fanouts of the net. Hierarchical boundaries are not considered as leaf loads.
This option is mutually exclusive with the *net_list* argument.

-depth level
Indicates to only display buffer trees with more levels than *level*. This option can only be used with the **-tree** option.

net_list
Specifies to process only those nets on the *net_list*. If *net_list* is not specified, all nets in the current instance are processed.
This argument is mutually exclusive with the **-tree** option.

DESCRIPTION

The **report_net_fanout** command displays net fanout or buffer tree information in the current design. If a *net_list* is specified, the report is generated only for the specified nets.

All reports cross hierarchical boundaries as if the nets were flattened. This is consistent with the **report_net** command.

The **report_net_fanout** command can be used to find high fanout nets, buffer trees, or long buffer chains in the current design or the current instance.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows how to find high fanout nets:

```
prompt> report_net_fanout -high_fanout
```

The following example shows how to find high fanout nets in a particular set of nets:

```
prompt> report_net_fanout -high_fanout [get_nets scan*]
```

The following example shows buffer trees with 10 leaf pins:

```
prompt> report_net_fanout -tree -threshold 9 -bound 10
```

The following example shows how to find long buffer chains.

```
prompt> report_net_fanout -tree -depth 8 -bound 1
```

SEE ALSO

```
create_buffer_tree(2)  
current_design(2)  
current_instance(2)  
remove_buffer_tree(2)  
report_buffer_tree(2)
```

report_net_routing_layer_constraints

Reports the routing layer constraints for the specified nets.

SYNTAX

```
status report_net_routing_layer_constraints  
nets  
[-output file_name]
```

Data Types

<i>nets</i>	collection
<i>file_name</i>	string

ARGUMENTS

nets
Specifies the nets for which to report routing layer constraints.
This is a required option.

-output *file_name*
Generates a Tcl script that contains the **set_net_routing_layer_constraints** commands used to set the routing layer constraints for the specified nets.
You can use this file to set the same constraints in another session.

DESCRIPTION

This command reports the routing layer constraints for the specified nets.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports the routing layer constraints for a net named netA.

```
prompt> report_net_routing_layer_constraints {netA}
```

SEE ALSO

```
remove_net_routing_layer_constraints(2)  
route_opt(2)  
set_net_routing_layer_constraints(2)
```

report_net_routing_rules

Reports the nondefault routing rules for the specified nets.

SYNTAX

```
status report_net_routing_rules  
nets  
[-freeze]  
[-minorchange]  
[-normal]  
[-output file_name]
```

Data Types

<i>nets</i>	collection
<i>file_name</i>	string

ARGUMENTS

nets
Specifies the nets for which to report nondefault routing rules.
This is a required argument.

-freeze
Reports the specified nets with the **freeze** attribute.

-minorchange
Reports the specified nets with the **minorchange** attribute.

-normal
Reports the specified nets with the **normal** attribute.

-output *file_name*
Generates a Tcl script with the specified file name that contains **set_net_routing_rule** commands for the current net routing rule settings of the specified nets. You can use this script to set the same net routing rules in another session.

DESCRIPTION

This command lists the nondefault routing rules for the specified nets. In addition, you can generate a Tcl script that contains **set_net_routing_rule** commands for the current net routing rule settings of the specified nets.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports the nondefault routing rules for the nets named CLK1 and CLK2.

```
prompt> report_net_routing_rules {CLK1 CLK2}

*****
Report : net routing rules
Design : super_watch
Version: C-2009.06
Date   : Fri May  8 21:39:28 2009
*****

----- Nets with non-default routing rules -----
CLK1 :   SP1
CLK2 :   SP1

----- All other nets use DEFAULT routing rule -----
-----
```

The following example reports the nondefault routing rules for the nets named CLK3 and CLK4.

```
prompt> report_net_routing_rules {CLK3 CLK4}

*****
Report : net routing rules
Design : super_watch
Version: C-2009.06
Date   : Fri May  8 21:40:28 2009
*****

----- All nets use DEFAULT routing rule -----
-----
```

The following example reports the nondefault routing rules for all nets.

```
prompt> report_net_routing_rules [get_nets -hier]

*****
Report : net routing rules
Design : super_watch
Version: C-2009.06
Date   : Fri May  8 21:41:28 2009
*****

----- Nets with non-default routing rules -----
CLK1 :   SP1
CLK2 :   SP1

----- All other nets use DEFAULT routing rule -----
-----
```

The following example reports nets with the **freeze**, **minorchange**, and **normal**

attributes.

```
prompt> report_net_routing_rules [get_nets io_c2*] \  
-freeze -minorchange -normal
```

```
*****
```

Report : net routing rules

```
-freeze  
-minorchange  
-normal
```

Design : super_watch

Version: C-2009.06

Date : Fri May 8 21:42:28 2009

```
*****
```

----- Nets with freeze attribute -----

```
io_c2[3]  
io_c2[0]
```

----- Nets with minorchange attribute -----

```
io_c2[4]  
io_c2[1]
```

----- Nets with normal attribute -----

```
io_c2[6]  
io_c2[11]  
io_c2[15]  
io_c2[12]  
io_c2[5]  
io_c2[7]  
io_c2[8]  
io_c2[2]  
io_c2[13]  
io_c2[10]  
io_c2[14]  
io_c2[9]
```

SEE ALSO

set_net_routing_rule(2)

report_noise

Reports static noise for the worst pins or the specified pins, or all violators.

SYNTAX

```
status report_noise
[-nworst_pins worst_pin_count]
[names]
[-slack_lesser_than slack_limit]
[-all_violators]
[-verbose]
[-significant_digits digits]
[-slack_type slack_type]
```

Data Types

<i>worst_pin_count</i>	int
<i>names</i>	string
<i>slack_limit</i>	float
<i>digits</i>	int

ARGUMENTS

-nworst_pins *worst_pin_count*
Specifies the number of load pins to be reported. Any number greater than 1 is accepted; the default value is 1.

names
Lists pin names. By default, this option is off.

-slack_lesser_than *slack_limit*
Indicates that only those pins with a slack less (more negative) than *slack_limit* are to be shown.

-all_violators
Indicates that only violating pins (negative slack) are to be shown. This option cannot be used with the **-slack_lesser_than** option. If this option is used with the **-nworst_pins** option, the number of violating pins will be limited by that value.

-verbose
Shows more details about the calculation of total noise on each load pin, including the individual contribution of each aggressor .

-significant_digits *digits*
Specifies the number of digits after the decimal point to be displayed for time values in the generated report, default value is 2.

-slack_type *slack_type*
Specifies the type of slack to be used. Valid values are area, height. A *slack_type* of area reports slack as the voltage margin multiplied by the noise bump width. The voltage margin is defined by the noise bump height and noise

immunity curves. This setting is the default. A *slack_type* of height reports noise slack as the voltage margin.

DESCRIPTION

This command reports static noise (width, height, and slack) for the worst pins or the specified list of pins. It computes "between rail noise," which is above low and below high noise. It uses slack type height or area.

To properly use the **report_noise** command in ICC, you must enable static noise by setting The **set_si_options** command option **-static_noise** to **true**.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example reports the static noise values for the **5** worst pins.

```
prompt> report_noise -nworst_pins 5
```

```
*****
Report : noise
        -nworst_pins 5
Design : Xtalk_test
Version: A-2007.12-ICC-BETA2-1
Date   : Thu Sep 27 18:17:14 2007
*****
```

noise_region: above_low

pin name	width	height	slack
U954/B	1.091	0.290	0.082
U81352/I	0.808	0.250	0.124
U71379/B1	0.808	0.250	0.142
U8767/I	1.530	0.189	0.178
U7775/B1	1.530	0.189	0.186

noise_region: below_high

pin name	width	height	slack
U8767/I	2.191	0.407	0.035
U7775/B1	2.191	0.407	0.055
U81352/I	1.113	0.357	0.094
REG_DATA[4]	1.443	0.392	0.108
U13260/A1	2.191	0.408	0.119

The following example reports the detailed noise information for the load pin X3[11]

```
prompt> report_noise X3[11] -verbose -sign 3 -slack_type height
```

```
report_noise
1910
```

```

*****
Report : noise
        -significant_digits 3
        -slack_type height
        -verbose
Design : Xtalk_test2
Version: A-2007.12-ICC-BETA2-1
Date   : Thu Sep 27 19:17:14 2007
*****

```

slack type: height

noise_region: above_low

pin name	width	height	slack

X3[11]			
Aggressors:			
A2[3]	4.446	0.057	
G1[2]	4.199	0.031	
n1168	0.000	0.000	
Y1[27]	0.000	0.000	
X3[23]	0.000	0.000	
n2525	4.687	0.019	
n14489	3.381	0.102	
n14572	3.407	0.016	
W12[16]	4.372	0.071	
W12[24]	4.507	0.059	
Total:	2.809	0.355	-0.005

noise_region: below_high

pin name	width	height	slack

X3[11]			
Aggressors:			
A2[3]	8.487	0.080	
G1[2]	8.249	0.043	
n1168	0.000	0.000	
Y1[27]	0.000	0.000	
X3[23]	0.000	0.000	
n2525	8.613	0.028	
n14489	7.678	0.121	
n14572	7.702	0.019	
W12[16]	8.372	0.101	
W12[24]	8.461	0.085	
Total:	5.554	0.477	-0.127

1

SEE ALSO

```

set_si_options(2)
report_noise_calculation(2)

```

report_noise_calculation

Displays the actual calculation of noise information for the specified net arc.

SYNTAX

```
status report_noise_calculation  
[-significant_digits digits]  
-from from_pin  
-to to_pin
```

Data Types

digits int

ARGUMENTS

-significant_digits *digits*
 Specifies the number of digits after the decimal point to be displayed for the values in the generated report, default value is 2. This option controls only the number of digits displayed, not the precision used internally for analysis.

-from *from_pin* -to *to_pin*
 Specifies the start and end points of a net arc within a design. The *from_pin* must be a driver pin or port, and the *to_pin* must be the load pin or port of the same net.

DESCRIPTION

Displays details of the noise calculation for the specified net arc. The report shows some general information about the victim, as well as details of noise bump and noise slack calculations.

The noise calculation section includes the noise bumps induced from each of the aggressors. The aggressor attribute column shows more details about each of the effective aggressors. Aggressors which are filtered are not included in this report. The following is the list of possible attributes an aggressor can have:

Attributes: A - aggressor is active S - aggressor is screened

Active aggressor is an aggressor which contributes to the worst case alignment scenario of all the aggressors.

Aggressor can be screened from the analysis if it has no significant impact on the victim.

The noise slack calculation section shows the details of how the area slack and height slack is derived for the victim, and also what kind of constraint was used to derive the slack. The height slack is the difference of the actual bump height versus the bump height which causes the failure with the same width. The area slack is the area that needs to be added to the actual bump to cause a noise failure.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example specifies a noise calculation report for the driver pin exetop0/e_dptop0/e_flag0/I27/Y and load pin exetop0/e_dptop0/e_flag0/U1/A :

Report : noise calculation
-from exetop0/e_dptop0/e_flag0/I27/Y
-to exetop0/e_dptop0/e_flag0/U1/A

Design : Xtalk_test
Version: A-2007.12-ICC-INTERNAL
Date : Thu Jul 19 15:43:20 2007

Analysis mode	: report_at_source
Region	: above_low
Victim driver pin	: exetop0/e_dptop0/e_flag0/I27/Y
Victim driver library cell	: MX4X4
Victim net	: exetop0/e_dptop0/e_flag0/N194
Steady state resistance source	: estimation set value
Driver voltage swing	: 1.620

Attributes:

A - aggressor is Active
S - aggressor is screened

	Height	Width	Area	Aggressor Attributes

Aggressors:				
exetop0/e_rndm0/n35	0.000	0.000	0.000	S
exetop0/e_rndm0/n10	0.000	0.000	0.000	S
exetop0/e_rndm0/n19178	0.000	0.000	0.000	S
exetop0/e_rndm0/n18454	0.019	0.238	0.002	A
exetop0/s_AEI2_0_	0.000	0.000	0.000	S
Total:	0.019	0.238	0.002	

Noise slack calculation:

Constraint type: user margin

	Height	Area	

Required	0.567	(0.567	* 0.238) -
Actual	0.019	(0.019	* 0.238)

```

-----
Slack                      0.548                      0.131

Analysis mode              : report_at_source
Region                    : below_high
Victim driver pin         : exetop0/e_dptop0/e_flag0/I27/Y
Victim driver library cell : MX4X4
Victim net                 : exetop0/e_dptop0/e_flag0/N194

Steady state resistance source : estimation set value
Driver voltage swing          : 1.620

```

Attributes:

```

  A - aggressor is Active
  S - aggressor is screened

```

	Height	Width	Area	Aggressor Attributes

Aggressors:				
exetop0/e_rndm0/n35				
	0.028	0.843	0.012	A
exetop0/e_rndm0/n10				
	0.025	0.702	0.009	A
exetop0/e_rndm0/n19178				
	0.000	0.000	0.000	S
exetop0/e_rndm0/n18454				
	0.029	0.282	0.004	A
exetop0/s_AEI2_0_				
	0.000	0.000	0.000	S
Total:	0.082	0.603	0.025	

Noise slack calculation:

Constraint type: user margin

	Height	Area

Required	0.567	(0.567 * 0.603) -
Actual	0.082	(0.082 * 0.603)

Slack	0.485	0.293

SEE ALSO

```

set_si_options(2)
report_noise(2)

```


report_operating_conditions

Display a specific or all the operating conditions in a library.

SYNTAX

```
int report_operating_conditions  
-library library_name  
[-name op_cond_name]
```

Data Types

<i>library_name</i>	string
<i>op_cond_name</i>	string

ARGUMENTS

-library *library_name*
Specifies the name of the library that stores the operating conditions.

-name *op_cond_name*
Specifies the name of the operating conditions.

DESCRIPTION

The **report_operating_conditions** command reports a specific or all the operating conditions in the specified library.

To create a new operating conditions, use **create_operating_conditions**.

To set operating conditions on the current design, use **set_operating_conditions**.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports a specific operating conditions called "BEST" in the library "a_lib".

```
prompt> report_operating_conditions -library a_lib -name BEST
```

The following example reports all the operating conditions in the library "IBM_CMOS5S6_SC".

```
prompt> report_operating_conditions -library IBM_CMOS5S6_SC
```

SEE ALSO

`create_operating_conditions(2)`
`set_operating_conditions(2)`

report_optimize_dft_options

Reports options for physical design-for-test (DFT) optimization.

SYNTAX

status **report_optimize_dft_options**

ARGUMENTS

The **report_optimize_dft_options** command has no argument.

DESCRIPTION

This command reports the value of the physical DFT options available in command **set_optimize_dft_options**. The options are **-repartitioning_method** and **-single_dir_option**.

SEE ALSO

optimize_dft(2)
set_optimize_dft_options(2)

report_optimize_pre_cts_power_options

Reports the option settings for **optimize_pre_cts_power** command.

SYNTAX

status_value **report_optimize_pre_cts_power_options**

ARGUMENTS

The **report_optimize_pre_cts_power_options** command has no arguments.

DESCRIPTION

This command reports all the option values for the **optimize_pre_cts_power** command. The main options to enable pre-CTS power optimizations are -low_power_placement and -clock_gating in **set_power_options** command. All other options are set by the **set_optimize_pre_cts_power_options** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports the current options to be used by the **optimize_pre_cts_power**.

```
prompt> report_optimize_pre_cts_power_options
```

SEE ALSO

```
optimize_pre_cts_power(2)  
set_optimize_pre_cts_power_options(2)  
set_power_options(2)
```

report_parameter

Reports the details of a parameter, such as name, type, current value, default value, valid range, and a brief description.

SYNTAX

```
int report_parameter
[-name name]
[-module route | groute | droute | trackAssign | ek | preroute | sr | all]
[-type integer | real | string]
```

Data Types

name string

ARGUMENTS

-name *name*
Specifies the name of the parameter to be reported.

-module route | groute | droute | trackAssign | ek | preroute | sr | all

| ek | preroute | sr | all
Specifies the module for which to report parameters. The valid module names are as follows:

route
groute
droute
trackAssign
ek
preroute
sr
all
By default, or if you use this option with the **all** argument, the tool reports the parameters for all of the parameters in all of the modules supported by the **report_parameter** command.

-type integer | real | string
Specifies the parameter type.

DESCRIPTION

This command reports the details of a parameter, such as name, type, current value, default value, valid range, and a brief description. If you do not specify any arguments, it reports all of the parameters in all of the supported modules.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports the details of the *timingdriven* parameter:

```
prompt> report_parameter -name timingdriven
```

The following example reports the details of parameters in the **groute** module:

```
prompt> report_parameter -module groute
```

The following example reports the details on all of the parameters in all of the modules supported by the **report_parameters** command:

```
prompt> report_parameter
```

SEE ALSO

`set_parameter(2)`

report_path_group

Reports information about path groups in the current design.

SYNTAX

```
status report_path_group  
[-nosplit]  
[-expanded]  
[-scenario scenario_list]
```

ARGUMENTS

-nosplit

Prevents line splitting and facilitates writing software to extract information from the report output. Most design information is listed in fixed-width columns. When the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

-expanded

Expands the paths when reporting.

-scenario *scenario_list*

Reports path groups for given list of scenarios of a multiscenario design. Inactive scenarios will be skipped in the report. Each scenario is reported separately. If this option is not given, only the current scenario is reported.

DESCRIPTION

The **report_path_group** command produces a report showing information about path groups in the current design. The report includes path groups automatically created by the **create_clock** command and groups manually created with the **group_path** command. Path groups are used to affect the calculation of maximum delay cost during optimization. You can display the cost of each group using the **report_constraint** command.

To remove all path groups in the current design, use the **reset_design** command. To remove certain paths from their current groups, use **group_path -default**.

Multicorner-Multimode Support

By default, this command uses information from the current scenario. You can select different scenarios by using the **-scenario** option.

EXAMPLES

The following example reports all timing attributes set on the design:

```
prompt> report_path_group
```

```

*****
Report : path_group
Design : counter
Version: 1998.01
Date   : Wed Jul  9 13:51:14 1997
*****

```

Group Name	Weight	Range	

default	1.00	0.00	
CLK	1.00	0.00	
BLUE	1.00	5.00	
Paths:			
From	Through	To	Group Name

*	*	CLK	CLK
Red_reg[5]	*	Blue_reg[5]	BLUE

SEE ALSO

```

create_clock(2)
current_design(2)
group_path(2)
report_constraint(2)
reset_design(2)

```


report_pg_net

Reports power and ground net information for the opened Milkyway design.

SYNTAX

```
status report_pg_net
[-net net_list]
[-connections]
```

Data Types

net_list list

ARGUMENTS

-net *net_list*
Reports only those nets specified in **-net** option. If you do not specify **-net** option, all power and ground nets are reported.

-connections
Reports information about pins connected to the power and ground nets. By default, this option is set as false.

DESCRIPTION

The **report_pg_net** command displays information about the power and ground nets of the current opened Milkyway design. By default, will report all the power and ground nets in all hierarchical level of the design. Option **"-net"** can be used to report only specified power and ground nets. **"-net"** option can take a collection of power and ground nets, the collection can be generated by **get_nets** command with **"-all"** option for power and ground net selection.

For each net, will report net name, net type which can be either "power" or "ground", UPF power domain name of the net if available, number of pins and port which connected to the net, and the detail list of connected pins and port if option **"-connections"** specified.

if option **"-connections"** is specified, will report all pins and port which connected to the net.

.SH EXAMPLES The following is an example of report_pg_net output:

```
prompt> report_pg_net -net [get_nets -physical {VDD*}]
```

```
***** Report: pg net Options: -net Design: top
Version: B-2008.09-ICC-SP3 Date: Thu Nov 6 15:34:10 2008
*****
```

```
Net Type Power Domain Num connections -----
----- VDD1 power 2 modA/VDD1 power
hier_domain 6 -----
```

----- total 2 nets

1

.SH "SEE ALSO" get_nets(2)

report_physical_bus

Displays information about physical bus.

SYNTAX

```
status report_physical_bus  
[physical_bus_list]
```

Data Types

physical_bus_list list

ARGUMENTS

physical_bus_list
Specifies the list of physical buses to be reported.
You can specify element of the list with a name pattern of physical buses or a collection containing physical buses.
By default, all physical buses in the current design are reported.

DESCRIPTION

This command displays information about the specified physical buses.

In the report, nets in a physical bus are listed following the order in that physical bus.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates a new physical bus then reports it with the command **report_physical_bus**.

```
prompt> create_physical_bus -sort ascending -nets [get_nets UPC_DATA*] bus1  
{bus1}
```

```
prompt> report_physical_bus bus1
```

```
*****  
Report : physical_bus  
Design : CORE  
Version: B-2008.09-ICC  
Date   : Sun Apr  6 07:20:27 2008  
*****
```

```
Physical Bus  Net  
-----
```

bus1	UPC_DATA[0]	UPC_DATA[1]	UPC_DATA[2]
	UPC_DATA[3]	UPC_DATA[4]	UPC_DATA[5]
	UPC_DATA[6]	UPC_DATA[7]	UPC_DATA[8]
	UPC_DATA[9]	UPC_DATA[10]	UPC_DATA[11]

1

SEE ALSO

`create_physical_bus(2)`
`get_physical_buses(2)`
`update_physical_bus(2)`
`remove_physical_bus(2)`

report_physical_signoff_options

Reports option values set by set_physical_signoff_options that are shared (in common) by signoff_drc command and signoff_metal_fill command, etc.

SYNTAX

```
status report_physical_signoff_options
[-default]
```

ARGUMENTS

-default
Report default options of set_physical_signoff_options.

DESCRIPTION

This command reports option values that have been set by set_physical_signoff_options command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example runs the **report_physical_signoff_options** command.

```
prompt> set_physical_signoff_options \
-exec_cmd icv \
-dp_hosts {machine_1 machine_2 machine_3 machine_4} \
-num_cpus 4 \
-drc_runset drc.rs \
-fill_runset fill.rs \
-mapfile layermap
```

```
prompt> report_physical_signoff_options
```

```
*****
Report : physical_signoff_options
Design : my_design
Version: C-2009.06-ICC
Date   : Tue Feb 17 15:55:58 2009
*****
```

Current physical signoff options

```
-----
      exec_cmd = icv
      drc_runset = drc.rs
      fill_runset = fill.rs
```

```
mapfile = layermap
dp_hosts = machine_1 machine_2 machine_3 machine_4
num_cpus = 4
```

SEE ALSO

```
signoff_drc(2)
signoff_metal_fill(2)
set_physical_signoff_options(2)
```

report_pin_guides

Displays information about the pin guides.

SYNTAX

```
status report_pin_guides
[-pins pins_collection
  | -nets nets_collection]
[pin_guides]
```

Data Types

<i>pins_collection</i>	collection
<i>nets_collection</i>	collection
<i>pin_guides</i>	collection

ARGUMENTS

-pins *pins_collection*
Specifies the pins for which to report the associated pin guides.

-nets *nets_collection*
Specifies the nets for which to report the associated pin guides.

pin_guides
Specifies the pin guides to report. If you do not specify this option, the command reports all pin guides.

DESCRIPTION

This command reports details about the specified pin guides, the pin guides attached to the specified pins, the pin guides attached to the specified nets, or all pin guides (if no options are specified).

Each plan group pin guide is output in the following format:

```
Pin Guide: name
  BBox: {{xmin ymin} {xmax ymax}}
  PlanGroup: {parent_name}
  Nets: {child_name ...}
```

Each soft macro pin guide is output in the following format:

```
Pin Guide: name
  BBox: {{xmin ymin} {xmax ymax}}
  SoftMacro: {parent_name}
  Pins: {child_name ...}
```

where:

name -

Pin Guide Name
{xmin ymin} {xmax ymax}} -
Bounding Box of the Pin Guide
{parent_name} -
Parent Plan Group or Soft Macro name
{child_name ...} -
List of the names of the pins or nets attached to the pin guide

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports all pin guides attached to the pins A, B, or C.

```
prompt> report_pin_guides -pins [get_pins {A B C}]
```

SEE ALSO

`create_pin_guide(2)`
`get_pin_guides(2)`

report_pin_name_synonym

Reports pin name synonym definitions.

SYNTAX

```
status report_pin_name_synonym
[-nosplit]
```

ARGUMENTS

-nosplit
Prevents line splitting. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

DESCRIPTION

The **report_pin_name_synonym** command displays all of the currently defined pin name synonyms.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLE

The following command set shows the pin name synonym settings and then uses the **report_pin_name_synonym** command to report pin name synonyms:

```
prompt> set_pin_name_synonym SYN_CD CD
1
prompt> set_pin_name_synonym SYN_QN QN
1
prompt> set_pin_name_synonym -full_name \
mid1/bot1/LSR0P_at_bot/SYN_R mid1/bot1/LSR0P_at_bot/R
1
prompt> set_pin_name_synonym -full_name this/is/a/synonym mid1/FJK2SP_at_mid/TI
1
prompt> set_pin_name_synonym -full_name mid1/FJK2SP_at_mid/J mid2/bot2/LSR0P_at_bot/
Q
1
prompt> report_pin_name_synonym

*****
Report : pin name synonym
Design : top
Version: X-2005.09
Date   : Wed Jun 29 10:29:04 2005
*****
```

Synonym	Pin Name	Type

this/is/a/synonym	mid1/FJK2SP_at_mid/	
TI full name		
mid1/FJK2SP_at_mid/J	mid2/bot2/LSR0P_at_bot/	
Q full name		
mid1/bot1/LSR0P_at_bot/SYN_R	mid1/bot1/LSR0P_at_bot/	
R full name		
SYN_QN	QN	simple
name		
SYN_CD	CD	simple
name		

1		

The following example first removes all of the pin name synonyms and the reports the updated information:

```
prompt> remove pin_name_synonym -all
1
prompt> report pin_name_synonym

*****
Report : pin name synonym
Design : top
Version: X-2005.09
Date   : Wed Jun 29 10:29:04 2005
*****

No pin name synonym
1
```

SEE ALSO

```
remove_pin_name_synonym(2)
set_pin_name_synonym(2)
```

report_pin_shape

Displays information about a list of pin shapes.

SYNTAX

```
status report_pin_shape
pin_shape_list
```

Data Types

```
pin_shape_list      list
```

ARGUMENTS

```
pin_shape_list
    Specifies the list of pin shapes that are to be reported.
    You can specify an element of the list with a name pattern of pin shapes or
    a collection containing pin shapes.
```

DESCRIPTION

This command displays information about the specified pin shapes.

EXAMPLES

The following example reports a pin shape tc4/sram_Q[4] using the **report_pin_shape** command.

```
prompt> report_pin_shape tc4/sram_Q[4]
*****
Report : pin_shape
Design : CORE
Version: C-2009.06-ICC
Date   : Sun Dec  6 07:20:27 2008
*****

Pin Shape      Layer      Status      BBox
-----
tc4/sram_Q[4]  M6        Fixed      {{560.000 1707.000} {626.000 1773.000}}
-----
1
```

SEE ALSO

```
get_pin_shapes(2)
pin_shape_attributes(3)
```

report_placement_utilization

Reports the placement utilization for the entire design or a certain region.

SYNTAX

```
status report_placement_utilization  
[-non_fixed_only Boolean]  
[-grid_size Float]  
[-verbose Boolean]  
[-coordinates {X1 Y1 X2 Y2}]
```

ARGUMENTS

-non_fixed_only *Boolean*

Specifies that the utilization reporting will be for "non_fixed standard cell" only. By default, the placement utilization number will be reported for "non_fixed + fixed standard cells". If this option has been specified, the utilization number will be for "non_fixed standard cell only".

The two formulas for calculating the "non_fixed + fixed standard cell" utilization and "non_fixed standard cell only" utilization are different:

a) "non_fixed + fixed" standard cell utilization: The formula is: $(\text{Non-Fixed\_Std\_Cell\_Area} + \text{Fixed\_Std\_Cell\_Area}) / (\text{Total\_Area} - \text{All\_Blockage\_Area})$

Here "Total_Area" is the area of the entire chip or a certain region if option "-coordinate" has been specified. "All Blockage Area" includes "placement blockage", "hard macros" and "complete pnet blockage".

b) "non-Fixed only" standard cell utilization:

The formula is:

$(\text{Non-Fixed\_Std\_Cell\_Area}) / (\text{Total\_Area} - \text{All\_Blockage\_Area} - \text{Fixed\_Std\_Cell\_Area})$

-grid_size *Float*

Specifies the grid size of each "sub-region". Besides the entire chip or a user specified region, this command will also divide the given area (either entire chip or a user specified region if option "-coordinate" has been specified) into a set of "sub-regions" and report the corresponding utilization of the "sub-region". So users may have better understanding for the "evenness" of the cell distribution. The "grid_size" will be the size of each "sub-region". By default, the grid size is 5x placement site heights. The Minimum "grid_size" will be 3x placement site heights. The unit for "grid_size" is in micron.

-verbose *Boolean*

Specifies that the utilization number of each "sub-region" will be reported. By default, only the statistic summary of the "sub-region" utilization will be reported.

DESCRIPTION

The **report_placement_utilization** command reports the utilization number for the entire design or a certain region if option "-coordinate" has been specified.

Besides the utilization number, this command will also report the corresponding "total area", "blockage area", "hard macro area", etc. These types of information will be also useful for the users.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example runs the **report_placement_utilization** command for the entire chip.

```
prompt> report_placement_utilization
```

The following example runs the **report_placement_utilization** command for a user-specified region.

```
prompt> report_placement_utilization -coordinate {20 20 130 130}
```

SEE ALSO

```
legalize_placement(2)  
check_legality(2)  
report_design(2)
```

report_pnet_options

Issues a report of the options set with the **set_pnet_options** command.

SYNTAX

```
int report_pnet_options
```

ARGUMENTS

There are no arguments to this command.

DESCRIPTION

The **report_pnet_options** command issues a report about the options set on the metal layers using the **set_pnet_options** command. The report is written out in table format. The table columns correspond to the different options used with the **set_pnet_options** command. If the Astro legality checker is enabled, then the route types that are considered are reported below the table. If the Astro legality checker is enabled, then the route types that are considered are reported below the table.

For further information, refer to the **set_pnet_options** command man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows a report of the options in table format:

```
prompt> report_pnet_options
```

```
*****
Report : pnet options
Design : A7S_top
Version: V-2004.06-PCXG-ALPHA8
Date   : Mon Mar 29 15:24:11 2004
*****
```

Layer	Blockage	Min_width	Min_height	Via_additive
METAL1	partial	8.64	12.92	via additive
METAL2	partial	8.64	12.92	via additive
METAL3	complete	0.00	12.92	via additive
METAL4	none	---	---	via additive

Route types considered:

```
report_pnet_options
1936
```

```
pg ring, pg strap, pg pin connection, clock ring  
clock strap, path
```

Route types considered:

```
pg ring, pg strap, pg pin connection, clock ring  
clock strap, path
```

SEE ALSO

```
legalize_placement(2)  
remove_pnet_options(2)  
set_pnet_options(2)
```

report_port

Displays information about ports of the current instance or the current design.

SYNTAX

```
integer report_port  
[-drive]  
[-verbose]  
[-physical]  
[-only_physical]  
[-nosplit]  
[-significant_digits digits]  
[port_list]
```

Data Types

<i>digits</i>	integer
<i>port_list</i>	list

ARGUMENTS

- drive
Specifies that the port report is to include only the sections showing the drive capability of input and inout ports. By default, this option is off.
- verbose
Specifies that the port report is to include all port information. By default, only a summary section is shown that lists all ports and their direction.
- physical
Reports the physical location of the port. By default, this option is off.
- only_physical
Reports the physical only ports. By default, this option is off.
- nosplit
Prevents line-splitting and facilitates writing software to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column. By default, this option is off.
- significant_digits *digits*
Specifies the number of digits to the right of the decimal point that the command reports. Using this option overrides the value set by the **report_default_significant_digits** variable. Allowed values are **0** through **13**. The default is **2** for all fields except "Pin Load" and "Wire Load," which have a default of **4**. Specifying a valid value for significant digits overwrites the existing default value.
- port_list*
Indicates that the port report is to include only the specified ports and

cannot be used if the current instance is set. By default, all ports in the current instance or current design are listed.

DESCRIPTION

This command displays information about ports in the design of the current instance or the current design. If the current instance has been set, the report is generated for the design of that instance. Otherwise the report is generated for the current design.

By default, a brief report is produced that includes all ports in the design. If you use the **-verbose** option, all types of information are included. If you use the **-drive** option, only drive information for input and inout ports is shown. If you specify a *port_list* value, the command includes only those ports in the report.

Attributes pertaining to the **set_driving_cell** command that assigned the driving cells to the ports are reported in the "Attrs" column of the driving cell section of the verbose report. The letter **D** indicates that the **-dont_scale** option of the **set_driving_cell** command was used, and the letter **N** indicates that the **-no_design_rule** option was used.

If the current instance has been set, the report is generated for the design of that instance. Note that attributes on these ports have no effect unless the current design is changed to that subdesign.

The "Input Delay" and "Output Delay" portions of the report list the minimum rise, minimum fall, maximum rise, and maximum fall delays. They also list the clock to which the delay is relative. If the clock name is followed by (f), the delay is relative to the falling edge of the clock. Otherwise the delay is relative to the rising edge. If the clock name is followed by (l), input delay is considered as if coming from a level-sensitive latch or output delay is considered as if going to a level-sensitive latch. By default, the delay is relative to a rising edge-triggered device.

The "Max Tran" and "Min Tran" columns list the maximum and minimum rise and fall times set by the **set_transition** command to this port.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example shows a brief port report:

```
prompt> report_port
```

```
*****
Report : port
Design : counter
Version: v3.1
```

Date : Mon 1992

Port	Dir	Pin Load	Wire Load	Max Trans	Max Cap	Connection Class	Attrs
A	in	0.0000	0.0000	--	--	--	
B	in	0.0000	0.0000	--	--	--	
C	in	0.0000	0.0000	--	--	--	
CL	in	0.0000	0.0000	--	--	--	
CLK	in	0.0000	0.0000	--	--	--	
D	in	0.0000	0.0000	--	--	--	
JOKE	in	0.0000	0.0000	--	--	--	
L	in	0.0000	0.0000	--	--	--	
P	in	0.0000	0.0000	--	--	--	
RESET	in	0.0000	0.0000	--	--	--	
T	in	0.0000	0.0000	--	--	--	
CO	out	2.0000	0.0000	--	--	--	
QA	out	2.0000	0.0000	--	--	--	
QB	out	2.0000	0.0000	--	--	--	
QC	out	2.0000	0.0000	--	--	--	
QD	out	2.0000	0.0000	--	--	--	

The following example shows a verbose port report:

prompt> **report_port -verbose**

Report : port

-verbose

Design : s24099

Version: v3.4a-development

Date : Thu Oct 12 11:15:22 1995

Port	Dir	Pin Load	Wire Load	Max Trans	Max Cap	Connection Class	Attrs
in1	in	0.0000	0.0000	--	--	--	
in2	in	0.0000	0.0000	--	--	--	
in3	in	0.0000	0.0000	--	--	--	
in4	in	0.0000	0.0000	--	--	--	
sel1	in	0.0000	0.0000	--	--	--	
sel2	in	0.0000	0.0000	--	--	--	
out1	out	0.0000	0.0000	--	--	--	
out2	out	0.0000	0.0000	--	--	--	
b1	inout	0.0000	0.0000	--	--	--	
b2	inout	0.0000	0.0000	--	--	--	

Port	External Number Points	Fanout Wireload Model
in1	0	--

report_port
1940

in2	0	--
in3	0	--
in4	0	--
sel1	0	--
sel2	0	--
out1	0	--
out2	0	--
b1	0	--
b2	0	--

Input Delay						
Input Port	Min		Max		Related Clock	Max Fanout
	Rise	Fall	Rise	Fall		
in1	--	--	--	--	--	--
in2	--	--	--	--	--	--
in3	--	--	--	--	--	--
in4	--	--	--	--	--	--
sel1	--	--	--	--	--	--
sel2	--	--	--	--	--	--
b1	--	--	--	--	--	--
b2	--	--	--	--	--	--

Driving Cell				
Input Port	Rise	Fall	Mult	Attrs
in1	lsi_10k/AN2/Z	lsi_10k/AN2/Z	--	N

Input Port	Pin Drive		Min	Min	Min	Cell
	Rise	Fall	Trans	Cap	Fanout	Deg
in1	--	--	--	--	--	--
in2	--	--	--	--	--	--
in3	--	--	--	--	--	--
in4	--	--	--	--	--	--
sel1	--	--	--	--	--	--
sel2	--	--	--	--	--	--
b1	--	--	--	--	--	--
b2	--	--	--	--	--	--

Output Delay						
Output Port	Min		Max		Related Clock	Fanout Load
	Rise	Fall	Rise	Fall		
out1	--	--	--	--	--	0.00
out2	--	--	--	--	--	0.00
b1	--	--	--	--	--	0.00
b2	--	--	--	--	--	0.00

1

The following is an example of a verbose port report specifying the -
significant_digits option:

prompt> **report_port -verbose -significant_digits 3**

Report : port

-verbose

-significant_digits 3

Design : char1

Version: V-2004.06

Date : Thu Dec 25 23:20:11 2003

Port	Dir	Pin Load	Wire Load	Max Trans	Max Cap	Connection Class	Attrs
i1	in	0.000	0.000	--	1.524	--	
i2	in	0.000	0.000	2.139	--	--	
i3	in	0.000	0.000	--	--	--	
o1	out	0.000	0.000	--	--	--	
o2	out	0.000	0.000	--	--	--	

Port	External Number Points	Max Wireload Model	Min Wireload Model	Min Pin Load	Min Wire Load
i1	1	--	--	--	--
i2	1	--	--	--	--
i3	1	--	--	--	--

Input Delay

Input Port	Min Rise	Min Fall	Max Rise	Max Fall	Related Clock	Max Fanout
i1	3.436	3.436	6.436	6.436	--	--
i2	3.436	3.436	6.436	6.436	--	--
i3	3.515	3.617	6.515	6.617	--	--

Input Port	Max Drive Rise	Max Drive Fall	Min Drive Rise	Min Drive Fall	Resistance Max	Min Cap	Min Fanout	Cell Deg
i1	--	--	--	--	--	--	0.548	--
i2	--	--	--	--	--	--	--	--
i3	--	--	--	--	--	--	--	--

Input Port	Max Tran Rise	Max Tran Fall	Min Tran Rise	Min Tran Fall
i1	3.832	3.832	--	--
i2	6.000	1.850	3.000	1.250
i3	3.832	3.832	--	--

Output Delay

Output Port	Min Rise	Min Fall	Max Rise	Max Fall	Related Clock	Fanout Load

report_port

o1	3.061	3.061	3.061	3.061	--	0.000
o2	3.061	3.061	3.061	3.061	--	0.000

1

SEE ALSO

characterize(2)
current_design(2)
current_instance(2)
report_net(2)
set_cell_degradation(2)
set_connection_class(2)
set_drive(2)
set_driving_cell(2)
set_equal(2)
set_fanout_load(2)
set_input_delay(2)
set_load(2)
set_logic_one(2)
set_logic_zero(2)
set_max_capacitance(2)
set_max_fanout(2)
set_max_transition(2)
set_min_capacitance(2)
set_opposite(2)
set_output_delay(2)
set_unconnected(2)
report_default_significant_digits(3)

report_port_protection_diodes

Reports the port protection diodes in the current design.

SYNTAX

status **report_port_protection_diodes**

ARGUMENTS

This command has no arguments.

DESCRIPTION

The **report_port_protection_diodes** command reports the port protection diodes in the current design.

The reporting format is shown as below:

```
***** Report : port protection diodes Design :  
top Version: A-2007.12-SP2 Date : Fri Jan 25 17:58:35 2008  
*****
```

Name_of_diode_cell diode_location name_of_connected_port

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example runs the **report_port_protection_diodes** command.

```
prompt> report_port_protection_diodes
```

SEE ALSO

insert_port_protection_diodes(2)

report_power

Calculates and reports dynamic and static power for a design or instance.

SYNTAX

```
int report_power
[-net]
[-cell]
[-only cell_or_net_list]
[-hier]
[-hier_level level_value]
[-verbose]
[-cumulative]
[-flat]
[-exclude_boundary_nets]
[-include_input_nets]
[-analysis_effort low | medium | high]
[-nworst number]
[-sort_mode mode]
[-histogram [-exclude_leq le_val
| -exclude_geq ge_val]]
[-nosplit]
[-scenario scenario_list]
```

Data Types

<i>cell_or_net_list</i>	object_list
<i>level_value</i>	integer
<i>number</i>	integer
<i>mode</i>	string
<i>le_val</i>	float
<i>ge_val</i>	float
<i>scenario_list</i>	list

ARGUMENTS

-net

Reports the power consumption of nets. Use the **-net** option alone, or use it with the **-cell** option. By default, only the design's summary power information is reported when neither option is specified.

-cell

Reports the power consumption of cells. When using the **-cell** option, some entries in the power report might not apply to certain cells. The column entry for those cells is annotated with N/A. Use the **-cell** option alone or with the **-net** option to report on cells and nets. By default, only the design's summary power information is reported when neither option is specified.

-only *cell_or_net_list*

Specifies a list of cells and/or nets to display with **-net** or **-cell**. With this option, only the cells and/or nets in the *cell_or_net_list* are listed in the power report. If both **-net** and **-only** are specified, the

cell_or_net_list must contain at least one net. Similarly, if both **-cell** and **-only** are specified, then the *cell_or_net_list* must contain at least one cell. If the **-net**, **-cell**, and **-only** options are specified together, the *cell_or_net_list* must contain at least one net and one cell.

-hier

Specifies that the report be in a hierarchical format, with power information on a block-by-block basis. Use the **-hier_level** option to limit the number of levels of hierarchy shown in the report. The **-sort**, **-cumulative**, **-nworst** options are ignored for this report. The switching, internal, and leakage power numbers are reported for each hierarchical block. The hierarchy is shown through indentations.

In DC-Topographical mode, with **-hier** option the power report includes estimated clock tree power numbers if power prediction was turned on in the last **compile_ultra** run. This entry is marked *CLOCK_TREE_EST*.

-hier_level level_value

Specifies the number of levels of hierarchy to display in the hierarchical report. This option can have any positive integral value greater than 1. Hierarchy levels deeper than the ones specified in this option are not shown in the hierarchical report. This option only modifies the hierarchical report and is ignored for all other types of reports.

-verbose

Displays additional detailed information about the power of the cells and/or nets. This option is valid only if **-net** and/or **-cell** is specified.

-cumulative

Reports the cumulative dynamic power of the design cells and/or nets. The fanin cumulative power of a cell or net is the dynamic power of the transitive fanin of the start point. Similarly, the fanout cumulative power is the dynamic power of the transitive fanout of the start point. The transitive fanin and fanout include and stop at sequential cells and primary design ports. The cumulative dynamic power includes the switching activity of all nets, and the internal power of all cells in the fanin or fanout. For every cell in the transitive fanin and fanout, both the cell's internal power and the switching power of the nets driven by the cell are included. The fanout cumulative power of a net includes the net's switching power but not the internal power of the cell(s) driving the net. The cumulative report is displayed after the standard cell or net report. The **-cumulative** option is only valid if the **-net** or **-cell** flag is specified.

-flat

Specifies that the power report traverse the hierarchy and report objects at all lower-levels (as if the design's hierarchy were flat). The default is to report objects at only the current level of hierarchy. For cell reports, if **-flat** is not specified, the power reported for a subdesign is the total power estimate for that subdesign, including all of its contents.

-exclude_boundary_nets

Specifies an option that is now obsolete.

-include_input_nets

Includes the switching power of primary input nets in the power report. The default is to exclude boundary input nets. This option affects the nets that

are chosen to be displayed in the net-specific report as well as the value of the total switching power. This option does not affect the cell leakage and internal power values.

-analysis_effort *low | medium | high*

Provides a tradeoff between runtime and accuracy. The default value is **low**. Specifying **low** effort results in the fastest runtime and the lowest accuracy of power estimates. Specifying **medium** or **high** effort results in a longer run that has increasing levels of accuracy. The analysis effort is considered only during the estimation of switching activity information, and, therefore, has an effect only when the design is not fully annotated with switching activity.

-nworst *number*

Filters the report so that it displays only the highest *number* power objects. This option is valid only if either **-net** and/or **-cell** is specified.

-sort_mode *mode*

Determines the sorting mode for report order and **-nworst** selection. The valid sorting modes for the **-net** or **-cell** options are as follows:

-net option	-cell option
-----	-----
name	name
cumulative_fanout	cumulative_fanout
cumulative_fanin	cumulative_fanin
net_static_probability	cell_internal_power
net_switching_power	cell_leakage_power
net_toggle_rate	dynamic_power
total_net_load	

When using both **-net** and **-cell** and specifying a sorting mode, you must select a sorting mode that is valid for both options. The mode is used for both the cell and the net reports.

If you do not explicitly set the sorting mode, a default is chosen based on the mode of the **report_power** command:

Mode	Implicit default
-----	-----
-net	net_switching_power
-cell	cell_internal_power
-net -cell	dynamic_power

-histogram [**-exclude_leq** *le_val* | **-exclude_geq** *ge_val*]

Displays a histogram-style report showing the number of nets in each power range. The **-exclude_leq** and **-exclude_geq** arguments are used to exclude data values less than *le_val* or greater than *ge_val*, respectively. Useful for displaying the range and variation of power in the design. This option displays the histogram report only if either **-net** or **-cell** is specified.

-nosplit

Prevents line-splitting and facilitates writing software to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds its column's width, the next field begins on a new line, starting in the correct column.

`-scenario scenario_list`

Reports power for given list of scenarios of a multiscenario design. Inactive scenarios will be skipped in the report. Each scenario is reported separately. If this option is not given, only the current scenario is reported.

DESCRIPTION

The **report_power** command calculates and reports power for a design. The command uses the user-annotated switching activity to calculate the net switching power, cell internal power, and cell leakage power, and displays the calculated values in a power report. The **report_power** command needs switching activity information on all design nets, and uses a switching activity propagation mechanism to estimate switching activity information on non-annotated design objects.

The options enable you to specify cells and/or nets for reporting. The default operation is to display the summary power values for only the current design. If a current instance is specified, **report_power** displays the summary power values for that instance. The instance's power is estimated in the context of the higher-level design, which means using the switching activity and load of the higher-level design.

The **-verbose** power information. The **-flat**, **-exclude_boundary_nets**, **-nworst**, and **-sort_mode** options enable the filtering of objects that are selected by **report_power**. The **-sort_mode** option also affects the formatting of the power reports by modifying the order of nets and/or cells that are displayed by **report_power**.

The **-cumulative** and **-histogram** options cause additional sections to be displayed in the power reports. The cumulative power section contains transitive fanin and fanout values for cells and/or nets in the design. The power histogram classifies the nets or cells into groups of power values, allowing for easier visual analysis of the range of power values and of the distribution of the nets/cells across that range. The **-histogram** options enable pruning of objects in the histogram by excluding values greater than or less than specified values.

Power analysis uses the current tool's mechanism to obtain loads. For instance, wire load models are used for the case of non-back-annotated non-topographical synthesis, back-annotated capacitances are used when these are available, etc.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example shows a **report_power** summary report. A medium effort analysis is performed to estimate the design's power values.

```
prompt> report_power -analysis medium
```

```
Information: Updating design information... (UID-85)  
Performing probabilistic propagation through design.
```

```

*****
Report : power
        -analysis_effort medium
Design : ALARM_BLOCK
Version: v3.2a
Date   : Sun Jun 19 15:45:24 1994
*****

```

Library(s) Used:

power_libdb (File: /remote/libraries/power_lib.db)

Operating Conditions:

Wire Loading Model Mode: enclosed

Design	Wire Loading Model	Library
ALARM_BLOCK	0.5K_TLM	power_lib.db
ALARM_STATE_MACHINE	0.5K_TLM	power_lib.db
ALARM_COUNTER	0.5K_TLM	power_lib.db
ALARM_COUNTER_DW01_inc_6_0	0.5K_TLM	power_lib.db

Global Operating Voltage = 4.75

Power-specific unit information :

Voltage Units = 1V

Capacitance Units = 50.029999ff

Time Units = 1ns

Dynamic Power Units = 10uW (derived from V,C,T units)

Leakage Power Units = 1nW

Cell Internal Power = 165.1648 uW (32%)

Net Switching Power = 348.8617 uW (67%)

Total Dynamic Power = 514.0266 uW (100%)

Cell Leakage Power = 76.0000 nW

The following example shows a net power report sorted by **net_switching_power** and filtered to display only the 5 nets with highest switching power. A low effort analysis is performed to estimate the design's power values.

prompt> **report_power -net -flat -nworst 5**

```

*****
Report : power
        -net
        -analysis_effort low
        -nworst 5
        -flat
        -sort_mode net_switching_power
Design : ALARM_BLOCK
Version: v3.2a

```

Date : Sun Jun 19 15:45:26 1994

Library(s) Used:

power_lib.db (File: /remote/libraries/power_lib.db)

Operating Conditions:

Wire Loading Model Mode: enclosed

Design	Wire Loading Model	Library
ALARM_BLOCK	0.5K_TLM	power_lib.db
ALARM_STATE_MACHINE	0.5K_TLM	power_lib.db
ALARM_COUNTER	0.5K_TLM	power_lib.db
ALARM_COUNTER_DW01_inc_6_0	0.5K_TLM	power_lib.db

Global Operating Voltage = 4.75

Power-specific unit information :

Voltage Units = 1V

Capacitance Units = 50.029999ff

Time Units = 1ns

Dynamic Power Units = 10uW (derived from V,C,T units)

Leakage Power Units = 1nW

Net	Total Net Load	Static Prob.	Toggle Rate	Switching Power	Attrs
ACOUNT/CLK	20.467	0.500	0.1000	115.5149	
ACOUNT/n493	23.193	0.985	0.0250	32.7255	
ASM/n225	9.165	0.985	0.0250	12.9314	
ACOUNT/HRS_OUT[3]	6.365	0.537	0.0303	10.8763	
ACOUNT/HRS_OUT[2]	5.161	0.537	0.0303	8.8202	
Total (5 nets)				18.0868 uW	

The following example displays a cell report in which an additional cumulative cell power report is generated. The cells are sorted by cumulative fanout power values, and only the top 5 are reported. A low effort analysis is performed to estimate the design's power values.

```
prompt> report_power -cell -flat -cumulative \  
-sort_mode cumulative_fanout -nworst 5
```

Report : power

-cell

-analysis_effort low

-nworst 5

-cumulative

-flat

-sort_mode cumulative_fanout

Design : ALARM_BLOCK

Version: v3.2a

report_power
1950

Date : Sun Jun 19 15:45:28 1994

Library(s) Used:

power_lib.db (File: /root/libraries/power_lib.db)

Operating Conditions:

Wire Loading Model Mode: enclosed

Design	Wire Loading Model	Library
ALARM_BLOCK	0.5K_TLM	power_lib.db
ALARM_STATE_MACHINE	0.5K_TLM	power_lib.db
ALARM_COUNTER	0.5K_TLM	power_lib.db
ALARM_COUNTER_DW01_inc_6_0	0.5K_TLM	power_lib.db

Global Operating Voltage = 4.75

Power-specific unit information :

Voltage Units = 1V

Capacitance Units = 50.029999ff

Time Units = 1ns

Dynamic Power Units = 10uW (derived from V,C,T units)

Leakage Power Units = 1nW

Attributes

h - Hierarchical cell

Cell	Cell Internal Power	Driven Net Switching Power	Tot Dynamic Power (% Cell/Tot)	Cell Leakage Power	Attrs
ACOUNT/MINS_OUT_reg[1]	3.8997	13.2200	17.120 (22%)	1.0000	
ACOUNT/MINS_OUT_reg[3]	10.8977	2.0806	12.978 (83%)	1.0000	
ACOUNT/MINS_OUT_reg[0]	10.8987	2.0744	12.973 (84%)	1.0000	
ACOUNT/MINS_OUT_reg[4]	10.8974	2.0869	12.984 (83%)	1.0000	
ACOUNT/MINS_OUT_reg[5]	10.8977	2.0770	12.975 (83%)	1.0000	
Totals (5 cells)	4.7491uW	2.1538uW	6.903uW (68%)	5.0000nW	

Cell	Cumulative Transitive Fanin Power	Cumulative Transitive Fanout Power
ACOUNT/MINS_OUT_reg[1]	17.11972	182.40425
ACOUNT/MINS_OUT_reg[3]	12.97823	173.69908
ACOUNT/MINS_OUT_reg[0]	12.97306	173.68782
ACOUNT/MINS_OUT_reg[4]	12.98429	172.32205
ACOUNT/MINS_OUT_reg[5]	12.97466	172.30254

(5 cells)

SEE ALSO

`propagate_switching_activity(2)`
`set_switching_activity(2)`
`power_keep_license_after_power_commands(3)`

report_power_calculation

Displays the calculation of the internal power for a pin, the leakage power for a cell, or the switching power for a net.

SYNTAX

```
int report_power_calculation  
pin_cell_or_net_list  
[-state_condition boolean_eq_of_pins | default | all]  
[-path_source pin_name | default | all]  
[-rise]  
[-fall]  
[-verbose]  
[-nosplit]
```

Data Types

<i>pin_cell_or_net_list</i>	object_list
<i>boolean_eq_of_pins</i>	string
<i>pin_name</i>	string

ARGUMENTS

pin_cell_or_net_list

Specifies the objects on which the power calculation reports. All objects in the list must be of the same type. The type of power calculation depends on the object type, as follows:

- The tool reports the internal power calculation for a pin.
- The tool reports the leakage power calculation for a cell.
- The tool reports the switching power calculation for a net.

-state_condition *boolean_eq_of_pins* | default | all

Reports only the tables or polynomials with matching state conditions. This option is valid only for pin or cell type objects. Use **-state_condition default** to specify the default state condition. Use **-state_condition all** to consider all states for reporting.

-path_source *pin_name* | default | all]

Reports only the tables or polynomials with matching path sources. This option is valid only for pin type objects. Use **-path_source default** to specify the default path condition (no path source). Use **-path_source all** to specify that all path sources must be considered for reporting.

-rise

Limits the report of internal power calculation to only the rise power.

-fall

Limits the report of internal power calculation to only the fall power.

-verbose

Increases the amount of information that is reported for cells or pins. For cells, the leakage power calculation report is appended by an internal power calculation report for all pins and all possible states and path sources of the cells in the object list. For pins, all of the possible states and paths are printed (equivalent to specifying **-state_condition all -path_source all**).

-nosplit

Prevents line-splitting and facilitates writing software to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds its column's width, the next field begins on a new line, starting in the correct column.

DESCRIPTION

The **report_power_calculation** command provides detailed power calculation information for the specified pin, cell, or net. Use this information for debugging or verifying power data in a technology library. Before using this command to display details of internal or leakage power calculation, read the technology library file into the system using the **read_lib** command. This command automatically compiles the library and enables the **report_power_calculation** command for internal and leakage power. Otherwise, the built-in security mechanism terminates the command when issued for pin internal power or cell leakage power calculation. This restriction does not apply to net switching power calculation.

It is considered best practice to use the **report_power** command before issuing **report_power_calculation**. This ensures the propagation of any missing switching activity. If the design is not completely annotated with switching activity when **report_power_calculation** is executed, the missing activity is propagated with the same analysis effort that was used during the last issued **report_power** command. If no power analysis is performed prior to issuing the **report_power_calculation** command, a low analysis effort is used for switching activity propagation.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example shows the calculation of path dependent internal power from input pin B to output pin Y. The contributions for rise and fall internal power are shown separately.

```
prompt> report_power_calculation U4/Y -state default -path B
```

```
*****
```

```
Report : power_calculation
```

```
        U4/Y
```

```
        -state_condition default
```

```
        -path_source B
```

```
Design : rpc
```

```
report_power_calculation
```

```
1954
```


Version: V-2004.06
Date : Thu Feb 5 12:10:44 2004

Library(s) Used:

example (File: /root/libraries/example.db)

Operating Conditions: typical Library: example
Wire Load Model Mode: segmented

Design	Wire Load Model	Library
-----	-----	-----
rpc	a	example

Global Operating Voltage = 0.9
Power-specific unit information :
Voltage Units = 1V
Capacitance Units = 1.000000pf
Time Units = 1ns
Dynamic Power Units = 1mW (derived from V,C,T units)
Leakage Power Units = 1nW

PD Pin Internal Power Calculation
cell: U4
pin: Y
path source: B

Rise internal energy per transition = 0.143423
library model: NLPM
table variables:
X = total_output_net_capacitance = 0.4
Y = input_net_transition = 0.3
relevant portion of lookup table:

	(X)	0.1050	(X)	0.3550	
(Y)	0.0500	(Z)	0.1310	(Z)	0.1410
(Y)	0.4510	(Z)	0.1320	(Z)	0.1420

$$Z = A + B \cdot X + C \cdot Y + D \cdot X \cdot Y$$

A =	0.1267	B =	0.0400
C =	0.0025	D =	0.0000

Z = 0.143423
(no scaling since the library has no internal power k-factors)

SDPD Rise Pin Toggle Rate = 0.164
sdpd rise pin toggle rate = path weight * sd rise pin toggle rate
path weight = 1
sd rise pin toggle rate = 0.164 (estimated)

PD Rise Pin Internal Power = 0.0235214
pin internal power = internal energy * pin toggle rate

Fall internal energy per transition = 0.143423
library model: NLPM

```

table variables:
  X = total_output_net_capacitance = 0.4
  Y = input_net_transition = 0.3
relevant portion of lookup table:
      (X)  0.1050      (X)  0.3550
(Y)  0.0500      (Z)  0.1310      (Z)  0.1410
(Y)  0.4510      (Z)  0.1320      (Z)  0.1420

```

$$\begin{aligned}
 Z &= A + B \cdot X + C \cdot Y + D \cdot X \cdot Y \\
 A &= 0.1267 & B &= 0.0400 \\
 C &= 0.0025 & D &= 0.0000
 \end{aligned}$$

```

Z = 0.143423
(no scaling since the library has no internal power k-factors)

```

```

SDPD Fall Pin Toggle Rate = 0.164
sdpd fall pin toggle rate = path weight * sd fall pin toggle rate
path weight = 1
sd fall pin toggle rate = 0.164 (estimated)

```

```

PD Fall Pin Internal Power = 0.0235214
pin internal power = internal energy * pin toggle rate

```

```

Total Pin Internal Power = 0.0470429

```

The following is an example of a state dependent leakage power calculation report. An SPPM leakage power model was used for this example.

```

prompt> report_power_calculation U5 -state "A B"

```

```

*****
Report : power_calculation
      U5
      -state_condition A B
Design : rpc
Version: V-2004.06
Date   : Thu Feb  5 12:00:48 2004
*****

```

```

Library(s) Used:

```

```

      example (File: /root/libraries/example.db)

```

```

Operating Conditions: typical  Library: example
Wire Load Model Mode: segmented

```

Design	Wire Load Model	Library
-----	-----	-----
rpc	a	example

```

Global Operating Voltage = 0.9
Power-specific unit information :
  Voltage Units = 1V
  Capacitance Units = 1.000000pf
  Time Units = 1ns

```

Dynamic Power Units = 1mW (derived from V,C,T units)
Leakage Power Units = 1nW

SD Cell Leakage Power Calculation

cell: U5
state condition: A B

Leakage Power Value = 0.1097
library model: SPPM
domain: D1
original polynomial:
"0.0134+0.0002a+0.0742b+0.0006ab"
variables:
a = temperature = 40
b = voltage = 0.9
reduced value = 0.1097
(no scaling for SPPM leakage power)

State Probability = 0.18 (estimated)

SD Leakage Power = 0.01974
cell leakage power = leakage power value * state probability

The following report shows how switching power calculation is performed:

prompt> **report_power_calculation q**

Report : power_calculation
q
Design : rpc
Version: V-2004.06
Date : Thu Feb 5 12:12:32 2004

Library(s) Used:

example (File: /root/libraries/example.db)

Operating Conditions: typical Library: example
Wire Load Model Mode: segmented

Design	Wire Load Model	Library
rpc	a	example

Global Operating Voltage = 0.9
Power-specific unit information :
Voltage Units = 1V
Capacitance Units = 1.000000pf
Time Units = 1ns
Dynamic Power Units = 1mW (derived from V,C,T units)
Leakage Power Units = 1nW

Net Switching Power Calculation

```
net: q  
driver: U4/Y
```

```
Switching Energy Per Transition = 0.162  
switching energy = 0.5 * capacitance * voltage ^ 2  
total net capacitance = 0.4  
voltage = 0.9
```

```
Net Toggle Rate = 0.8 (user annotated)
```

```
Switching power = 0.1296  
net switching power = switching energy * net toggle rate
```

SEE ALSO

```
report_lib(2)  
report_power(2)  
set_switching_activity(2)
```

report_power_domain

Reports information about the specified power domain.

SYNTAX for UPF Mode

status **report_power_domain**

[*power_domains*]
[-scope *instance_name*]
[-operating_condition]

Data Types for UPF Mode

power_domains list
instance_name string

SYNTAX for Non-UPF Mode

status **report_power_domain**

[*power_domains*]

Data Types for Non-UPF Mode

power_domains list

ARGUMENTS

power_domains

Specifies the power domains to be reported. The option value can be a collection of power domains or name patterns.

The command fails if it does not find the specified power domain.

In UPF mode, if this option is not specified, the command reports all power domains in the specified scope, or the current scope, if none is specified.

In non-UPF mode, if this this option is not specified, the command reports all power domains in the current design.

-scope *instance_name*

This option is available only in UPF mode.

Specifies the scope in which the power domain is to be reported. The instance name is the name of a hierarchical cell.

If this option is not specified, the power domain is reported for the current scope.

If this option is specified, the *power_domains* list must be a simple name, not a collection.

-operating_condition

This option is available only in UPF mode.

Reports the operating conditions. If the design is not a multivoltage design, no operating conditions are reported.

DESCRIPTION

The **report_power_domain** command enables you to get detailed information about existing power domains. If you specify the power domains, only those power domains are reported; otherwise all power domains in the current scope (UPF mode) or current design (non-UPF mode) are reported.

The following information is reported for each power domain:

- The hierarchical name relative to the current scope
- The current scope
- The elements of the current scope
- The supply nets available for use (supply nets that are not domain supply nets, a part of any isolation or retention strategies, and connected to switches)
- The primary power and ground net
- The isolation strategies and the isolation power and ground nets,
- The retention strategies and its retention power and ground nets,
- The power switches
- The input and output supply nets
- The maximum voltage of each supply net is reported if it has been set by the **set_voltage** command.
- If the **-operating_condition** option is used, then operating conditions are also reported.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following UPF mode example reports the power domain in the current scope:

```
prompt> report_power_domain
*****
Report : power_domain
Design : top
Version: A-2007.12-SP1
Date   : Wed Nov 21 03:15:21 2007
*****
```

```

Power Domain           : mid1/A
Current Scope          : top
Elements               : mid1/bot1, mid1/bot2, mid1
Available Supply Nets  : SN_A_1, SN_A_2, SN_A_3, SN_A_4, SN_A_5, SN_A_6

```

```

Connections           -- Power --           -- Ground --
Primary:              mid1/A_VDD (1.08)      mid1/A_VSS (1.08)
Isolation: ISO_STR_1   mid1/ISO_VDD (1.88)    mid1/ISO_VSS (1.88)
Retention: RET_STR_1   mid1/R_VDD (1.68)      mid1/R_VSS (1.68)
Retention: RET_STR_2   mid1/R_VDD_1 (1.55)    mid1/R_VSS_1 (1.55)

```

```

Switches              -- Input --           -- Output --
Switch: mid1/SW1       mid1/SW_NET_1         mid1/SW_NET_2
Switch: mid1/SW2       mid1/SW_NET_3         mid1/SW_NET_4

```

1

The following UPF mode example reports the power domain and the operating condition for the current scope.

prompt> **report_power_domain PD1 -operating_condition**

```

*****
Report : power_domain
Design : top
Version: C-2009.06
Date   : Wed Mar 11 03:15:21 2009
*****

```

```

-----

Power Domain           : PD1
Current Scope          : top
Elements               : sub
Available Supply Nets  : SN_A_1, SN_A_2

Connections           -- Power --           -- Ground --
Primary:              A_VDD (1.08)          A_VSS (1.08)
Isolation: ISO_STR_1   ISO_VDD (1.88)       ISO_VSS (0.00)
Retention:
Switches              -- Input --           -- Output --
Switch:

```

```

Max Operating Conditions: slow
Min Operating Conditions: fast
Max Operating Voltage: 0.900
Min Operating Voltage: 1.100
Max Operating Temperature: 125.000
Min Operating Temperature: -40.000

```

1

The following non-UPF mode example reports about all power domains in the current design:

```

prompt> create_power_domain T
1

prompt> create_power_domain A -object_list [get_cells PD0_INST] \
    -power_down -power_down_ctrl [get_nets A]
1

prompt> create_power_domain B -power_down \
    -object_list [get_cells PD1_INST]
1

prompt> report_power_domain

```

```

*****

```

```

Report : power_domain
Design : top
Version: Y-2006.06-H03
Date   : Tue Apr  4 15:28:18 2006
*****

```

```

Total of 3 power domains defined for design 'top'.

```

```

-----

Power Domain : T  [AO]
Blocks : <top level>
Connections :      -- Power --              -- Ground --
    Primary          T_VDD                  T_VSS
    Backup            <none>                 <none>
    Internal          T_VDD                  T_VSS

```

```

-----

Power Domain : A
Blocks : PD0_INST
Power Down : A
Connections :      -- Power --              -- Ground --
    Primary          A_VDD                  A_VSS
    Backup            VDD_Backup             VSS_Backup
    Internal          A_VDD                  A_VSS

```

```

-----

Power Domain : B
Blocks : PD1_INST
Connections :      -- Power --              -- Ground --
    Primary          B_VDD                  B_VSS
    Backup            <none>                 <none>
    Internal          B_VDD                  B_VSS

```

```

1

```


SEE ALSO

`connect_power_domain(2)`
`create_power_domain(2)`
`remove_power_domain(2)`

report_power_gating

Reports the power gating style of retention registers in the design.

SYNTAX

```
int report_power_gating  
[cell_or_design_list]  
[-missing]  
[-unconnected]
```

Data Types

cell_or_design_list list

ARGUMENTS

cell_or_design_list
Specifies a list of cells or designs in which the power gating styles of cells are reported. The default is the current design.

-missing
Reports the sequential cells which don't have the power gating style.

-unconnected
Only reports the retention registers that are not stitched.

DESCRIPTION

This command reports the power gating style of retention registers. The retention registers in the libraries have a cell level attribute *power_gating_cell* to specify the power gating styles of the registers. The command only reports the cells having the *power_gating_style* attributes set by *set_power_gating_style* command. it requires a Power Compiler license to run.

If -missing is specified, the command will only report the sequential cells which don't have the power gating style.

If -unconnected is specified, the command will only report the retention registers whose power gating pins have not been stitched yet.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example indicates that the power gating styles of retention registers in design *FOO* will be reported.

```
report_power_gating  
1964
```

```
prompt>>report_power_gating F00
```

SEE ALSO

```
set_power_gating_style(2)  
power_enable_power_gating(3)
```

report_power_guidefp

Reports existing power guides with different information.

SYNTAX

```
report_power_guidefp  
[power_guide_list]
```

ARGUMENTS

power_guide_list
The list of power guides to be reported. This is optional.

DESCRIPTION

The command **report_power_guide** reports the given power guides. If no specific power guide is given, all existing power guides will be reported.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following is an example output of report_power_guide command :-

```
prompt> create_bounds -name AO_WELL -coordinate {10 10 20 20} -exclusive
prompt> create_voltage_area -name VA_SHUTDOWN -coordinate {5 5 30 30}
[get_cells A_INST]
prompt> set_power_guide -name AO_WELL
prompt> report_power_guide AO_WELL
*****
Report Power Guide
*****
Name      : AO_WELL
Type      : Always-on
Geometry  : {(10 10) (20 20)}
Voltage Area Information :
  Container Voltage Area : VA_SHUTDOWN
*****
```

SEE ALSO

```
create_bounds(2)
create_voltage_area(2)
set_power_guide(2)
unset_power_guide(2)
```

report_power_net_info

Reports the power net information for the current design.

SYNTAX

```
integer report_power_net_info
```

ARGUMENTS

The **report_power_net_info** command has no arguments.

DESCRIPTION

The **report_power_net_info** command reports the power net information for the current design.

EXAMPLE

The following example creates the power domain and then uses **report_power_net_info** to report the information:

```
prompt> create_power_domain T
1
prompt> create_power_net_info T_VDD -power
1
prompt> create_power_net_info T_VSS -gnd
1
prompt> create_power_domain T
1
prompt> connect_power_domain T -primary_power_net T_VDD \
      -primary_ground_net T_VSS
1
prompt> report_power_net_info
```

```
*****
Report : power_net
Design : top
Version: Y-2006.06
Date   : Tue Apr  4 16:11:52 2006
*****
```

Total of 2 power nets defined.

Power Net 'T_VDD' (power)

Primary Power Hookups:	T
Internal Power Hookups:	T

Power Net 'T_VSS' (ground)

Primary Ground Hookups:	T
Internal Ground Hookups:	T

1

SEE ALSO

create_power_net_info(2)
connect_power_domain(2)

report_power_options

Reports different power optimization options.

SYNTAX

status **report_power_options**

ARGUMENTS

None.

DESCRIPTION

The **report_power_options** command reports power optimization options.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows the use of the command.

```
report_power_options
```

SEE ALSO

```
set_power_options(2)
```

report_power_pin_info

Reports the power pin information for technology library cells or leaf cells. In UPF mode, the command reports power pin information only for instantiated cells and not the library cells.

SYNTAX

```
status report_power_pin_info
      object_list
```

Data Types

object_list list

ARGUMENTS

object_list
 Specifies a list of technology library cells or a list of leaf cells. Library cells are not allowed as arguments in UPF mode.

DESCRIPTION

The **report_power_pin_info** command reports the power pin information for technology library cells or leaf level cells in the current design.

When a list of library cells is specified, the name, the types and the voltage specification of the power pins are reported. Library cells are not allowed as arguments in the UPF mode.

When a list of leaf cells is specified in the *object_list*, the supply (power and ground) nets that are connected to the power pins are also reported. If a supply connection is marked with an asterisk (*), it is an exception (explicit) connection. Exception supply connections are connections that are established with the **connect_supply_net** command in UPF mode, or with the **connect_power_net_info** command in non-UPF mode.

Exception supply connections can occur in the following situations:

- Explicit specification of the **connect_supply_net** or **connect_power_net_info** command in UPF mode and non-UPF mode, respectively.
- Power intent specification in the UPF file.
- Level-shifter cell insertion during compilation.

In any of these cases you can observe the exception connections using the **write_script** or **save_upf** commands. The connections appear as **connect_supply_net** (or **connect_power_net_info** in non-UPF mode) in the files written out.

In UPF mode, the best-case and worst-case voltages of the connected power net are reported, along with the name of the power net. If a voltage is not specified on the

connected net, a hyphen (-) is printed in the respective voltage fields.

Hierarchical cells are ignored by this command. In UPF mode, the tool issues the message "No power pins to report".

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports power pin information in UPF mode:

```
prompt> fbreport_power_pin_info

*****
Report : power_pin_info
Design : top
Version: B-2008.09
Date   : Mon Mar 24 02:04:49 2008
*****
```

Note: Power connections marked by (*) are exceptional

Cell	Power Pin	Type	Best Case	Worst Case	Connected
	Name		Voltage	Voltage	Power
					Net
I0	PWR	Primary Power	-	-	-
I0	GND	Primary Ground	-	-	-

1

The following example shows the voltage on the connected power nets set:

```
prompt> report_power_pin_info [get_cells * -hier]

*****
Report : power_pin_info
Design : top
Version: B-2008.09
Date   : Mon Mar 24 02:04:49 2008
*****
```

Note: Power connections marked by (*) are exceptional

Cell	Power Pin	Type	Best Case	Worst Case	Connected Power
	Name		Voltage	Voltage	Net

PD0_INST/I0	PWR	Primary Power	1.08	0.00	PD0_VDD
PD0_INST/I0	GND	Primary Ground	0.00	0.00	PD0_VSS
PD0_INST/I1	PWR	Primary Power	1.08	0.00	PD0_VDD
PD0_INST/I1	GND	Primary Ground	0.00	0.00	PD0_VSS
I0	PWR	Primary Power	1.08	0.00	PD0 (*)
I0	GND	Primary Ground	-	-	-
I1	PWR	Primary Power	1.08	0.00	PD0 (*)
I1	GND	Primary Ground	-	-	-

1

In the following example a hierarchical cell is passed as an argument:

```
prompt> report_power_pin_info [get_cells PD0_INST]
```

```
*****
Report : power_pin_info
Design : top
Version: B-2008.09
Date   : Mon Mar 24 02:04:49 2008
*****
```

No power pins to report.

1

The following example uses the **report_power_pin_info** command to report the power pin information in non-UPF mode:

```
prompt> report_power_pin_info [get_cells -hier]
```

```
*****
Report : power pin info
Design : top
Version: Y-2006.06
Date   : Tue Mar 28 19:04:51 2006
*****
```

Note: Power connections marked by (*) are exceptional

Cell	Power Pin Name	Type	Voltage	Power Net Connected
PD0_INST/I0	PWR	primary_power	1.0000	
PD0_INST/I0	GND	primary_ground	0.0000	
PD0_INST/I1	PWR	primary_power	1.0000	
PD0_INST/I1	GND	primary_ground	0.0000	
I0	PWR	primary_power	1.0000	
I0	GND	primary_ground	0.0000	
I1	PWR	primary_power	1.0000	
I1	GND	primary_ground	0.0000	

1

```
prompt> report_power_pin_info [get_lib_cells -of [get_cells -hier]]
```

```
report_power_pin_info
```

1972

Report : power pin info
 Design : top
 Version: Y-2006.06
 Date : Tue Mar 28 19:04:51 2006

Library Cell	Power Pin Name	Type	Voltage
-----	-----	-----	-----
INVX2	PWR	primary_power	1.0000
INVX2	GND	primary_ground	0.0000
BUFX2	PWR	primary_power	1.0000
BUFX2	GND	primary_ground	0.0000
AND2X1	PWR	primary_power	1.0000
AND2X1	GND	primary_ground	0.0000

prompt> **connect_power_net_info** I0 -power_pin_name PWR \
 -power_net_name exp_VDD

1

prompt> **connect_power_net_info** PD0_INST/I0 -power_pin_name PWR \
 -power_net_name exp_VDD

1

prompt> **report_power_pin_info** [get_cells -hier]

Report : power pin info
 Design : top
 Version: Y-2006.06
 Date : Tue Mar 28 19:04:51 2006

Note: Power connections marked by (*) are exceptional

Cell	Power Pin Name	Type	Voltage	Power Net Connected
-----	-----	-----	-----	-----
PD0_INST/I0	PWR	primary_power	1.0000	exp_VDD (*)
PD0_INST/I0	GND	primary_ground	0.0000	A_VSS
PD0_INST/I1	PWR	primary_power	1.0000	A_VDD
PD0_INST/I1	GND	primary_ground	0.0000	A_VSS
I0	PWR	primary_power	1.0000	exp_VDD (*)
I0	GND	primary_ground	0.0000	T_VSS
I1	PWR	primary_power	1.0000	T_VDD
I1	GND	primary_ground	0.0000	T_VSS

1

prompt> **disconnect_power_net_info** I1 -power_pin_name PWR

1

prompt> **report_power_pin_info** [get_cells -hier]

```

Report : power pin info
Design : top
Version: Y-2006.06
Date   : Tue Mar 28 19:04:51 2006
*****

```

Note: Power connections marked by (*) are exceptional

Cell	Power Pin Name	Type	Voltage	Power Net Connected

PD0_INST/I0	PWR	primary_power	1.0000	exp_VDD (*)
PD0_INST/I0	GND	primary_ground	0.0000	A_VSS
PD0_INST/I1	PWR	primary_power	1.0000	A_VDD
PD0_INST/I1	GND	primary_ground	0.0000	A_VSS
I0	PWR	primary_power	1.0000	exp_VDD (*)
I0	GND	primary_ground	0.0000	T_VSS
I1	PWR	primary_power	1.0000	T_VDD
I1	GND	primary_ground	0.0000	T_VSS
1				

SEE ALSO

```

connect_power_domain(2)
connect_power_net_info(2)
connect_supply_net(2)
disconnect_power_net_info(2)

```

report_power_switch

Reports all of the specified power switches. This command is supported only in UPF mode.

SYNTAX

```
status report_power_switch
[-verbose]
```

ARGUMENTS

-verbose
Reports details about the specified power switches.

DESCRIPTION

The *report_power_switch* command enables you to get the information of the specified power switches. The power switch information reported includes its relative hierarchical name that also contains the scope in which it is created, the current scope name, the name of the simple power domain of which the switch is a part, its output supply net, and its input supply net.

The command also prints out a verbose report if the **-verbose** switch is present. The verbose report contains all of the information specified above along with the ctrl_port names, the ack_port names, and the on and off state names with their respective Boolean functions.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports information about all power switches within the power domain named PD1 in the current scope:

```
prompt> report_power_switch SWA
```

```
-----
Power Switch           : SWA
Current Scope          : mid1
Switch Power Domain    : A
Input port net         : SNA_2
Output port net        : SNA_1
-----
```

1

The following example reports the information about the same power switches as

above, this time using the **-verbose** option to show more detail:

```
prompt> report_power_switch SWA -verbose
```

```
-----  
Power Switch           : SWA  
Current Scope          : mid1  
Switch Power Domain    : A  
Input port net         : SNA_2  
Output port net        : SNA_1  
Control Ports          : ctrl  
Acknowledge Ports      : ackport1, ackport2  
  
On States              : <on1, ctrl>, <on2, !logic1 & ctrl>,  
                        <on3, !logic1 & !ctrl>, <on4, !logic1 | ctrl>,  
                        <on5, !logic1 | !ctrl>  
Off States              : <off1, outport>, <off2, outport>  
-----
```

1

SEE ALSO

```
create_power_switch(2)  
remove_power_switch(2)
```

report_preferred_routing_direction

Reports the preferred routing direction for all routing layers.

SYNTAX

string **report_preferred_routing_direction**

ARGUMENTS

The **report_preferred_routing_direction** command has no arguments.

DESCRIPTION

The **report_preferred_routing_direction** command reports the preferred routing direction for all layers from in-memory information.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following is an example of a preferred routing direction report:

```
prompt> report_preferred_routing_direction
```

```
*****  
Report : Layers  
Design : core_chip  
Version: Y-2006.06  
Date   : Thu Mar 23 03:43:06 2006  
*****
```

Layer Name	Library	Design	Tool understands
metal1	Horizontal	Not Set	Horizontal
metal2	Vertical	Not Set	Vertical
metal3	Horizontal	Not Set	Horizontal
metal4	Vertical	Not Set	Vertical
metal5	Horizontal	Vertical	Vertical
metal6	Vertical	Vertical	Vertical

WARNING: Consecutive layers have the same routing direction.

1

SEE ALSO

`remove_preferred_routing_direction(2)`
`set_preferred_routing_direction(2)`

report_preroute_drc_strategy

Reports preroute drc options set in **set_preroute_drc_strategy** to change the internal rules in the following PG commands: **create_rectangular_rings**, **create_pad_rings**, **create_power_straps**, **create_preroute_vias**, **preroute_instances**, **preroute_standard_cells**. These settings are not persistent in the Milkyway design.

SYNTAX

status **report_preroute_drc_strategy**

ARGUMENTS

The **report_preroute_drc_strategy** command has no arguments.

DESCRIPTION

Reports preroute drc options set in **set_preroute_drc_strategy** to change the internal rules in the following PG commands: **create_rectangular_rings**, **create_pad_rings**, **create_power_straps**, **create_preroute_vias**, **preroute_instances**, **preroute_standard_cells**. Recommended to check how DRC rules are applied in the above commands.

The reported options have the following meaning:

DRC spacing: Specifies the mode for design rule checking, either *radial* or *manhattan*. A *radial* rule checks for spacing violations within a circular area. A *manhattan* rule checks for spacing violations within the defined spacing distance horizontally and vertically.

Protect pin access edge: If True (1), it will prevent the Signal Pin Access Edge from being blocked by the Prerouter.

Treat fat blockages as fat: If True (1), fat blockages are treated as fat wires. Otherwise, design rule checking treats fat blockages as thin wires.

Use fat vias if wires meet width requirements: If True (1), fat vias will be used for non-turning layer changes if either wire meets the width requirement for fat vias.

Ignore contents of std cell pins in DRC checking: If True (1), design rule checking is not done for objects in standard cells. RISKY!

Ignore top level pins in DRC checking: If True (1), design rule checking is not done for top-level pins. RISKY!

Ignore same net metal-metal in DRC checking: If True (1), design rule checking ignores same net metal-to-metal rule. RISKY!

Quick DRC check: If True (1), design rule checking skips certain rules. RISKY!

Merge thin wires in DRC checking: If True (1), prerouter checks and merges thin

wires into fat wires for design rule checking. SLOW!

Turn off all DRC check: If True (1), no design rule checking is done. EXTREMELY RISKY!

Report detailed info about failed connections: If True (1), detailed information about the failed PG command operation is reported.

Range within which prerouter may perform jogging: If greater than 0, a jogging range distance within which the prerouter might perform limited jogging, is set. Jogging range defines the maximum distance for a dogleg connection on the same layer, it is not for layer switching.

Min metal layer to be used by prerouter: Shows the minimum metal layer used for prerouting.

Max metal layer to be used by prerouter: Shows the maximum metal layer used for prerouting.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

```
report_preroute_drc_strategy

Current Preroute DRC options
-----
DRC spacing = Radial
Protect pin access edge = 0
Treat fat blockages as fat = 0
Use fat vias if wires meet width requirements = 0
Ignore contents of std cell pins in DRC checking = 0
Ignore top level pins in DRC checking = 0
Ignore same net metal-metal in DRC checkin = 0
Quick DRC check = 0
Merge thin wires in DRC checking = 0
Turn off all DRC check = 0
Report detailed info about failed connections = 0
Range within which prerouter may perform jogging = 0
Min metal layer to be used by prerouter = 31 (M1)
Max metal layer to be used by prerouter = 39 (M9)
```

SEE ALSO

```
set_preroute_drc_strategy(2)
create_rectangular_rings(2)
create_pad_rings(2)
create_power_straps(2)
create_preroute_vias(2)
preroute_instances(2)
preroute_standard_cells(2)
```

report_primetime_options

Reports the PrimeTime options for the **signoff_opt** command.

SYNTAX

status **report_primetime_options**

ARGUMENTS

This command does not take any arguments.

DESCRIPTION

report_primetime_options command to examine the options which are set by **set_primetime_options** command.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example runs the **report_primetime_options** command.

```
prompt> open_mw_cel -lib my_lib my_cel
prompt> report_primetime_options

*****
Report : PrimeTime_options
Design : my_design
Version: X-2005.12-ICC
Date   : Mon Aug 15 15:55:58 2005
*****

PrimeTime exec dir: /PrimeTime/linux/syn/bin
```

SEE ALSO

set_primetime_options(2)
signoff_opt(2)
place_opt(2)
clock_opt(2)
route_opt(2)

report_pst

Report power states in current design (UPF mode only).

SYNTAX

```
int report_pst
[-verbose]
[-supplies supply_list]
[-scope instance_name]
[-power_nets supply_net_list]
[-compress]
[-trace_name]
[-significant_digits digit]
[-width line_width]
[-column_space column_space]
[-derived]
```

Data Types

<i>supply_list</i>	list
<i>instance_name</i>	string
<i>digit</i>	integer

ARGUMENT

- verbose
All power states will be reported explicitly even when all power state of a net or port are listed in the PST. By default, the PST shows '*' when all states of nets or ports are used.
- supplies *supply_list*
A list of supply net or port names to be included in the report. The order of power nets in this list also determines the order they are shown in the report. By default, all power nets in current design are included.
- scope *instance_name*
Specifies a design instance different from current design. The resulting PST contains the supply states derived from all PST contained in the specified design instance.
- power_nets *supply_net_list*
Same as '-supplies' argument. To be obsolete in the next service pack.
- compress
Ignored. To be removed in the next service pack.
- trace_name
Ignored. To be removed in the next service pack.
- significant_digits *digit*
Ignored. To be removed in the next service pack.

-width *line_width*

Ignored. To be removed in the next service pack.

DESCRIPTION

This command reports the resulting power state table (PST) of the current design. The resulting PST is combined by all user PSTs contained in the current design or in its logical hierarchy.

If the PST has not been created, the power state table is full, meaning that any power state is allowed. Once the PST is created, the table becomes empty and new power states could be created and added to the table by the command of `add_pst_state`.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

SEE ALSO

`create_pst(2)`
`add_port_state(2)`
`add_pst_state(2)`

report_qor

Displays QoR information and statistics for the current design.

SYNTAX

```
status report_qor  
[-significant_digits digits]  
[-physical]  
[-scenario scenario_list]
```

Data Types

<i>digits</i>	integer
<i>scenario_list</i>	list

ARGUMENTS

-significant_digits *digits*
Specifies the number of digits to the right of the decimal point that are reported. Allowed values are from 0 through 13. The default is 2 for all sections except the Area section, which has a default of 6. Specifying a valid value for significant digits overwrites the existing default value. Using this option overrides the value set by the **report_default_significant_digits** variable.

-physical
Reports a QoR metrics snapshot.

-scenario *scenario_list*
Reports the QoR for the specified list of scenarios of a multi-scenario design. Inactive scenarios are skipped in the report. Each scenario is reported separately. If this option is not specified, the command reports the QoR on all scenarios.

DESCRIPTION

This command reports information on timing path group details and cell count, and current design statistics including combinational, non-combinational, and total area. The command also reports static power, design rule violations, and compile time details.

Under the Cell Count section, the Leaf Cell Count reported includes all leaf cells that are not constant cells. Constant cells are omitted in this count.

Multicorner-Multimode Support

This command uses information from active scenarios only.

This command supports an option to specify which active scenarios it should work with.

EXAMPLES

The following is an example of a QoR report:

```
prompt> report_qor
```

```
*****
Report : qor
Design : qhead
Version: V-2004.06
Date   : Wed Feb  4 21:01:55 2004
*****
```

Timing Path Group 'coreclk'

```
-----
Levels of Logic:          27.00
Critical Path Length:     4.33
Critical Path Slack:      -0.43
Total Negative Slack:     -54.67
No. of Violating Paths:   199.00
-----
```

Timing Path Group 'default'

```
-----
Levels of Logic:          9.00
Critical Path Length:     1.70
Critical Path Slack:      -0.20
Total Negative Slack:     -57.84
No. of Violating Paths:   389.00
-----
```

Cell Count

```
-----
Hierarchial Cell Count:    26
Hierarchial Port Count:    1898
Leaf Cell Count:           12916
-----
```

Area

```
-----
Combinational Area:       21062.576172
Noncombinational Area:    6550.667969
Net Area:                  76896.007812
-----
```

```
Cell Area:                 27613.437500
Design Area:               104509.445312
```

Design Rules

```
-----
Total Number of Nets:      13234
Nets With Violations:      481
-----
```

Hostname: n/a

Compile CPU Statistics

```
-----
Resource Sharing:          0.00
Logic Optimization:       0.00
Mapping Optimization:     0.00
-----
Overall Compile Time:     0.00
```

1

The following is an example of a QoR report using **-significant_digits** to show 4 digits to the right of the decimal point:

```
prompt> report_qor -significant_digits 4
```

```
*****
Report : area
Report : qor
Design : qhead
Version: V-2004.06
Date   : Mon Jan 19 03:17:57 2004
*****
```

Timing Path Group 'coreclk'

```
-----
Levels of Logic:          27.0000
Critical Path Length:     4.3301
Critical Path Slack:      -0.4301
Total Negative Slack:     -54.6667
No. of Violating Paths:   199.0000
-----
```

Timing Path Group 'default'

```
-----
Levels of Logic:          9.0000
Critical Path Length:     1.7038
Critical Path Slack:      -0.2038
Total Negative Slack:     -57.8426
No. of Violating Paths:   389.0000
-----
```

Cell Count

```
-----
Hierarchial Cell Count:   26
Hierarchial Port Count:   1898
Leaf Cell Count:          12916
-----
```

Area

```
-----
Combinational Area:       21062.6289
Noncombinational Area:    6550.6680
Net Area:                  76896.0078
```

```
-----  
Cell Area:                27613.4375  
Design Area:              104509.4453  
-----
```

Design Rules

```
-----  
Total Number of Nets:      13234  
Nets With Violations:      481  
-----
```

Hostname: n/a

Compile CPU Statistics

```
-----  
Resource Sharing:          0.0000  
Logic Optimization:        0.0000  
Mapping Optimization:      0.0000  
-----  
Overall Compile Time:      0.0000  
-----
```

1

SEE ALSO

`report_default_significant_digits(3)`

report_qor_snapshot

Reports existing QoR snapshot of timing, DRC, area, clock, power, etc. The report utility gets snapshot information from the location specified by the **icc_snapshot_storage_location** variable or from the specified location from the -directory option. The output of the report will be stored at the location specified by the **icc_snapshot_storage_location** variable.

NOTE: This is the new version of the **report_qor_snapshot** command that supports both multicorner-multimode designs as well as non-multicorner-multimode designs.

SYNTAX

```
void report_qor_snapshot  
[-name name]  
[-directory directory]  
[-display]  
[-save_as report_name]
```

Data Types

name string

ARGUMENTS

-name *name*
 Specifies the name of the QoR snapshot to be reported.

-directory *directory*
 Specify the location(s) of the snapshot. This option will take a list of directories.

-display
 An interactive browser will pop up and allow users to navigate through individual snapshot(s). The interactive browser is defined by variable *gui_online_browser*.

-save_as *report_name*
 Specify the name of the report. The report will be saved under this name, users then can access the report at a later time.

DESCRIPTION

This command reports all of the available QoR snapshots under *design_dir/snapshot*. If no snapshot exists, a message is displayed.

The **-name** option specifies which snapshot to be reported. If **-name** is not specified, all saved snapshots are shown.

Multicorner-Multimode Support

This command uses information from active scenarios only.

EXAMPLES

In the following example, **report_qor_snapshot** will report QoR snapshot named *preroute* which is saved under *design/snapshot* directory.

```
prompt> report_qor_snapshot -name preroute
```

SEE ALSO

```
create_qor_snapshot(2)  
remove_qor_snapshot(2)  
gui_online_browser(3)  
icc_snapshot_storage_location(3)
```

report_qtm_model

Reports Quick Timing Model (QTM) data.

SYNTAX

```
string report_qtm_model  
[-global_parameters]  
[-ports]  
[-arcs]
```

ARGUMENTS

-global_parameters
Specifies that global parameters of the current QTM design be reported.

-ports
Specifies that ports of the current QTM design be reported.

-arcs
Specifies that timing arcs of the current QTM design be reported.

DESCRIPTION

The **report_qtm_model** command prints a report of the active QTM model. The information details the global parameters, ports, and arcs of the QTM model. By default, the command prints out the report of global parameters, ports, and arcs of the model.

If you want to see the global parameters in the QTM model, use the **-global_parameters** option.

If you want to see only the ports in the QTM model, use the **-ports** option.

If you want to see the arcs in the QTM model, use the **-arcs** option.

For a basic description of QTM, see the **create_qtm_model** manual page. For a more detailed description of QTM, see the *ICC Design Planning User Guide*.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following command reports all the attributes of the currently active QTM model.

```
prompt> report_qtm_model  
  
*****
```

```
Report : qtm_model
Design : test_lib
*****
```

```
create_qtm_model test_lib
set_qtm_technology -library slow -max_transition 1.0
set_qtm_global_parameter -param clk_to_output -value 1.0
set_qtm_global_parameter -param setup -value 0.5
set_qtm_global_parameter -param hold -value 0.0
create_qtm_path_type -lib_cell AN210 -input_pin A -output_pin Y -fanout 2 path1
create_qtm_drive_type -lib_cell AN210 -input_pin A -output_pin Y drive1
create_qtm_load_type -lib_cell AN210 -input_pin A load1
create_qtm_port -type clock {CLK}
create_qtm_port -type input {A}
create_qtm_port -type input {B}
create_qtm_port -type input {X}
create_qtm_port -type input {Y}
set_qtm_port_load -type load1 -factor 2 {A}
set_qtm_port_load -value 1.0 {B}
set_qtm_port_drive -type drive1 -type drive1 {X}
set_qtm_port_drive -value 1.0 {Y}
create_qtm_constraint_arc -setup -from CLK -edge rise -value 1.0 -to {A}
create_qtm_constraint_arc -setup -from CLK -edge rise -path_type path1 -
path_factor 2 -to {A}
create_qtm_delay_arc -from {CLK} -to {X} -edge rise -value 1.0
create_qtm_delay_arc -from {CLK} -to {Y} -edge rise -path_type path1 -
path_factor 7
report_qtm_model
```

The following command reports the global parameters of the currently active QTM model.

```
prompt> report_qtm_model -global_parameters
```

```
*****
Report : qtm_model
Design : test_lib
*****
```

```
create_qtm_model test_lib
set_qtm_technology -library slow -max_transition 1.0
set_qtm_global_parameter -param clk_to_output -value 1.0
set_qtm_global_parameter -param setup -value 0.5
set_qtm_global_parameter -param hold -value 0.0
create_qtm_path_type -lib_cell AN210 -input_pin A -output_pin Y -fanout 2 path1
create_qtm_drive_type -lib_cell AN210 -input_pin A -output_pin Y drive1
create_qtm_load_type -lib_cell AN210 -input_pin A load1
report_qtm_model
```

SEE ALSO

`create_qtm_model(2)`
`save_qtm_model(2)`

report_rail_options

Reports the current option settings for the **analyze_rail** command.

SYNTAX

report_rail_options

ARGUMENTS

This command has no arguments.

DESCRIPTION

Prints out the current option settings for the **analyze_rail** command.

EXAMPLE

The following example reports the current option settings for the **analyze_rail** command:

```
prompt> report_rail_options
```

```
*****
Report : rail_options
Design : micro
Version: C-2009.06
Date   : Mon Feb 23 23:12:55 2009
*****
```

Current rail options

```
-----
output_dir = /myDesign/myWorkingDirectory/PrimeRail_run
pad_master_file = ""
pad_instance_file = ""
use_pins_as_pads = true
user_defined_tap_file = /myDesign/input_data/vdd.tap
packaging_file = ""
switching_activity =
pr_exec_dir = ""
pt_exec_dir = ""
host = ""
sdc = ""
spef = ""
verilog = ""
upf = ""
vd_threshold = ""
```

SEE ALSO

set_rail_options(2)

report_rail_options

analyze_rail(2)

report_reference

Displays information about references in the current instance or in the current design.

SYNTAX

```
integer report_reference  
[-nosplit]  
[-hierarchy]
```

ARGUMENTS

-nosplit

Prevents line-splitting and facilitates writing software to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

-hierarchy

Displays information about references across the hierarchy in the current instance or the current design.

DESCRIPTION

Displays information about all references in the current instance or the current design. If the current instance has been set, the report is generated for the design of that instance. Otherwise the report is generated for the current design.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following is an example of a reference report:

```
prompt> report_reference  
  
*****  
Report : reference  
Design : CONTROL  
Version: Y-2006.06  
Date   : Mon Dec 12 14:09:33 2005  
*****  
  
Attributes:  
  b - black box (unknown)  
  bo - boundary optimization
```


d - dont_touch
 h - hierarchical
 n - noncombinational
 r - removable
 s - synthetic module
 u - contains unmapped logic

Reference	Library	Unit Area	Count	Total Area	Attributes
CTLX		101.00	1	101.00	h
INV	tech_lib	1.00	1	1.00	
JMM		0.00	0	0.00	b, d
JMM1		0.00	0	0.00	b
JPMP		0.00	1	0.00	h, u
MX1P	tech_lib	8.00	8	64.00	n
NAND2	tech_lib	1.00	6	6.00	
NAND3	tech_lib	2.00	1	2.00	
Total 8 references				174.00	

The following is an example of a reference report across a hierarchy:

prompt> **report_reference -hierarchy**

 Report : reference
 Design : top
 Version: Y-2006.06
 Date : Mon Dec 12 04:25:17 2005

Attributes:

b - black box (unknown)
 bo - boundary optimization
 d - dont_touch
 h - hierarchical
 n - noncombinational
 r - removable
 s - synthetic module
 u - contains unmapped logic

Reference	Library	Unit Area	Count	Total Area	Attributes
AN3	tech_lib	2.00	8	16.00	
mid_0		56.00	1	56.00	h, n
Total 2 references				72.00	

 Design: mid_0

Reference	Library	Unit Area	Count	Total Area	Attributes
low_0		28.00	1	28.00	h, n

```

low_5                28.00          1      28.00      h, n
-----
Total 2 references                                56.00

*****
Design: low_0
*****
Reference      Library      Unit Area   Count     Total Area   Attributes
-----
FD1            tech_lib      7.00        4        28.00        n
-----
Total 1 references                                28.00

*****
Design: low_5
*****
Reference      Library      Unit Area   Count     Total Area   Attributes
-----
FD1            tech_lib      7.00        4        28.00        n
-----
Total 1 references                                28.00

```

SEE ALSO

report_cell(2)
report_design(2)

report_retention_cell

Displays information about retention cells in the current scope. This command is supported only in UPF mode.

SYNTAX

```
status report_retention_cell
[retention_cells]
[-domain power_domains]
[-strategy retention_strategy_names]
[-verbose]
```

Data Types

<i>retention_cells</i>	list
<i>power_domains</i>	list
<i>retention_strategy_names</i>	string

ARGUMENTS

retention_cells
Specifies the retention cells that are to be reported. If the specified cells do not exist in the current scope, the command fails. By default, all retention cells in the current scope are reported when no other options are specified.

-domain *power_domains*
Specifies the power domains to report all retention cells in the design belonging to the power domains. If the specified power domains do not exist in the current scope, the command fails.

-strategy *retention_strategy_names*
Specifies the names of the retention strategies to report all retention cells in the design that belong to the given retention strategies. When specifying the retention strategy, you must also specify a single power domain with the **-domain** option. If the specified retention strategy is not a part of the specified power domain, the command fails.

-verbose
Reports retention strategy details in addition to information about the retention cells.

DESCRIPTION

The **report_retention_cell** command provides detailed information of all retention cells in the current scope. If the **-domain** option is specified, the tool reports on all retention cells in the specified domains. If the **-strategy** option is specified the tool reports on all retention cells under the specified strategy for the given power domain.

The report shows the instance name of the retention cell, the library reference name

of the retention cell, and the retention strategy. If the **-verbose** option is specified, the report also shows the details of the retention strategy, which include the name of the retention strategy, the names of the save and restore signals, the save sense and restore sense values, and the names of the power and ground nets.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports the information about all of the retention cells in the current scope:

```
prompt> report_retention_cell

-----

Power Domain : PD1

=====
| Ret. Cell          | Ret. Lib. Cell          | Strategy          |
=====
| q2_reg             | RTCLDFFPQ_F1_SNPM      | retention_1       |
| q1_reg             | RTCLDFFPQ_F1_SNPM      | retention_1       |
=====

-----

1
```

The following example reports detailed information about all of the retention cells in the current scope:

```
prompt> report_retention_cell -verbose

-----

Power Domain : PD1

=====
| Ret. Cell          | Ret. Lib. Cell          | Strategy          |
=====
| regout_reg[0]      | RTCLDFFPQ_F1_SNPM      | retention_strategy_1 |
| regout_reg[1]      | RTCLDFFPQ_F1_SNPM      | retention_strategy_1 |
| regout_reg[2]      | RTCLDFFPQ_F1_SNPM      | retention_strategy_1 |
| regout_reg[3]      | RTCLDFFPQ_F1_SNPM      | retention_strategy_1 |
=====

=====
| Strategy          | Save | Save | Restore | Restore | Power | Ground |
=====
```

	Signal	Sense	Signal	Sense	Net	Net
retention_strategy_1	SAVE1	High	RESTORE1	High	SN1	-

1

SEE ALSO

```
map_retention_cell(2)
report_isolation_cell(2)
report_level_shifter(2)
set_retention(2)
set_retention_control(2)
```

report_route_opt_strategy

Reports the parameters that influence route_opt flow.

SYNTAX

```
int report_route_opt_strategy
```

ARGUMENTS

The **report_route_opt_strategy** command has no arguments.

DESCRIPTION

The **report_route_opt_strategy** command reports the parameters that influence route_opt in optimization and routing.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example runs the **report_route_opt_strategy** command.

```
prompt>set_route_opt_strategy -effort high
```

```
prompt> report_route_opt_strategy
```

```
***** route_opt strategy for the
design: Route Mode : Timing Fix Hold Mode : route_base Search-Repair loops : 10 ECO
Search-Repair loops : 5 Optimize Wire/Via Search-Repair loops : 5 Optimize Wire/Via
CPU limit : -1 Crosstalk Optimization loops : 3
*****
```

SEE ALSO

```
set_route_opt_strategy(2)
```

report_route_options

Reports route option settings. These settings are used by several routing commands.

SYNTAX

```
status report_route_options
[-default]
[-output file_name]
```

Data Types

file_name string

ARGUMENTS

-default
Reports the default settings of the route options.

-output *file_name*
Generates a Tcl script that contains **set_route_options** commands for the current route option settings. You can use this file to set the same route option settings in another session.

DESCRIPTION

This command reports the route option values that have been set by the **set_route_options** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows the use of the command.

```
prompt> report_route_options
```

SEE ALSO

```
route_opt(2)
set_route_options(2)
```

report_route_zrt_common_options

Reports the settings of route options that are common among the Zroute router commands.

SYNTAX

status **report_route_zrt_common_options**

ARGUMENTS

The **report_route_zrt_common_options** command has no arguments.

DESCRIPTION

This command reports the settings of router options that have been set by the **set_route_zrt_common_options** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command reports the settings of the common route options.

```
prompt> report_route_zrt_common_options
```

SEE ALSO

```
set_route_zrt_common_options(2)  
get_route_zrt_common_options(2)
```


report_route_zrt_detail_options

Reports the settings of the detail router options.

SYNTAX

status **report_route_zrt_detail_options**

ARGUMENTS

The **report_route_zrt_detail_options** command has no arguments.

DESCRIPTION

This command reports the settings of the detail router options that have been set by the **set_route_zrt_detail_options** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command reports the settings of the detail router options.

```
prompt> report_route_zrt_detail_options
```

SEE ALSO

```
get_route_zrt_detail_options(2)  
set_route_zrt_detail_options(2)
```

report_route_zrt_global_options

Reports the settings of the global router options.

SYNTAX

status **report_route_zrt_global_options**

ARGUMENTS

The **report_route_zrt_global_options** command has no arguments.

DESCRIPTION

This command reports the settings of the global router options that have been set by the **set_route_zrt_global_options** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports the global router settings.

```
prompt> report_route_zrt_global_options
```

SEE ALSO

```
get_route_zrt_global_options(2)  
set_route_zrt_global_options(2)
```

report_route_zrt_track_options

Reports the settings of track assignment options.

SYNTAX

status **report_route_zrt_track_options**

ARGUMENTS

None.

DESCRIPTION

This command reports the settings of track assignment options that have been previously set by using the **set_route_zrt_track_options** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports the track assignment options.

```
prompt> report_route_zrt_track_options
```

SEE ALSO

```
get_route_zrt_track_options(2)  
set_route_zrt_track_options(2)
```

report_routing_rules

Reports on design-specific nondefault routing rules defined by the **define_routing_rule** command.

SYNTAX

```
status report_routing_rules  
[rule_name]  
[-output file_name]
```

Data Types

<i>rule_name</i>	string
<i>file_name</i>	string

ARGUMENTS

rule_name
Specifies the name of the design-specific nondefault routing rule to report on. By default, the tool reports all design-specific nondefault routing rules.

-output *file_name*
Generates a Tcl script that contains the **define_routing_rule** commands that define the specified nondefault routing rules.

DESCRIPTION

This command reports on design-specific nondefault routing rules defined by the **define_routing_rule** command. It does not report rules defined in the physical library.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports on the design-specific nondefault rule named WideMetal.

```
prompt> report_routing_rules WideMetal
```

SEE ALSO

```
define_routing_rule(2)  
set_net_routing_rule(2)
```

report_rp_group_options

Reports relative placement group attributes on the specified relative placement groups.

SYNTAX

```
status report_rp_group_options  
rp_groups
```

Data Types

rp_groups collection or list

ARGUMENTS

rp_groups
Specifies the relative placement groups for which you want to report the attributes. If relative placement groups are not specified, the attributes for all relative placement groups will be reported.

DESCRIPTION

The **report_rp_group_options** command reports the attributes of the specified relative placement groups. The attributes were set by using either the **create_rp_group** or the **set_rp_group_options** command.

This command is supported only for designs that do not contain multiply-instantiated designs.

Relative placement usage is available with Design Compiler Ultra.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses **report_rp_group_options** to report the attributes for a relative placement group:

```
prompt> report_rp_group_options {example3::top_group example3::block2}
```

```
*****  
Report : The RP group options.  
Design : example3  
Version: Y-2006.06-ICC-SP1-DEV  
Date   : Fri Jun  9 00:58:54 2006  
*****
```

```
RP group:  example3::top_group
  utilization      : 1.000000
  ignore          : FALSE
  x_offset         : 0
  y_offset         : 0
  cts_option       : fixed_placement
RP group:  example3::top_group
  utilization      : 1.000000
  ignore          : FALSE
1
```

SEE ALSO

```
create_rp_group(2)
set_rp_group_options(2)
```

report_saif

Reports statistics on the switching activity annotation on the current design or instance.

SYNTAX

```
int report_saif  
[-hier]  
[-flat]  
[-type rtl | gate]  
[-rtl_saif]  
[-missing]  
[-only cell_or_net_list]  
[-annotated_flag]
```

Data Types

cell_or_net_list object_list

ARGUMENTS

-type rtl | gate

Specifies whether switching activity annotations are to be reported for objects annotated by an RTL backward SAIF file or a gate-level backward SAIF file.

An RTL backward SAIF file is generated using RTL simulation, and contains the switching activity of synthesis invariant objects. These are objects that do not change during synthesis, and include the design ports, and the outputs of sequential and three-state cells. Calling the **read_saif** command with **-type rtl** reports the switching activity annotation on design ports, sequential cell outputs, and three-state outputs. Use the **-type rtl** argument after reading an RTL backward SAIF.

A gate-level backward SAIF file is generated using gate-level simulation, and contains the switching activity of all design nets. Calling the **read_saif** command with **-type gate** reports the switching activity annotation on the design ports, nets, and leaf cell pins. Use the **-type gate** argument after reading a gate-level backward SAIF file.

When the **-type** argument is not used, the default **-type gate** is assumed.

-rtl_saif

Specifies that switching activity annotations are to be reported for objects annotated by an RTL backward SAIF file.

Specifying the **read_saif** command with the **-rtl_saif** flag is equivalent to running the command with **-type rtl**.

-missing

Specifies that the report lists the design elements that do not have user switching activity annotation. This report does not include constant value nets (logic one/zero nets) and pins and ports connected to such nets. This type of nets, pins, and ports are annotated with default switching activity if they are not user annotated.

`-only cell_or_net_list`

Specifies that switching activity annotation is to be reported only for the specified object list.

`-annotated_flag`

Specifies that the report list the design elements that have user-annotated switching activity.

DESCRIPTION

The **report_saif** command reports the switching activity annotation on the nets, ports, and cells in the current design or instance.

The **report_saif** command can be used to display switching activity annotation of synthesis invariant objects by using the **-rtl_saif** flag or the equivalent **-type rtl** argument. If these options are not used, then **report_saif** displays information on the design ports, nets, and non-hierarchical cell pins.

The report generated by **report_saif** lists the percentage of objects with switching activity that is user-annotated, default-annotated and propagated. Switching activity can be annotated by the user by the **set_switching_activity**, **read_saif**, and **merge_saif** commands.

During power calculations, the tool estimates the switching activity of objects that are not user-annotated by propagating the user or default annotated switching activity. Switching activity is annotated with default values on objects whose switching activity can be derived accurately, such as clocks and nets driven by constants, or on objects that cannot be annotated using the propagation mechanism, such as the design primary inputs and outputs of black-box cells. Switching activity is default annotated and propagated automatically by the **report_power** command.

Propagated switching activity is less accurate than switching activity obtained by simulation, so the percentage values in the user-annotated column of the report generated by **report_saif** are an indication of the accuracy of power reports.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The examples below report switching activity for the current design.

In the following example, the **report_saif** command is used after the **read_saif** command on a gate-level design. With the **-missing** option set, the report lists the nonannotated ports, nets, and pins.

```
prompt> report_saif -hier -missing
```

```
*****
```

```
Report : saif
```

```
-gate
```



```

-flat
-missing
Design : addr_con
Version: 2003.12
Date   : Tue Apr  8 13:04:04 2003
*****

```

Object type	User Annotated (%)	Default Annotated (%)	Propagated Activity (%)	Total
Nets	251(99.21%)	1(0.40%)	1(0.40%)	253
Ports	59(98.33%)	1(1.67%)	0(0.00%)	60
Pins	251(99.60%)	0(0.00%)	1(0.40%)	252

List of nonannotated ports :

```
mem_config(0)
```

List of nonannotated nets :

```
n677
net41
```

List of nonannotated pins :

```
U519/A
```

In the following example, pin and net annotations are displayed only for the cell {last_addr_regx6x}. One of the cell pins and the corresponding net are not annotated.

```
prompt> report_saif -missing -only {last_addr_regx6x}
```

```

*****
Report : saif
-missing
-only
Design : addr_con
Version: 2003.12
Date   : Tue Apr  8 13:03:59 2003
*****

```

Object type	User Annotated (%)	Default Annotated (%)	Propagated Activity (%)	Total
Nets	3(75.00%)	0(0.00%)	0(0.00%)	4
Ports	0(0.00%)	0(0.00%)	0(0.00%)	0
Pins	3(75.00%)	0(0.00%)	0(0.00%)	4

List of nonannotated nets :

n677

List of nonannotated pins :

last_addr_regx6x/Q

The following example generates a switching activity report on synthesis invariant objects:

```
prompt> report_saif -rtl
```

```
*****
Report : saif
        -rtl_saif
Design : cpu
Version: 2003.12
Date   : Tue Apr  8 13:03:59 2003
*****
```

Object type	User Annotated (%)	Default Annotated (%)	Propagated Activity (%)	Total

Ports	13 (100.00%)	0 (0.00%)	0 (0.00%)	13
Seq Cells	620 (99.68%)	0 (0.00%)	0 (0.00%)	622
Tri Cells	36 (90.00%)	0 (0.00%)	0 (0.00%)	40

SEE ALSO

```
read_saif(2)
report_power(2)
set_switching_activity(2)
```

report_scan_chain

Reports the scan chains defined on the current design.

SYNTAX

status **report_scan_chain**

ARGUMENTS

The **report_scan_chain** command has no arguments.

DESCRIPTION

This command reports all scan chain information. The chains are defined on the current open CEL.

Using SCANDEF data, the report displays the chain's start and the stop port or pin, partition label, validation status, and all scan chain components. The scan chain components are shown in the order in which they appear in the chain of the netlist, from the START point to the STOP point.

The status can be NOT YET VALIDATED, VALIDATED/V or FAILED/F.

- NOT YET VALIDATED indicates that the **check_scan_chain** command has not been run to determine the status.
- VALIDATED indicates that the SCANDEF data in the report is consistent with the netlist.
- FAILED indicates an inconsistency between the SCANDEF data and the netlist.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following is an example of a scan chain report:

```
prompt> report_scan_chain

*****
Report : Scan Chain
Design : CORE
Version: W-2004.12
Date   : Thu Jun  2 14:02:00 2005

*****
```

```
SCANCHAIN    : ScanGroup_zw_test_test_si1
START        : test_si1
STOP         : U1/LOCKUP/D
PARTITION    : CLK1_45_45
STATUS       : VALIDATED
```

```
-----
U1/scrs_reg  ( IN D1 ) ( OUT Q )
U1/dcrs_reg  ( IN D1 ) ( OUT Q )
```

```
SCANCHAIN    : ScanGroup_U1/LOCKUP_Q
START        : U1/LOCKUP/Q
STOP         : U1/LOCKUP1/D
PARTITION    : CLK1_45_45
STATUS       : VALIDATED
```

```
-----
U1/rrst_L_reg ( IN D1 ) ( OUT Q )
U1/stxen_reg  ( IN D1 ) ( OUT Q )
```

1

SEE ALSO

```
check_scan_chain(2)
optimize_dft(2)
read_def(2)
trace_scan_chain(2)
```

report_scenario_options

Reports the scenario options set by the **set_scenario_options** command.

SYNTAX

```
status report_scenario_options  
[-scenarios scenario_list]
```

ARGUMENTS

```
-scenarios scenario_list  
    Specifies a list of scenarios, active or inactive, to be reported. By default,  
    the command reports scenario options for the current scenario.
```

DESCRIPTION

This command reports the values of scenario options set using the **set_scenario_options** command, for the current scenario or the scenarios listed using the **-scenarios** option. This command can report scenario options for inactive scenarios as well, except for `leakage_only` and `hold_only` scenarios.

EXAMPLES

The following example uses **report_scenario_options** command to check the settings for the current scenario:

```
prompt> create_scenario MODE1  
Warning: Discarding all scenario specific information previously defined in  
this session. (UID-1008)  
Current scenario is: MODE1  
1  
  
prompt> set_scenario_options -setup false  
1  
  
prompt> report_scenarios_options  
  
*****  
Report : scenario options  
Design : TEST03  
Version: C-2009.06  
Date   : Wed Mar 19 14:57:08 2008  
*****  
  
Scenario: MODE1 is active.  
  
leakage_only : false  
hold_only   : false  
setup       : false  
hold        : true
```

```
leakage_power : true
dynamic_power : true
```

The following example uses the **report_scenario_options** command to report all scenarios:

```
prompt> create_scenario MODE1
Warning: Discarding all scenario specific information previously defined in
this session. (UID-1008)
Current scenario is: MODE1
1

prompt> set_scenario_options -hold_only true
1
prompt> create_scenario MODE2
Current scenario is: MODE2
1

prompt> set_scenario_options -hold false
1

prompt> report_scenario_options -scenarios [all_scenarios]

*****
Report : scenario options
Design : TEST03
Version: C-2009.06
Date   : Wed Mar 19 14:57:08 2008
*****

Scenario: MODE1 is active.

leakage_only : false
hold_only    : true
setup        : false
hold         : true
leakage_power : true
dynamic_power : true

Scenario: MODE2 is active.

leakage_only : false
hold_only    : false
setup        : true
hold         : false
leakage_power : true
dynamic_power : true
```

SEE ALSO

```
set_scenario_options(2)
report_scenarios(2)
```

```
create_scenario(2)
current_scenario(2)
all_scenarios(2)
```

report_scenarios

Reports scenario setup information for a multi-scenario design.

SYNTAX

status **report_scenarios**

ARGUMENTS

None.

DESCRIPTION

The **report_scenarios** command reports scenario setup information for a multi-scenario design. It reports all defined scenarios, all active scenarios, the current scenario, the CTS scenario, and the leakage-only scenario setting information, as well as other scenario-specific information for active scenarios. The scenario-specific information includes the library used in linking for the scenario, the design operating condition for the scenario, and the TLUPlus files for the scenario.

Multicorner-Multimode Support

This command uses information from all scenarios.

EXAMPLES

The following example shows sample output of the **report_scenarios** command.

```
prompt> report_scenarios

*****
Report : scenarios
Design : TCS03
Scenario(s): MODE1
Version: Version: A-2007.12-ICC
Date   : Wed Mar 19 14:57:08 2008
*****

All scenarios (Total=1): MODE1
All Active scenarios (Total=1): MODE1
Current scenario      : MODE1
CTS scenario          : not defined.

Scenario #0: MODE1 is active.
Scenario options: Leakage-only
Has timing derate: No

Library(s) Used:
  cb_max (File: /remote/lib/cb_max.db)
```


Operating condition(s) Used:
Analysis Type : bc_wc
Max Operating Condition: cb_max:cb_max
Max Process : 1.20
Max Voltage : 1.08
Max Temperature: 125.00
Min Operating Condition: cb_max:cb_max
Min Process : 1.20
Min Voltage : 1.08
Min Temperature: 125.00

TLU+ Files Used:
There is no TLU+ settings available.

Number of leakage-only scenario(s): 1
Number of hold-only scenario(s): 0
Warning: All the active scenarios are leakage-only scenario.
This is not allowed. Please check your settings!
1

SEE ALSO

`create_scenario(2)`
`set_scenario_options(2)`

report_si_options

Reports signal integrity options used for analysis or optimization.

SYNTAX

```
int report_si_options
```

ARGUMENTS

None.

DESCRIPTION

This command reports signal integrity options used for analysis or optimization.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows how to use this command.

```
prompt> report_si_options
*****
Report : si options
Design : some_design
Version: C-2009.06-ICC-INTERNAL
Date   : Wed Mar  4 22:24:04 2009
*****

*****
SI options:
Delta Delay Computation:           true
Static Noise Computation:         false
Timing window analysis for SI:    false
Reselection in timing window:     false
Min Delta Delay for SI:           true
Analysis Effort:                  low
Max Transition Mode:              normal_slow
Static Noise Thresholds:          0.35 (0.57V) above low
                                   0.35 (0.57V)
                                   below high
Route xtalk prevention:           false
Route xtalk prevention threshold: 0.35
*****
```

SEE ALSO

`set_si_options(2)`

report_signal_em

Reports signal electromigration results for the violating or specified nets. Also generates a repair file consisting of variable routing rules that can be loaded into the database for rerouting.

SYNTAX

```
status report_signal_em
[-healing_factor healing_factor_value]
[-violation_rule_type mean | abs_avg | rms | peak | auto]
[-repair_file repair_file_name]
[-min]
[-max]
[-violated]
[-verbose]
[list_of_nets]
```

ARGUMENTS

- healing_factor** *healing_factor_value*
The healing factor is used for calculating the average current. The default is 1.0.
- violation_rule_type** mean | abs_avg | rms | peak | auto
Reports the current density values using only the specified rule. If **auto** is selected, the report includes all the rules for which there are constraints defined in the database. The default is **auto**.
- repair_file** *repair_file_name*
Specifies the name of the file to which you want to write the variable routing rules for interconnect electromigration repair.
- min**
Uses minimum corner timing values to calculate the current density. If neither **-min** nor **-max** is specified, both minimum and maximum values are calculated.
- max**
Uses maximum corner timing values to calculate current density. If neither **-min** nor **-max** is specified, both minimum and maximum values are calculated.
- violated**
Reports only violating nets. If you specify the *list_of_nets* argument, it reports only the violating nets among the specified nets. Otherwise, it reports all the violating nets in the design.
- verbose**
Reports additional information for all specified nets or violating nets, including layer name, Rel_AbsAvg, and Rel_RMS.

DESCRIPTION

The **report_signal_em** command provides a report of signal (interconnect) electromigration information for the current design.

The command performs electromigration analysis of each net by calculating the current on every edge of the net and comparing it with the current thresholds defined in the design library. It also generates a repair file consisting of variable routing rules that can be loaded into the database for rerouting.

Current density information is not saved as net attribute. Each time you run the **report_signal_em** command, the tool calculates the current density for each specified net.

Before running **report_signal_em**, make sure the design has signal electromigration constraints defined in the Milkyway design library. You can use Library Compiler or the Milkyway tool to load a

Also, be sure that switching activity has been defined for all boundary nets.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example shows a signal electromigration report for net "n1" using both minimum and maximum corners. The rule type is RMS and the repair file name is "foo".

```
prompt> report_signal_em -repair "foo" [get_nets n1]
```

SEE ALSO

```
read_saif(2)  
set_switching_activity(2)
```

report_skew_group

Reports the user-defined skew groups in the design.

SYNTAX

```
status report_skew_group
[-clock clock_name]
[-name skew_group_name]
```

Data Types

<i>clock_name</i>	string
<i>skew_group_name</i>	string

ARGUMENTS

-clock *clock_name*
Reports only the skew groups that are on the specified clock path.

-name *skew_group_name*
Reports the skew group parameters of the specified skew group.

DESCRIPTION

The command reports the user-defined skew groups in the design. If you specify the **-clock** option, the command reports the skew groups that are on the specified clock path. To report the skew group with a specific name, use the **-name** option.

By default, the command lists all user-defined skew groups in the design.

EXAMPLES

The following example lists all skew groups that are on clock path clk1.

```
prompt> report_skew_group -clock clk1
CTS: Marking Ignore Pins....

*****
Report      : Skew group
Design      : Hier_design
Version     : C-2009.06-ICC-INTERNAL
Date        : Wed Feb 25 04:53:07 2009
*****

=====Skew Group Data=====

1. Skew Group Name : A
   Target Delay     : 0.000
   Target Skew      : 0.000
```

```
Pin List : 10
  A/B/C/FF4/CLK
  A/B/C/FF3/CLK
Clock List :
  clk1
```

```
2. Skew Group Name : B
Target Delay      : 0.000
Target Skew       : 0.000
Pin List : 10
  A/B/C/FF1/CLK
  A/B/C/FF2/CLK
Clock List :
  clk1
```

1

SEE ALSO

```
set_skew_group(2)
remove_skew_group(2)
```

report_spacing_rules

Reports inter-cell spacing rules from the library.

SYNTAX

```
status report_spacing_rules
-all
  | -of_library_cells {library_cell_collection}
```

ARGUMENTS

-all
Reports all inter-cell spacing rules in the current design.

DESCRIPTION

This command reports inter-cell spacing rules that have been set with commands `set_lib_cell_spacing_label` and `set_spacing_label_rule` in the current design. One option of `-all` or `-of_library_cells` is required.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows inter-cell spacing report

```
prompt> report_spacing_rules -all

*****
* Report: inter-cell spacing constraints
* Design: MEM
* Date: Wed May 30 11:22:19 2007
*****

  left lib_cell  orient  right lib_cell  orient  illegal spacing
-----
          OR4X4      N          OR4X2      FN          1-2
          AND4X2      N          AND2X2      FN           0
          AND2X2      N          NOR2X4       N          3-5 7-9
          AND3X2      N          OR2X4       FN          1-4 7-9
          NAND3BX1    FN          NOR2X4       N          0 3 5-8
-----
Info: 5 inter-cell spacing rules found
```

The following example shows inter-cell spacing report for library cell AND2X2.

```
prompt> report_spacing_rules -or_library_cells {AND2X2}
```



```
*****
* Report: inter-cell spacing constraints
* Design: MEM
* Date: Wed May 30 11:22:19 2007
*****
```

left lib_cell	orient	right lib_cell	orient	illegal spacing
AND4X2	N	AND2X2	FN	0
AND2X2	N	NOR2X4	N	3-5 7-9

Info: 2 inter-cell spacing rules found

SEE ALSO

```
set_lib_cell_spacing_label(2)
set_spacing_label_rule(2)
remove_all_spacing_rules(2)
```

report_split_clock_gates_options

Reports the option settings of the **split_clock_gates** command.

SYNTAX

status **report_split_clock_gates_options**

ARGUMENTS

The **report_split_clock_gates_options** command has no arguments.

DESCRIPTION

This command reports the option settings of the **split_clock_gates** command.

SEE ALSO

split_clock_gates(2)
set_split_clock_gates_options(2)
reset_split_clock_gates_options(2)

report_starrcxt_options

Reports StarRC options for the **signoff_opt** command.

SYNTAX

status **report_starrcxt_options**

ARGUMENTS

This command does not take any arguments.

DESCRIPTION

report_starrcxt_options command to examine the options which are set by **set_starrcxt_options** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example runs the **report_starrcxt_options** command.

```
prompt> open mw_cel -lib my_lib my_cel
prompt> report_starrcxt_options

*****
Report : StarRCXT_options
Design : my_design
Version: X-2005.12-ICC
Date   : Mon Aug 15 15:55:58 2005
*****

StarRCXT exec dir: /StarRCXT/linux/bin
StarRCXT max nxtgrd file: /design/max.nxtgrd
StarRCXT map file: /design/map_file
```

SEE ALSO

set_starrcxt_options(2)
signoff_opt(2)
place_opt(2)
clock_opt(2)
route_opt(2)

report_supply_net

Reports all the supply nets in the current scope. This command is supported only in UPF mode.

SYNTAX

```
status report_supply_net
[-include_exception]
```

ARGUMENTS

```
-include_exception
    Includes the exception connections of the supply net.
```

DESCRIPTION

The **report_supply_net** command reports detailed information about the supply nets in the current scope. The report for a supply net includes the following information: its full name, the scope in which it was created, its maximum and minimum voltages if set using the **set_voltage** command, its resolve type, the various supply states and the exception pins on the power net if the **-include_exception** option is specified. The names of the power domains to which the supply net belongs are also printed. The name of the power domain is suffixed with an asterisk (*) if the net is set as the domain supply net for that domain.

If the voltage of the supply net is not set, a '- U -' is printed, indicating that the voltage is undefined.

This command returns 1 on success, 0 otherwise.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports information about the supply nets SN1 in the current scope.

```
prompt> report_supply_net
```

```
*****
Report : supply_net
Design : top
Version: A-2007.12-SP1
Date   : Wed Nov 21 03:15:21 2007
*****
```

Supply Net : SN1
Current Scope : top
Operating Voltage : -- Best Case -- -- Worst Case --
 (1.00) (1.33)

Supply States : active_state, off_state
Resolve Type : unresolved
Exception Connections : I0/PWR, I1/PWR, PD0_INST/I1/PWR
Power Domain : T A(*)

1

SEE ALSO

create_supply_net(2)
create_power_domain(2)
connect_supply_net(2)
set_domain_supply_net(2)

report_supply_port

Reports information about the supply ports in the current scope. This command is supported only in UPF mode.

SYNTAX

status **report_supply_port**

ARGUMENTS

The **report_supply_port** command has no arguments.

DESCRIPTION

The **report_supply_port** command enables you to get detailed information about supply ports within the current scope. If *supply_port_name* is not specified, the tool reports on all supply ports within the current scope.

The report includes the full name of the supply port, the scope in which it is created, the direction, the supply states, and the supply net to which it is connected.

Supply ports that are present on switches can be printed by specifying the argument with the switch path name, slash (/), and supply port name as follows:

```
report_supply_port switch_full_path_name/supply_port_name
```

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports on all supply ports in the current scope:

```
prompt> report_supply_port
```

```
*****
Report : supply_port
Design : top
Version: A-2007.12-SP1
Date   : Wed Nov 21 03:14:25 2007
*****
```

```
-----
Supply Port      : SP1
Current Scope    : top
```

```
Direction      : Input
Supply net     : S_NET_1
Supply state names : active_state off_state
```

1

The following example reports on a supply port on a power switch:

```
prompt> report_supply_port SWA/outport
```

```
*****
Report : supply_port
Design : top
Version: A-2007.12-SP3
Date   : Thu Mar 13 21:50:40 2008
*****
```

```
-----

Supply Port      : SWA/outport
Current Scope    : mid1
Direction        : Output
Supply net       : No supply net connected to this port
Supply state names : No supply states defined for the port

-----
```

1

SEE ALSO

`create_supply_port(2)`

report_target_library_subset

Reports target library subsets on the design.

SYNTAX

```
int report_target_library_subset  
[-object_list cells]  
[-top]
```

ARGUMENTS

-object_list *cells*
Specifies the cells on which the target library subset will be reported.

-top
If specified, the target library subset being specified on the root design is reported.

DESCRIPTION

Reports the target library subset on the design based on the specified options. If none of the two options are specified, it reports the target library subset setting on the entire design.

NOTE: it only reports the target library subset information which is explicitly set by the user, does not report the inherited target library subset information, does not report the information from `-milkyway_reflibs` from `set_target_library_subset`.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports the target library subset for the design and for block u1.

```
prompt> report_target_library_subset -top -object_list [get_cells u1]
```

SEE ALSO

```
set_target_library_subset(2)  
remove_target_library_subset(2)  
check_target_library_subset(2)  
set_operating_conditions(2)  
target_library(3)
```


report_threshold_voltage_group

Reports the percentage of cells for each threshold voltage group in the design.

SYNTAX

```
int report_threshold_voltage_group  
[cell_or_design_list]  
[-verbose]
```

Data Types

cell_or_design_list list

ARGUMENTS

cell_or_design_list
Specifies a list of cells or designs on which to report the threshold voltage groups. If not specified, the report is generated on the current design.

-verbose
Lists all of the cells belonging to each threshold voltage group. If this option is omitted, the default summary report shows only the number and percentage of cells.

DESCRIPTION

This command reports the number and percentage of cells in each threshold voltage group for the specified list of cells or designs. The threshold voltage group is defined in the technology libraries with the **threshold_voltage_group** attribute on the cell, or with the **default_threshold_voltage_group** attribute on the library. You can also set the attributes on the objects using **set_attribute** command. The attributes can be any string values, but the values must be related to the threshold voltage of the cell to be meaningful.

If **-verbose** is specified, the command lists all of the cells that belong to each threshold voltage group. Without the option, the tool generates a summary that reports only the number and percentage of cells.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports the summary of cells in each threshold voltage group for the current design:

```
prompt> report_threshold_voltage_group
```

```

*****
Threshold Voltage Group Report

Threshold Voltage Group      Number of Cells      Percentage
*****
hvt                          588                  46.41%
lvt                          677                  53.43%
*****

```

SEE ALSO

```

set_attribute(2)
set_max_leakage_power(2)

```

report_tie_nets

Report tie high and tie low nets in the current design.

SYNTAX

```
status report_tie_nets
```

ARGUMENTS

The **report_tie_nets** command has no arguments.

DESCRIPTION

The **report_tie_nets** command reports all tie high and tie low nets in the current design. The command reports the number of tie high nets and the number of tie low nets, also reports the connected pin and port for each tie net.

EXAMPLES

The following example shows the output of **report_tie_nets** command.

Design has 2 tie high net(s):

```
net tin3 connects with following port(s) or pin(s):  
  tin3  
  leftop/A
```

```
net SNPS_LOGIC1 connects with following port(s) or pin(s):  
  mid1/min2
```

Design has 2 tie low net(s):

```
net mid1/SNPS_LOGIC0 connects with following port(s) or pin(s):  
  mid1/bot1/bin1  
  mid1/S2/B
```

```
net SNPS_LOGIC0 connects with following port(s) or pin(s):  
  leftop/B
```

report_timing

Displays timing information about a design.

SYNTAX

```
status report_timing
[-to to_list]
  | -rise_to rise_to_list
  | -fall_to fall_to_list]
[-from from_list
  | -rise_from rise_from_list
  | -fall_from fall_from_list]
[-through through_list]
[-rise_through rise_through_list]
[-fall_through fall_through_list]
[-path short | full | full_clock | full_clock_expanded | only | end]
[-delay min | min_rise | min_fall | max | max_rise | max_fall]
[-nworst paths_per_endpoint]
[-max_paths max_path_count]
[-input_pins]
[-nets]
[-transition_time]
[-crosstalk_delta]
[-capacitance]
[-attributes]
[-physical]
[-slack_greater_than greater_slack_limit]
[-slack_lesser_than lesser_slack_limit]
[-lesser_path max_path_delay]
[-greater_path min_path_delay]
[-loops]
[-true [-true_threshold path_delay]]
[-justify]
[-enable_preset_clear_arcs]
[-significant_digits digits]
[-nosplit]
[-sort_by group | slack]
[-group group_name]
[-trace_latch_borrow]
[-derate]
[-scenario scenario_list]
[-temperature]
[-voltage]
```

Data Types

<i>to_list</i>	list
<i>rise_to_list</i>	list
<i>fall_to_list</i>	list
<i>from_list</i>	list
<i>through_list</i>	list
<i>rise_through_list</i>	list
<i>fall_through_list</i>	list

<i>paths_per_endpoint</i>	integer
<i>max_path_count</i>	integer
<i>greater_slack_limit</i>	float
<i>lesser_slack_limit</i>	float
<i>max_path_delay</i>	float
<i>min_path_delay</i>	float
<i>path_delay</i>	float
<i>digits</i>	integer
<i>group_name</i>	string
<i>scenario_list</i>	list

ARGUMENTS

-to to_list

Reports only the paths to the named pins, ports, or clocks. If you do not specify **-to**, the default is to report the longest path to an output port if the design has no timing constraints. If the design has timing constraints, the default is to report the path with the worst slack within each path group if **-group** is not present. If **-group** is given, the default is to report the path with the worst slack within the group specified by **-group**.

-rise_to rise_to_list

Same as the **-to** option, but applies only to paths **rising** at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths **captured** by **rising** edge of the clock at clock source, taking into account any logical inversions along the clock path.

-fall_to fall_to_list

Same as the **-to** option, but applies only to paths **falling** at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths **captured** by **falling** edge of the clock at the clock source, taking into account any logical inversions along the clock path.

-from from_list

Reports only the paths from the named pins, ports, or clocks. If you do not specify **-from**, the default is to report the longest path to an output port if the design has no timing constraints. If the design has timing constraints, the default is to report the path with the worst slack within each path group if **-group** is not present. If **-group** is given, the default is to report the path with the worst slack within the group specified by **-group**.

-rise_from rise_from_list

Same as the **-from** option, except that the path must **rise** from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths **launched** by **rising** edge of the clock at the clock source, taking into account any logical inversions along the clock path.

-fall_from fall_from_list

Same as the **-from** option, except that the path must **fall** from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths **launched** by **falling** edge of the clock at the clock source, taking into account any logical inversions

along the clock path.

-through *through_list*

Reports only paths that pass through the named pins, ports, or clocks. If you do not specify **-through**, the default is to report the longest path to an output port if the design has no timing constraints. If the design does have timing constraints, the default is to report the path with the worst slack within each path group if **-group** is not present. If **-group** is given, the default is to report the path with the worst slack within the group specified by **-group**.

If you specify **-through** only once, the tool reports only the paths that travel through one or more of the objects in the list. You can specify **-through** more than once in one command invocation. For a discussion of the use of multiple **-through** options, see the DESCRIPTION section.

-rise_through *rise_through_list*

This option is similar to the **-through** option, but applies only to paths with a **rising** transition at the specified objects. You can specify **-rise_through** more than once in a single command invocation. For a discussion of multiple **-through**, **-rise_through**, and **-fall_through** options, see the DESCRIPTION section.

-fall_through *fall_through_list*

This option is similar to the **-through** option, but applies only to paths with a **falling** transition at the specified objects. You can specify **-fall_through** more than once in a single command invocation. For a discussion of multiple **-through**, **-rise_through**, and **-fall_through** options, see the DESCRIPTION section.

-path *short | full | full_clock | full_clock_expanded | only | end*

Specifies how to display the timing path. By default, the full path is displayed. If **full_clock** is specified, the report is similar to the report you get with **full** but also reports the full clock paths for propagated clocks. If **full_clock_expanded** is specified, the report is similar to the report you get with **full_clock** but also reports the full clock path from the primary clock to the related generated clock source. If you specify **short**, only start and end points are displayed. If you specify **only**, only the path is displayed without the accompanying required-time and slack calculation. If you specify **end**, the report has a column format that shows one line for each path, showing only the endpoint path total, required-time, and slack.

-delay *min | min_rise | min_fall | max | max_rise | max_fall*

Specifies the path type at the endpoint. The default is **max**.

-nworst *paths_per_endpoint*

Specifies the number of paths to report per endpoint. The default value is 1.

-max_paths *max_path_count*

Specifies the number of paths to report per path group. The default value is 1.

-input_pins

Shows input pins in the path report. The default is to show only output pins. This option also shows the delays of the nets connected to these pins.

-nets

Shows nets in the path report. The default is not to show nets. To show the delay for the nets, use the **-input_pins** option.

-transition_time

Shows the net transition time for each driving pin in the path report. The default is not to show the net transition time for each driving pin.

-crosstalk_delta

Shows the delta delay for each input pin in the path report. The default is not to show the delta delay for each input pin.

-capacitance

Indicates that total (lump) capacitance be shown in the path report. The default is not to show capacitance. For each driver pin, the total capacitance driven by the driver is displayed in a column preceding both incremental path delay and transition time (specified with **-transition_time**). When **-nets** is specified, the capacitance is printed on the lines with nets instead of the lines with driver pins.

-attributes

Shows the attributes specified in the **timing_report_attributes** variable. The current set of attributes supported are **dont_touch**, **dont_use**, **map_only**, and **size_only** for cells, and **dont_touch** and **ideal_net** for nets.

-physical

Shows the locations of the pins and the capacitive loads for the pins and nets in the path report. The loads are displayed as a pair of values with the first value being the wire capacitance of the net and the second value being the total capacitance driven by the driver. If the pin location cannot be determined, the cell location is displayed, with the coordinates in microns.

-slack_greater_than *greater_slack_limit*

Specifies that only those paths with a slack greater than *greater_slack_limit* are to be reported. This option can be combined with **-slack_lesser_than** to report only those paths inside or outside a given slack range.

-slack_lesser_than *lesser_slack_limit*

Specifies that only those paths with a slack less than *lesser_slack_limit* are to be reported. This option can be combined with **-slack_greater_than** to report only those paths inside or outside a given slack range.

-lesser_path *max_path_delay*

Selects only those paths with a delay less than *max_path_delay*. Combine this option with the **-greater_path** option to select only those paths inside or outside a given delay range.

-greater_path *min_path_delay*

Selects only those paths with a delay greater than *min_path_delay*. Combine this option with the **-lesser_path** option to select only those paths inside or outside a given delay range.

-loops

Reports only the timing loops in the design.

-true

Reports the longest (least-slack) true paths in the design. This option can require long runtimes for certain designs that have many false paths. Use the **true_delay_prove_true_backtrack_limit** and **true_delay_prove_false_backtrack_limit** variables to limit the amount of backtracking during the operation of **report_timing -true**. Use the **set_true_delay_case_analysis** command to specify a partial input vector to be considered for **-true** analysis. You cannot combine **-true** with the **-max_paths(>1)**, **-nworst(>1)**, **-lesser_path**, **-greater_path**, **-slack_lesser_than**, **-slack_greater_than**, **-loops**, or **-delay** (path type other than max) options. This option requires a DC Ultra license.

-true_threshold *path_delay*

Specifies a threshold path delay value, in library time units, to be used by the **-true** option to speed up the searching. This option is only used the **true** option. If you specify this option, **report_timing -true** returns the first path it finds that is greater than or equal to *path_delay*, rather than continuing to search for a longer one. This option requires a DC Ultra license.

-justify

Finds an input vector that sensitizes the reported paths, or reports that the path is false if no input vector is found. Use the **set_true_delay_case_analysis** command to specify a partial input vector to be considered for **-justify** analysis. You cannot use this option with the **-loops** option. This option requires a DC Ultra license.

-enable_preset_clear_arcs

Enables asynchronous timing arcs for this report. By default, asynchronous timing arcs are disabled during all timing verification. This option allows you to see the timing with these arcs enabled. Only the current report is affected.

-significant_digits *digits*

Specifies the number of digits to the right of the decimal point to report. Allowed values are from 0 through 13. The default is 2. Using this option overrides the value set by the **report_default_significant_digits** variable.

-nosplit

Prevents prevents line-splitting and facilitates writing software to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

-sort_by group | slack

Specifies the order in which the paths are reported. The default **-sort_by** key is **group**. By default, paths are sorted by costing groups. Within each group, the paths are ordered by slack. With **slack**, the paths are ordered by slack only.

-group *group_name*

Specifies the path group from which timing paths are selected for reporting, based on other specified options for reports. If **-group** is not specified, the reported paths are a subset of paths from all path groups. This option cannot be used with **-loops**.

-trace_latch_borrow

This option controls the type of report generated for a path that starts at a transparent latch. If the path startpoint borrows from the previous stage, using this option causes the report to show the entire set of borrowing paths that lead up to the borrowing latch, starting with a nonborrowing path or a noninverting sequential loop. By default, the report shows only the last path in the sequence of borrowing stages. Each stage is reported separately, showing the time borrowed and lent and the endpoints of the stage. The cumulative amount of borrowed time along a sequence of stages is not included in the report. The **-input_pins**, **-nets**, **-transition_time**, **-capacitance**, and **-significant_digits** option apply to every stage in the sequence of borrowing paths, but the remaining options (for example, **-from** and **-true**) apply only to the last stage reported.

-derate

Prints timing derate values for each path element. By default, no derate value is printed. When this option is specified, the **-input_pins** and **-nets** options are automatically turned on. Input delay and ideal clock network latency is not derated.

-scenario *scenario_list*

Option "-scenario all" has been replaced by "-scenario *scenario_list*" to reports timing for given list of scenarios of a multi-scenario design. please replace "all" setting in existing scripts with "[all_scenarios]" to get the same results as before. Inactive scenarios will be skipped in the report. Each scenario is reported separately, with up to **-max_paths** paths reported for each scenario. If this option is not given, only the current scenario is reported.

-temperature

For multivoltage designs, reports the operating condition temperature for each path element.

-voltage

For multivoltage designs, reports the operating condition voltage for each path element.

DESCRIPTION

The **report_timing** command provides a report of timing information for the current design. By default, the **report_timing** command reports the single worst setup path in each clock group.

The command options let you specify the number of paths reported, the types of paths reported, and the amount of detail included in the report. You can restrict the scope of paths reported by startpoints, endpoints, and intermediate points by using the **-from**, **-to**, and **-through** options; and by slack or clock group by using the **-slack_lesser_than** and **-group** options.

The timing report starts by showing the primary command settings, operating conditions, path startpoint, path endpoint, path group name, and path timing check type (max for a setup check, min for a hold check, and so on).

A table in the report shows point-by-point accounting of the delays along the path

from the startpoint to the endpoint. The default table has columns labeled Point, Incr, and Path. These columns list the points (cell pins) along the path, the incremental contribution to the delay at each point, and the cumulative delay up to that point, respectively. Hierarchical boundary crossings are listed as well, showing zero incremental delay at each crossing. You can optionally display net delays in the report by using the **-input_pins** option and net names by using the **-nets** option.

The symbols **r** and **f** in the Path column indicate the sense of the signal transition, either rising or falling, at that point in the path.

For a setup check, the path starts with the launch clock edge and ends at the data input of the capture device. The data arrival time shown in the table is the amount of elapsed time from the source of the launch clock edge to the arrival of data at the endpoint, taking into consideration the longest possible delays along the path.

Following this is the accounting for the required arrival time. The data required time shown in the table is the latest allowable arrival time for the data at the path endpoint, taking into account the nominal capture clock edge time, the clock network delay, the clock uncertainty, the least possible delay along the clock path, and the library setup time requirement for the capture device.

The slack value shown at the end of the report is the data required time minus the data arrival time. This represents the amount of time by which the timing constraint is met.

Back-annotations on path elements in the timing path are indicated by a character symbol in the Incr column. If the **-input_pins** option is used, each pin-to-pin delay spans either a net or cell. The symbol shown refers to the dominant annotation on this path element. (Certain annotations dominate others; for example, SDF takes precedence over back-annotated RC parasitics.) Without **-input_pins**, the delay shown may span both a net and a cell. If they have the same dominant annotation, the appropriate symbol is shown; otherwise, the symbol "H" appears to show that a hybrid of annotation types exists on the corresponding path segment.

Symbol	Annotation
-----	-----
H	Hybrid annotation
^	Ideal network latency annotation
*	SDF back-annotation
&	RC network back-annotation
\$	RC pi back-annotation
+	Lumped RC
<none>	Wire-load model or none

You can use multiple **-through** options in a single command to specify paths that traverse multiple points in the design. The following example specifies paths beginning at A1, passing through B1, then through C1, and ending at D1:

```
prompt> report_timing -from A1 -through B1 -through C1 -to D1
```

If more than one object is specified within one **-through** option, the path can pass through any of the objects. The following example specifies paths beginning at A1,

passing through either B1 or B2, then passing through either C1 or C2, and ending at D1:

```
prompt> report_timing -from A1 -through {B1 B2} -through {C1 C2} -to D1
```

Multicorner-Multimode Support

By default, this command uses information from the current scenario. You can select different scenarios by using the **-scenario** option.

EXAMPLES

The following example shows a timing report using only the default values:

```
prompt> report_timing
```

```
*****
Report : timing
        -path full
        -delay max
        -max_paths 1
Design : led
Version: v3.1a
Date   : Tue Apr  7 16:23:02 1992
*****
```

Operating Conditions:
Wire Loading Model Mode: top

Startpoint: c (input port)
Endpoint: z2 (output port)
Path Group: default
Path Type: max

Point	Incr	Path
input external delay	0.00	0.00 r
c (in)	0.00	0.00 r
u1/Z (IVA)	0.54	0.54 f
u0/Z (NR2)	1.20	1.74 r
u8/Z (IVA)	0.43	2.17 f
u7/Z (OR3)	1.24	3.41 f
z2 (out)	0.00	3.41 f
data arrival time		3.41
max_delay	0.00	0.00
output external delay	0.00	0.00
data required time		0.00
data required time		0.00
data arrival time		-3.41
slack (VIOLATED)		-3.41

The following example reports the longest path to Z1, without required-time and slack calculation:

```
prompt> report_timing -to z1 -nworst 2 -path only
```

```
*****
```

```
Report : timing
```

```
-path only
```

```
-delay max
```

```
-nworst 2
```

```
-max_paths 2
```

```
Design : led
```

```
Version: v3.1a
```

```
Date : Tue Apr 7 16:52:43 1992
```

```
*****
```

```
Operating Conditions:
```

```
Wire Loading Model Mode: top
```

```
Startpoint: c (input port)
```

```
Endpoint: z1 (output port)
```

```
Path Group: default
```

```
Path Type: max
```

Point	Incr	Path
input external delay	0.00	0.00 f
c (in)	0.00	0.00 f
u1/Z (IVA)	0.60	0.60 r
u17/Z (AO7)	0.53	1.13 f
u18/Z (OR3)	1.24	2.37 f
z1 (out)	0.00	2.37 f
data arrival time		2.37

```
Startpoint: d (input port)
```

```
Endpoint: z1 (output port)
```

```
Path Group: default
```

```
Path Type: max
```

Point	Incr	Path
input external delay	0.00	0.00 f
d (in)	0.00	0.00 f
u20/Z (IVA)	0.53	0.53 r
u17/Z (AO7)	0.53	1.06 f
u18/Z (OR3)	1.24	2.30 f
z1 (out)	0.00	2.30 f
data arrival time		2.30

The following example reports the endpoint path delay, required time, and slack for each path:

prompt> **report_timing -path end**

```
*****
Report : timing
        -path end
        -delay max
Design : led
Version: v3.1a
Date   : Tue Apr  7 16:28:07 1992
*****
```

Operating Conditions:
Wire Loading Model Mode: top

Endpoint	Path Delay	Path Required	Slack
z2	3.41 f	0.00	-3.41
z3	3.03 f	0.00	-3.03
z4	2.77 f	0.00	-2.77
z6	2.69 r	0.00	-2.69
z0	2.59 f	0.00	-2.59
z1	2.37 f	0.00	-2.37
z5	2.26 f	0.00	-2.26

The following example reports the start and end points of the path from a to z2:

prompt> **report_timing -from a -to z2 -path short**

```
*****
Report : timing
        -path short
        -delay max
        -max_paths 1
Design : led
Version: v3.1a
Date   : Tue Apr  7 16:29:40 1992
*****
```

Operating Conditions:
Wire Loading Model Mode: top

Startpoint: a (input port)
Endpoint: z2 (output port)
Path Group: default
Path Type: max

Point	Incr	Path
input external delay	0.00	0.00 f
a (in)	0.00	0.00 f
...		
z2 (out)	1.24	1.24 f
data arrival time		1.24

max_delay	0.00	0.00
output external delay	0.00	0.00
data required time		0.00

data required time		0.00
data arrival time		-1.24

slack (VIOLATED)		-1.24

The following example shows input pins in the report, in addition to the default values:

prompt> **report_timing -input_pins**

```
*****
Report : timing
        -path full
        -delay max
        -input_pins
        -max_paths 1
Design : led
Version: v3.1a
Date   : Tue Apr  7 16:32:28 1992
*****
```

Operating Conditions:
Wire Loading Model Mode: top

Startpoint: c (input port)
Endpoint: z2 (output port)
Path Group: default
Path Type: max

Point	Incr	Path

input external delay	0.00	0.00 r
c (in)	0.00	0.00 r
u1/A (IVA)	0.00	0.00 r
u1/Z (IVA)	0.54	0.54 f
u0/A (NR2)	0.00	0.54 f
u0/Z (NR2)	1.20	1.74 r
u8/A (IVA)	0.00	1.74 r
u8/Z (IVA)	0.43	2.17 f
u7/B (OR3)	0.00	2.17 f
u7/Z (OR3)	1.24	3.41 f
z2 (out)	0.00	3.41 f
data arrival time		3.41
max_delay	0.00	0.00
output external delay	0.00	0.00
data required time		0.00

data required time		0.00
data arrival time		-3.41

slack (VIOLATED) -3.41

The following example shows input pins and nets in the report, and does not show required time and slack calculation:

```
prompt> report_timing -input_pins -nets -path only
```

```
*****
Report : timing
        -path only
        -delay max
        -input_pins
        -nets
        -max_paths 1
Design : led
Version: v3.1a
Date   : Tue Apr  7 16:34:20 1992
*****
```

Operating Conditions:
Wire Loading Model Mode: top

Startpoint: c (input port)
Endpoint: z2 (output port)
Path Group: default
Path Type: max

Point	Incr	Path
input external delay	0.00	0.00 r
c (in)	0.00	0.00 r
c (net)	0.00	0.00 r
u1/A (IVA)	0.00	0.00 r
u1/Z (IVA)	0.54	0.54 f
cell24/n22 (net)	0.00	0.54 f
u0/A (NR2)	0.00	0.54 f
u0/Z (NR2)	1.20	1.74 r
cell24/n21 (net)	0.00	1.74 r
u8/A (IVA)	0.00	1.74 r
u8/Z (IVA)	0.43	2.17 f
cell24/n19 (net)	0.00	2.17 f
u7/B (OR3)	0.00	2.17 f
u7/Z (OR3)	1.24	3.41 f
z2 (net)	0.00	3.41 f
z2 (out)	0.00	3.41 f
data arrival time		3.41

The following example reports the longest true paths in the design.

```
prompt> report_timing -true
```

```
*****
Report : timing
        -path full
```

```

        -delay max
        -true
        -max_paths 1
Design : led
Version: v3.1-development
Date   : Fri Jan 21 18:42:18 1994
*****

```

```

Operating Conditions:
Wire Loading Model Mode: top

```

```

Startpoint: c (input port)
Endpoint: z2 (output port)
Path Group: default
Path Type: max

```

Point	Incr	Path

input external delay	0.00	0.00 r
c (in)	0.00	0.00 r
u1/Z (IVA)	0.54	0.54 f
u0/Z (NR2)	1.20	1.74 r
u8/Z (IVA)	0.43	2.17 f
u7/Z (OR3)	1.24	3.41 f
z2 (out)	0.00	3.41 f
data arrival time		3.41
max_delay	0.00	0.00
output external delay	0.00	0.00
data required time		0.00

data required time		0.00
data arrival time		-3.41

slack (VIOLATED)		-3.41

```

True-delay Input Vector

```

d (in)	0
a (in)	0
b (in)	0
c (in)	r

SEE ALSO

```

create_scenario(2)
current_scenario(2)
report_constraint(2)
set_operating_conditions(2)
set_timing_derate(2)
set_timing_ranges(2)
set_true_delay_case_analysis(2)
report_default_significant_digits(3)
timing_report_attributes(3)

```


report_timing_derate

Reports timing derate factors for the design or specified objects.

SYNTAX

```
string report_timing_derate  
[-include_inherited]  
object_list  
[-nosplit]  
[-scenario scenario_list]
```

Data Types

object_list list

ARGUMENTS

-include_inherited

Reports the derates on the object, including those that are inherited. The derates reported are the ones used in the delay calculation. The name of the cell or design from which the derate is inherited is shown in the "Inherited" column. When this option is not specified, only user defined values are reported.

object_list

Specifies the list of objects whose derate factors need to be reported. If an object list is not specified, all objects in the current design with user-defined derates are displayed. The objects in the list can be leaf cells, instances, library cells, and designs.

-nosplit

Prevents line-splitting and facilitates writing software to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

-scenario *scenario_list*

Reports timing derate for given list of scenarios of a multiscenario design. Inactive scenarios will be skipped in the report. Each scenario is reported separately. If this option is not given, only the current scenario is reported.

DESCRIPTION

The **report_timing_derate** command reports derate factors for the design or specified objects. When the **-include_inherited** option is specified, derate factors that are used in delay calculation are reported. The derate factors can be the ones set on the object by user, or inherited from other objects. By default, only user-defined derate factors are reported.

Multicorner-Multimode Support

By default, this command uses information from the current scenario. You can select different scenarios by using the **-scenario** option.

EXAMPLES

The following example reports derates for all of the objects in current design that have user-defined derate factors:

```
prompt> report_timing_derate
```

The next example reports all derates for the *s1* instance.

```
prompt> report_timing_derate s1 -include_inherited
```

SEE ALSO

```
report_timing(2)  
reset_timing_derate(2)  
set_timing_derate(2)
```

report_timing_requirements

Reports timing path requirements (user attributes) and related information.

SYNTAX

```
int report_timing_requirements
[-attributes]
[-ignored]
[-from from_list
  | -rise_from rise_from_list
  | -fall_from fall_from_list]
[-through through_list]
[-rise_through rise_through_list]
[-fall_through fall_through_list]
[-to to_list
  | -rise_to rise_to_list
  | -fall_to fall_to_list]
[-expanded]
[-nosplit]
```

Data Types

<i>from_list</i>	list
<i>rise_from_list</i>	list
<i>fall_from_list</i>	list
<i>through_list</i>	list
<i>rise_through_list</i>	list
<i>fall_through_list</i>	list
<i>to_list</i>	list
<i>rise_to_list</i>	list
<i>fall_to_list</i>	list

ARGUMENTS

-attributes

Lists the path delay attributes set on the design. If used with the **-from** or **-to** option, **-attributes** limits the report to the specified from and to objects. This option reports on attributes set by the **set_multicycle_path**, **set_false_path**, **set_max_delay**, and **set_min_delay** commands. The **-attributes** option is the default when no other options are specified. Any *max_time_borrow* attributes on objects are printed.

-ignored

Lists path timing attributes set on the current design that are ignored. For example, a false path may have been specified from port A to port Z1, but there is no timing path between those points. Use **-from** or **-to** to limit the report to certain paths. Ignored attributes on objects are printed, such as **max_time_borrow** on a cell other than a level-sensitive latch.

-from *from_list*

Specifies a list of clocks, ports, cells, and pins in the current design. The **-from** option limits the report to information that was set with **-from** to a

path-based command, such as **set_multicycle_path**.

-rise_from *rise_from_list*

Same as the **-from** option, except that the path must rise from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by rising edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of **-from**, **-rise_from**, and **-fall_from**.

-fall_from *fall_from_list*

Same as the **-from** option, except that the path must fall from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of **-from**, **-rise_from**, and **-fall_from**.

-through *through_list*

Specifies a list of ports, cells, and pins in the current design. The **-through** option limits the report to information that was set with **-through** to a path-based command, such as **set_multicycle_path**.

-rise_through *rise_through_list*

Same as the **-through** option, but applies only to paths with a rising transition at the specified objects. You can specify **-rise_through** more than once in a single command invocation as with the **-through** option.

-fall_through *fall_through_list*

Same as the **-through** option, but applies only to paths with a falling transition at the specified objects. You can specify **-fall_through** more than once in a single command invocation as with the **-through** option.

-to *to_list*

Specifies a list of clocks, ports, cells, and pins in the current design. The **-to** option limits the report to information that was set with **-to** to a path-based command, such as **set_multicycle_path**.

-rise_to *rise_to_list*

Same as the **-to** option, but applies only to paths rising at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths captured by rising edge of the clock at clock source, taking into account any logical inversions along the clock path. You can use only one of **-to**, **-rise_to**, and **-fall_to**.

-fall_to *fall_to_list*

Same as the **-to** option, but applies only to paths falling at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of **-to**, **-rise_to**, and **-fall_to**.

-expanded

Expands compressed timing exceptions on the design. Compressed timing exceptions applied on the design are reported as compressed exceptions by the **report_timing_requirements** command. Although every object in the from, to,

and through list gets enumerated, the exception is still a compressed exception. With the **-expanded** switch, these exceptions are expanded so that each of the from, to, and through list references at the most one element. It is considered best practice not to use this switch, since it can create a very large report.

-nosplit

Prevents line splitting and facilitates writing software to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

DESCRIPTION

The **report_timing_requirements** command produces a report showing information about timing requirements of the design.

Use the **reset_design** command to remove all timing attributes on the current design. To remove timing attributes from specified paths, use the **reset_path** command.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example lists all timing attributes set on the design:

```
prompt> report_timing_requirements

*****
Report : timing_requirements
        -attributes
Design  : counter
Version: v3.0
Date    : Thu Mar 19 19:28:32 1992
*****
```

From	Through	To	Setup	Hold
RESET	*	*	FALSE	FALSE
*	U1/Z, U5/A	CO	max=4.5	min=2.2
ffa	*	ffb	cycles=2	-

The following example lists all ignored timing attributes set on the design:

```
prompt> report_timing_requirements -ignored

*****
Report : timing_requirements
        -ignored
```

```

Design : counter
Version: v3.0
Date   : Mon Apr 20 19:03:38 1992
*****

```

From	Through	To	Setup	Hold
QA	U1/Z	*	max=10	-

Object	Type	Attributes
CLK	clock	max_time_borrow

SEE ALSO

```

current_design(2)
set_false_path(2)
set_max_delay(2)
set_max_time_borrow(2)
set_min_delay(2)
set_multicycle_path(2)

```

report_tlu_plus_files

Reports the files used for TLUPlus extraction.

SYNTAX

```
integer report_tlu_plus_files  
[-scenario scenario_list]
```

ARGUMENTS

```
-scenario scenario_list  
    Reports tlu plus files for given list of scenarios of a multiscenario design.  
    Inactive scenarios will be skipped in the report. Each scenario is reported  
    separately. If this option is not given, only the current scenario is  
    reported.
```

DESCRIPTION

This command reports the files used for virtual route and post route extraction using TLUPlus.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example runs the **report_tlu_plus_files** command.

```
prompt> report_tlu_plus_files
```

SEE ALSO

```
extract_rc(2)  
set_tlu_plus_files(2)
```

report_track

Reports the routing tracks for a specified layer or for all layers.

SYNTAX

```
int report_track
[-layer layer]
[-dir X | Y]
```

Data Types

layer string

ARGUMENTS

-layer layer
Specifies the routing layer to use the routing tracks. You can use layer name, layer number, or a collection containing one layer object. The default is to report the routing tracks on all layers.

-dir X | Y
Specifies the direction how routing tracks are placed. The valid values are **X** and **Y**; specify either. The default is to report routing tracks in both direction.

DESCRIPTION

This command reports a group of routing tracks for the routing layer.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports routing tracks on the floorplan.

```
prompt> report_track
*****
Report track
Design : design
Version: Y-2006.06-ICC-SP2
Date   : Thu Jul  6 00:34:03 2006
*****
Layer           Direction      Start           Tracks          Pitch           Attr
-----
Attributes :
    usr : User defined
    rt  : Route66 defined

report_track
2058
```


def : DEF defined

m2	Y	233.720	2861	0.560	
m2	X	233.720	2649	0.560	
m3	X	233.720	2649	0.560	
m3	Y	233.720	2861	0.560	
m2	Y	233.720	2861	0.560	
m4	Y	233.720	2861	0.560	
m4	X	233.720	2649	0.560	
m3	X	233.720	2649	0.560	
m5	X	233.720	2649	0.560	
m5	Y	233.720	2861	0.560	
m4	Y	233.720	2861	0.560	
m6	Y	233.720	2861	0.560	
m6	X	232.600	1327	1.120	
m5	X	232.600	1327	1.120	
m1	X	20.000	10	2.100	usr

1

prompt> **report_track -layer m3 -dir horizontal**

Report track

Design : design

Version: Y-2006.06-ICC-SP2

Date : Thu Jul 6 00:36:25 2006

Layer	Direction	Start	Tracks	Pitch	Attr

Attributes :

usr : User defined

rt : Route66 defined

def : DEF defined

m3	Y	233.720	2861	0.560	

1

SEE ALSO

create_track(2)

remove_track(2)

report_transitive_fanin

Reports logic in the transitive fanin of specified sinks.

SYNTAX

```
int report_transitive_fanin  
-to sink_list  
[-nosplit]
```

Data Types

sink_list list

ARGUMENTS

-to *sink_list*

Specifies a list of sink pins, ports, or nets in the design. The transitive fanin of each sink in the *sink_list* is reported. If a net is specified, the effect is the same as listing all driver pins on the net. This argument is required.

-nosplit

Prevents line splitting and facilitates writing software to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

DESCRIPTION

The **report_transitive_fanin** command produces a report showing the transitive fanin of specified sink pins, ports, or nets in the design. A pin is considered to be in the transitive fanin of a sink if there is a timing path through combinational logic from the pin to that sink. The fanin report stops at the clock pins of registers (sequential cells).

Use the **report_timing** command to show path delays in the design. Use the **report_transitive_fanout** command to see the fanout of a pin, port, or net.

If the *current_instance* is set, the report focuses on the fanin within the current instance. The report stops at the boundaries of the current instance, and paths outside the current instance are not reported. The report shows the hierarchical boundary pins on the current instance.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example shows the transitive fanin of two internal pins in the design:

```
prompt> report_transitive_fanin -to {u5/A u5/B}
```

```
*****
Report : transitive_fanin
Design : counter
Version: v3.1
Date   : Thu 1992
*****
```

The fanin network of sink u5/A is as follows:

Driver Pin	Load Pin	Type	Sense
u2/Z	u5/A	(net arc)	same

Load Pin	Driver Pin	Type	Sense
u2/A	u2/Z	IVA	opposite

Driver Pin	Load Pin	Type	Sense
b	u2/A	(net arc)	opposite

The fanin network of sink u5/B is as follows:

Driver Pin	Load Pin	Type	Sense
c	u5/B	(net arc)	same

SEE ALSO

```
current_design(2)
current_instance(2)
report_clock(2)
report_timing(2)
report_transitive_fanout(2)
```

report_transitive_fanout

Reports logic in the transitive fanout of specified sources.

SYNTAX

```
int report_transitive_fanout  
-clock_tree | -from source_list  
[-nosplit]
```

Data Types

source_list list

ARGUMENTS

-clock_tree

Indicates that all clock source pins and/or ports in the design are to be used as the list of sources. Clock sources are specified using the **create_clock** command. If there are no clocks, or if the clocks do not have sources, the report is empty. Use the **report_clock** command to list the sources for all clocks in the design. The **-clock_tree** option is a special form of the report that displays the clock trees (or "networks") in the design. Either the **-clock_tree** or the **-from** option must be specified. The options are mutually exclusive, so specify only one.

-from *source_list*

Specifies a list of source pins, ports, and/or nets in the design. The transitive fanout of each source in the *source_list* is reported. If a net is specified, the effect is the same as listing all load pins on the net. Either the **-clock_tree** or the **-from** option must be specified. The options are mutually exclusive, so specify only one.

-nosplit

Prevents line splitting and facilitates writing software to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

DESCRIPTION

The **report_transitive_fanout** command produces a report showing the transitive fanout of specified source pins or ports in the design. A pin is considered to be in the transitive fanout of a source if there is a timing path through combinational logic from the source to that pin. The fanout report stops at the inputs to registers (sequential cells). The source pins or ports are specified with either **-clock_tree** or **-from *source_list***.

Use the **report_timing** command to show path delays in the design. Use the **report_transitive_fanin** command to see the fanin of a pin, port, or net.

If the *current_instance* is set, the report focuses on the fanout within the current instance. The report stops at the boundaries of the current instance, and does not report paths outside the current instance. The report also shows the hierarchical boundary pins on the current instance.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example lists the transitive fanout for all clock sources in the design:

```
prompt> report_transitive_fanout -clock_tree
```

```
*****
Report : transitive_fanout
        -clock_tree
Design : counter
Version: v3.0
Date   : Thu 1992
*****
```

The fanout network of source CLK is as follows:

Driver Pin	Load Pin	Type	Sense
CLK	ffd/CP	(net arc)	same
CLK	ffc/CP	(net arc)	same
CLK	ffb/CP	(net arc)	same
CLK	ffa/CP	(net arc)	same

The following example shows the transitive fanout of two internal pins in the design:

```
prompt> report_transitive_fanout -from {ffa/Q ffb/Q}
```

```
*****
Report : transitive_fanout
Design : counter
Version: v3.0
Date   : Thu 1992
*****
```

The fanout network of source ffa/Q is as follows:

Driver Pin	Load Pin	Type	Sense
ffa/Q	QA/A	(net arc)	same

Load Pin	Driver Pin	Type	Sense

QA/A	QA/Z	IV	opposite
Driver Pin	Load Pin	Type	Sense
QA/Z	QA	(net arc)	opposite

The fanout network of source ffb/Q is as follows:

Driver Pin	Load Pin	Type	Sense
ffb/Q	QB/A	(net arc)	same
Load Pin	Driver Pin	Type	Sense
QB/A	QB/Z	IV	opposite
Driver Pin	Load Pin	Type	Sense
QB/Z	QB	(net arc)	opposite

SEE ALSO

```
create_clock(2)
current_design(2)
current_instance(2)
report_clock(2)
report_timing(2)
report_transitive_fanin(2)
```

report_units

Reports the units used for resistance, capacitance, timing, leakage power, current, and voltage in the flow. The units must be consistent with the main library units.

SYNTAX

string **report_units**

ARGUMENTS

The **report_units** command has no arguments.

DESCRIPTION

The **report_units** command displays the units used by the current design. To generate the report the design should be loaded and linked to a library. The units are always reported in reference to the MKS units like farad, amp, ohm, second, volt and watt.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows the report_units output.

```
*****
Report : units
Design : labf.mw
Version: Z-2007.03-ICC

Date    : Tue Aug 22 10:32:44 2006
*****
Units
-----
Time_unit      : 1e-09 Second(nS)
Capacitive_load_unit : 1e-12 Farad(pF)
Resistance_unit : 1000 Ohm(kohm)
Voltage_unit    : 1 Volt
Power_unit      : 1e-06 Watt(uW)
Current_unit    : 1 Amp
```

SEE ALSO

set_units(2)

report_voltage_area

Reports the voltage areas in the design.

SYNTAX

```
int report_voltage_area  
-all | patterns
```

Data Types

patterns list

ARGUMENTS

-all

Specifies that all voltage areas in the design must be reported. The voltage areas are created by using the **create_voltage_area** command for the hierarchical blocks.

patterns

Specifies the voltage areas to be reported. The patterns can be a collection handle of voltages or names of patterns. You can use the *get_voltage_areas* command to specify voltage areas to be reported.

DESCRIPTION

This command generates reports the user-specified voltage area constraints in the design, as specified by the **create_voltage_area** command. If you use the **-all** option, all voltage areas in the design, created by the **create_voltage_area** command are reported.

The report generated includes voltage area name, hierarchical blocks associated with voltage area, voltage area geometry, area of the voltage area, and utilization for the voltage area. The area of the voltage area is specified in microns.

Voltage areas can physically be completely nested: one voltage area lies completely inside another voltage area. When considering the area or utilization of the outside voltage area, the area of the inner voltage area is excluded.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports the voltage area *va1*.

```
prompt> report_voltage_area va1  
*****
```

```
report_voltage_area  
2066
```



```
Voltage area name val
Block(s) :PPL_LBUS_CORE
Voltage area geometry : {19.87 29.86 576.47 1148.89}
Voltage area region Area : 622854.88
Voltage area Utilization : 0.66
Voltage Area Guard-band (X, Y) : (1, 1)
*****
1
```

SEE ALSO

```
create_voltage_area(2)
get_voltage_areas(2)
update_voltage_area(2)
remove_voltage_area(2)
```

report_vt_filler_rule

Writes out multiple threshold voltage filler cell insertion rules.

SYNTAX

```
status report_vt_filler_rule  
-threshold_voltage vt_type_1 vt_type_2
```

ARGUMENTS

```
-threshold_voltage vt_type_1 vt_type_2  
    Specifies threshold voltage types of the rule to be writes out
```

DESCRIPTION

Use this command to report multiple threshold voltage filler rules of the design.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example will print out rules for threshold voltage "vtType1" and "vtType2"

```
prompt> report_vt_filler_rule -threshold_voltage {vtType1 vtType2}
```

The following example will print out all threshold voltage rules for current design

```
prompt> report_vt_filler_rule
```

Below is an exmaple of the output. The current design library is "design", with 2 reference libraries MVT_LIB_2 and LVT_LIB_2. No vt filler rule defined for cells in "design". one cell L1SUBCON1 in library MVT_LIB_2 is defined as a filler between threshold voltage vtType1 and vtType1. Two cells L2SUBCON1 and L2SUBCON2 in library LVT_LIB_2 are defined as fillers between vtType2 and vtType2. ;; VT fillers in library MVT_LIB_2 L1SUBCON1: vtType1vtType1Filler

```
;; VT fillers in library LVT_LIB_2 L2SUBCON1: vtType2vtType2Filler L2SUBCON2:  
vtType2vtType2Filler
```

```
;; VT fillers in library design
```

SEE ALSO

```
insert_stdcell_filler(2)  
set_cell_vt_type(2)  
set_vt_filler_rule(2)  
remove_vt_filler_rule(2)
```

report_write_stream_options

Reports stream out options for the **write_stream** command.

SYNTAX

```
status report_write_stream_options  
[-default]
```

ARGUMENTS

-default
Report the default value of write_stream options. If this option is not specified, current option values are reported.

DESCRIPTION

Command **report_write_stream_options** is used to examine the options which are set by **set_write_stream_options** command. If an option is not specified, the default value will be reported. Both **set_write_stream_options** and **report_write_stream_options** serve **write_stream**.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports the options for write_stream.

```
prompt> set_write_stream_options  
-child_depth 1  
-output_filling {gap}  
prompt> report_write_stream_options
```

SEE ALSO

```
set_write_stream_options(2)  
write_stream(2)
```

report_xtalk_route_options

Reports the settings of the global route and track assignment crosstalk options.

SYNTAX

```
status report_xtalk_route_options  
[-default]
```

ARGUMENTS

```
-default  
    Reports default options
```

DESCRIPTION

This command reports the settings of global route and trackAssignment xtalk options that have been set by the **set_xtalk_route_options** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows the use of the command.

```
prompt> report_xtalk_route_options
```

SEE ALSO

```
set_xtalk_route_options(2)
```

report_zrt_net_properties

Reports the Zroute property settings for a net.

SYNTAX

```
status report_zrt_net_properties  
-net net_name
```

Data Types

net_name collection of one item

ARGUMENTS

```
-net net_name  
    Specifies the net for which to get the Zroute property settings. You can  
    specify only a single net.  
    This is a required option.
```

DESCRIPTION

The **report_zrt_net_properties** command returns the values of all Zroute properties for the specified net.

EXAMPLES

The following example reports the Zroute property settings for the n1 net:

```
prompt> report_zrt_net_properties -net "n1"
```

SEE ALSO

```
set_zrt_net_properties(2)  
get_zrt_net_properties(2)
```

reset_clock_tree_optimization_options

Resets options used by clock tree optimization.

SYNTAX

```
status reset_clock_tree_optimization_options  
[-clock_trees clock_name_or_collection]  
[-enable_multicorner_optimization]  
[-gate_sizing]  
[-gate_relocation]  
[-preserve_levels]  
[-area_recovery]  
[-all]  
[-balance_rc]  
[-relax_insertion_delay]  
[-corner_target_skew]
```

ARGUMENTS

-clock_trees *clock_name_or_collection*

Specifies a set of clocks to be processed. Only clock names defined through `create_clock` or `create_generated_clock` are accepted. If there is a list of clock names, the options then apply to those clocks only. By default, the options apply for all the clocks.

-enable_multicorner_optimization

Reset the type of Multi-corner optimization to default. Multi-corner optimization default is none. This option can not be specified together with `"-clock_trees clock_name_or_collection"`. In addition to enabling multi-corner optimization option, user should also set min libraries using `set_min_library`; then we do 2-corner optimization. When `real_corners` is given, tool will perform optimization based on libraries provided. In the case of `virtual_corners`, tool will create multiple corners and do optimization.

-gate_sizing

Reset gate sizing to default. By default, `gate_sizing` is **true**. The gates in question are the user instantiated gates and buffers only and not introduced cells during `compile_clock_tree` or `optimize_clock_tree`; `Compile_clock_tree` or `optimize_clock_tree` introduced cells can always be sized, with the exception of boundary cells. User gates, for example, are those that are introduced during prior to `compile_clock_tree`, or user may have put them for some purpose.

-gate_relocation

Reset gate relocation to default. By default, `gate_relocation` is **true**. The gates in question are the user instantiated gates and buffers only and not introduced cells during `compile_clock_tree` or `optimize_clock_tree`; `Compile_clock_tree` or `optimize_clock_tree` introduced cells can always be relocated. User gates, for example, are those that are introduced prior to `compile_clock_tree`, or user may have put them for some purpose.

-preserve_levels

Reset preserve_levels to default. By default, preserve_levels is **false**. If set to true, then the number of levels will be preserved. This option is combining delay insertion and delay removal options. The target_early_delay, if specified with set_clock_tree_options, will override preserve_levels only at the root of the clock.

-area_recovery

Reset area_recovery to default. By default, area_recovery is **true**.

-all

Reset all of above options to default.

DESCRIPTION

This command is the UI resetting options for clock tree optimization in ICC. Before command optimize_clock_tree or clock_opt, user can use this command to reset options to specific clocks if -clock_trees is given, or reset options to the design which will be applied to all clocks that do not have the per clock option.

By disabling multi-corner optimization, the tool will not perform multi-corner optimization. User can use this command to enable sizing or relocation to cells that were introduced prior to compile_clock_tree. Area recovery is a tool to reduce clock buffer area, significantly in some cases. User can turn it on through this command.

This command only resets options to default for optimize_clock_tree. In clock_opt, compile_clock_tree and optimize_clock_tree are called. This reset_clock_tree_optimization_options only affect optimize_clock_tree part of clock_opt.

Options -gate_sizing, -gate_relocation are both under set_clock_tree_options and set_clock_tree_optimization_options. If they are under set_clock_tree_optimization_options, they control optimize_clock_tree; if under set_clock_tree_options, they control optimization during compile_clock_tree. Under set_clock_tree_options, there are -buffer_sizing/-buffer_relocation/-delay_insertion. They also only control compile_clock_tree.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example disables gate_sizing to clock PCI_CLK. Then reset gate_sizing to default (enable) for clock PCI_CLK.

```
prompt> set_clock_tree_optimization_options -clock_trees [get_clock PCI_CLK] \  
-gate_sizing false  
1  
prompt> reset_clock_tree_optimization_options -clock_trees  
[get_clock PCI_CLK] -gate_sizing
```

1

This example sets the multicorner optimization type to real_corners to all clocks. Then reset multicorner optimization to default (none) for all clocks.

```
prompt> set_clock_tree_optimization_options -  
enable_multicorner_optimization real_corners  
1  
prompt> reset_clock_tree_optimization_options -enable_multicorner_optimization  
1
```

SEE ALSO

```
optimize_clock_tree(2)  
clock_opt(2)  
compile_clock_tree(2)  
set_clock_tree_options(2)  
set_clock_tree_optimization_options(2)  
report_clock_tree_optimization_options(2)
```


reset_clock_tree_options

Resets clock tree options which are set by user using `set_clock_tree_options` command to their default values.

SYNTAX

```
void reset_clock_tree_options  
[-clock_trees pin_col_spec]  
[-root pin_col_spec]  
[-buffer_relocation]  
[-buffer_sizing]  
[-delay_insertion]  
[-gate_relocation]  
[-gate_sizing]  
[-insert_boundary_cell]  
[-layer_list]  
[-max_buffer_levels]  
[-max_capacitance]  
[-max_fanout]  
[-max_transition]  
[-max_rc_delay_constraint]  
[-max_rc_scale_factor]  
[-operating_condition]  
[-routing_rule]  
[-target_early_delay]  
[-target_skew]  
[-use_default_routing_for_sinks]  
[-top_mode]  
[-logic_level_balance]  
[-config_file_read]  
[-config_file_write]  
[-all]  
[-global]  
[-ocv_clustering]
```

Data Types

pin_col_spec list

ARGUMENTS

- `-clock_trees pin_col_spec`
Resets the specified options for only those clock trees listed in `clock_name_or_collection`. Each element of `clock_name_or_collection` is the name of the source (a top-level design input port, a hierarchical port, or a library cell instance pin) of each clock tree. If the `reset_clock_tree_options` command is used without the `-clock_trees` option, the command is applied to all currently-defined clock trees in the design.
- `-buffer_relocation`
Resets the buffer relocation option set by user in `set_clock_tree_options` command to its default value of "true". This option may either be reset on

specific clock trees, by using the `clock_trees` clock option, or on all clocks if `clock_trees` is not specified

-buffer_sizing

Resets the buffer sizing option set by user in `set_clock_tree_options` command to its default value of "true". This option may either be reset on specific clock trees, by using the `clock_trees` option, or on all clocks if `clock_trees` is not specified.

-gate_relocation

Resets the gate relocation option set by user in `set_clock_tree_options` command to its default value of "true". This option may either be reset on specific clock trees, by using the `clock_trees` option, or on all clocks if `clock_trees` is not specified.

-gate_sizing

Resets the gate sizing option set by user in `set_clock_tree_options` command to its default value of "false". This option may either be reset on specific clock trees, by using the `clock_trees` option, or on all clocks if `clock_trees` is not specified.

-insert_boundary_cell

Resets the `insert_boundary_cell` option set by user in `set_clock_tree_options` command to its default value of "false". This is a global option and will affect all clock trees.

-layer_list

Resets the list of layers specified by user in `set_clock_tree_options` command to its default value of all available layers. The list of layers will be used during routing of clock nets. This option may either be reset on specific clock trees, by using the `clock_trees` option, or on all clocks if `clock_trees` is not specified.

-max_buffer_levels

Resets the maximum number of levels specified by user in `set_clock_tree_options` command to its default value of "50". This option may either be reset on specific clock trees, by using the `clock_trees` option, or on all clocks if `clock_trees` is not specified.

-max_capacitance

Resets the maximum capacitance design rule checking (DRC) constraint set by user in `set_clock_tree_options` command to its default value of "0.6" (pf). This option may either be reset on specific clock trees, by using the `clock_trees` option, or on all clocks if `clock_trees` is not specified.

-max_fanout

Resets the maximum fanout design rule checking (DRC) constraint set by user in `set_clock_tree_options` command to its default value of "2000". This option may either be reset on specific clock trees, by using the `clock_trees` option, or on all clocks if `clock_trees` is not specified.

-max_transition

Resets the maximum transition design rule checking (DRC) constraint set by user in `set_clock_tree_options` command to its default value of "0.5" (ns). This option may either be reset on specific clock trees, by using the

clock_trees option, or on all clocks if clock_trees is not specified.

-max_rc_delay_constraint

Removes and unsets the maximum RC delay constraint set by user in set_clock_tree_options command. This option may either be reset on specific clock trees, by using the clock_trees option, or on all clocks if clock_trees is not specified.

-max_rc_scale_factor

Removes and unsets the scale factor associated with the maximum RC delay constraint set by user in set_clock_tree_options command. This option may either be reset on specific clock trees, by using the clock_trees option, or on all clocks if clock_trees is not specified.

-operating_condition

Resets the operating_condition option set by user in set_clock_tree_options command to its default value of "max". This is a global option and will affect all clock trees.

-routing_rule

Removes and unsets the nondefault routing rule specified by user in set_clock_tree_options command. Unsetting this option will cause the default routing rule to be used for routing the specified clock tree(s). This option may either be removed on specific clock trees, by using the clock_trees option, or on all clocks if clock_trees is not specified.

-target_early_delay

Resets the minimum insertion delay constraint specified by user in set_clock_tree_options command to its default value of ,Áú0,Àù. This option may either be reset on specific clock trees, by using the clock_trees option, or on all clocks if clock_trees is not specified.

-target_skew

Resets the maximum skew soft constraint specified by user in set_clock_tree_options command to its default value of "0". This option may either be reset on specific clock trees, by using the clock_trees option, or on all clocks if clock_trees is not specified.

-use_default_routing_for_sinks

Removes N, which is set by user in set_clock_tree_options command, to its default value of "0", which will cause the default routing rule to be used on the nets driving sinks at the bottom N levels of the clock tree. This is a global option and will affect all clock trees.

-logic_level_balance

Resets the balanced logic level option specified by user in set_clock_tree_options command to its default value of ,Áúfalse,Àù. This is a global option and will affect all clock trees.

-config_file_read

Removes the specified configuration file name set by user in set_clock_tree_options command to its default value of ,ÁúNone,Àù. This is a global option and will affect all clock trees.

-config_file_write
Removes the specified configuration dump file name set by user in set_clock_tree_options command and resets it to the default value of ,ÄÜNone,ÄÜ. This is a global option and will affect all clock trees.

-all
When specified without the clock_trees option, resets, for all clocks, all global and clock-specific options set by user in the set_clock_tree_options command, to their default values. When specified with the clock_trees option, only clock-specific options for the specified clock(s) will be removed, not global options.

-global
Resets all global options set by user in set_clock_tree_options command, to their default values. Note that clock cannot be used with global since these are options that apply to all clocks. -global is equivalent to "-use_default_routing_for_sinks -logic_level_balance -config_file_read -config_file_write ocv_clustering -insert_boundary_cell -operating_condition

-ocv_clustering
Resets the OCV(On Chip Variation)-aware register clustering option specified by user in set_clock_tree_options command to its default value of ,ÄÜfalse,ÄÜ. This is a global option and will affect all clock trees.

DESCRIPTION

This command resets clock tree options that have been previously set using set_clock_tree_options to their default values. If no other options except for -clock_trees are used, no options will be reset.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

This example sets some global and clock-specific clock tree options and then removes only the clock-specific options.

```
prompt> set_clock_tree_options -ocv_clustering true \
prompt> set_clock_tree_options -clock_trees clk -max_transition 0.45 \
prompt> reset_clock_tree_options -all clock_trees clk \
```

At this point, the max_transition on the clock clk will be reset to its default value of 0.5. OCV clustering will still be set since it is a global option.

This example sets some global and clock-specific clock tree options and then removes all options on all clock trees.

```
prompt> set_clock_tree_options -ocv_clustering true \
prompt> set_clock_tree_options -clock_trees clk -max_transition 0.45 \
prompt> set_clock_tree_options -clock_trees tck -target_skew 0.200 -
```

```
max_transition 0.45 \  
prompt> reset_clock_tree_options -all \
```

At this point, all options that were set (both global and clock-specific) will be reset to their default values for all clocks.

This example sets some clock-specific options on a clock and then resets them.

```
prompt> set_clock_tree_options -clock_trees CLKA  
        -buffer_relocation false -gate_relocation false \  
prompt> reset_clock_tree_options -clock_trees CLKA  
        -buffer_relocation -gate_relocation \
```

At this point, the `buffer_relocation` and `gate_relocation` options will be reset to their default value of `true`.

SEE ALSO

`set_clock_tree_options(2)`

reset_clock_tree_references

Removes previously defined references from a clock tree.

SYNTAX

```
status reset_clock_tree_references  
[-clock_trees clock_name_or_collection]  
[-references references]  
[-sizing_only]  
[-delay_insertion_only]  
[-boundary_cell_only]  
[-global]
```

Data Types

<i>clock_name_or_collection</i>	list
<i>references</i>	list

ARGUMENTS

-clock_trees *clock_name_or_collection*
Removes references for the specified list of clocks.

-references *references*
Removes the specified reference gates. If the **-clock_trees** option is used, clock specific reference lists are updated (Lists created by the **set_clock_tree_references -clock_trees** command.) If you do not use the **-clock_trees** option, global reference lists are updated (Lists created without the **-clock_trees** option). If the **-references** option is not used, all reference lists are deleted.

-sizing_only
Removes reference cells that are specified by **-sizing_only**.

-delay_insertion_only
Removes reference cells that are specified by **-delay_insertion_only**.

-boundary_cell_only
Removes reference cells that are specified by **-boundary_cell_only**.

-global
Remove all references from global lists.

DESCRIPTION

Use this command to remove the specified references from clock trees. If the **-clock_trees** option is used, reference lists of the specified clocks are updated. If the **-clock_trees** option is not used, global reference lists are updated.

The **-references** option can be used to remove specific reference cells from the reference lists. The **-sizing_only**, **-boundary_cell_only**, and **-delay_insertion_only**

options can be used with the **-references** option to update respective reference lists. For example, the **-sizing_only** option can be used to update lists created by **set_clock_tree_references -references -sizing_only**.

If **-references** option is not specified, all reference cells are removed. The **-global** option can be used to remove all references from global lists. Clock gates can also be specified as reference cells.

EXAMPLES

The following command removes the INVX4 cell for the CLK1 clock path.

```
prompt> reset_clock_tree_references -clock_trees CLK1 -references INVX4
```

The following command removes the DELAYX4 cell that are specified by the **-delay_insertion_only** option.

```
prompt> reset_clock_tree_references -clock_trees CLK1 \  
-references DELAYX4 -delay_insertion_only
```

The following command removes all references specified for clock CLK1.

```
prompt> reset_clock_tree_references -clock_trees CLK1
```

The following command removes all global references specified for the design.

```
prompt> reset_clock_tree_references -global
```

The following command removes reference cell INVX4 from global lists.

```
prompt> reset_clock_tree_references -references INVX4
```

The following command removes all references, both global and clock-specific references.

```
prompt> reset_clock_tree_references
```

The following command removes clock gates that are specified as references.

```
prompt> reset_clock_tree_references -references AND2X8 \  
-sizing_only -clock_trees CLK1  
prompt> reset_clock_tree_references -references AND2X16 -sizing_only
```

SEE ALSO

`set_clock_tree_references(2)`

reset_cts_batch_mode

Disables clock tree synthesis batch mode.

SYNTAX

status **reset_cts_batch_mode**

ARGUMENTS

None

DESCRIPTION

To reduce runtime in script that call CTS commands many times, in some cases, one clock at a time, the set command turns on the batch mode. It skips CTS pre and post process whenever it can. For example, if 3 compile_clock_tree commands are called, the first call will run the CTS pre-process, but skipped the post process. The second and the third calls will skip both pre and post process. The following command, like reset_cts_batch_mode, save_mw_cel or other trigger commands will make up the CTS post process.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

```
set_cts_batch_mode
```

```
compile_clock_tree -clock_trees clk1
```

```
compile_clock_tree -clock_trees clk2
```

```
report_cts_batch_mode  
optimize_clock_tree  
balance_inter_clkok_delay  
reset_cts_batch_mode
```

SEE ALSO

```
set_cts_batch_mode(2)  
report_cts_batch_mode(2)
```

reset_design

Removes from the current design all user-specified objects and attributes, except those defined using `set_attribute`.

SYNTAX

status **reset_design**

ARGUMENTS

The **reset_design** command has no arguments.

DESCRIPTION

The **reset_design** command removes from the current design all user-specified clocks, path groups, and attributes, except those defined using the **set_attribute** command. This command returns zero if there is no current design.

WARNING: Executing the **reset_design** has extremely wide-ranging consequences. This command removes all user-specified attributes on the design, with a result that is often equivalent to starting the design process from the beginning. If you want to reset only a few attributes, consider using the **remove_attribute** command. Alternatively, you may be able to remove selected attributes using the commands that set them. Many attribute-setting commands, such as **set_jtag_port**, have **-default** options that remove from the current design all attributes previously set by that command. Refer to the appropriate man page to determine whether a specific **set_*** command has the **-default** option.

An often unexpected side effect is that **reset_design** can change the values of some `dc_shell` variables. For example, if a `dc_shell` variable contains one or more instance-specific objects (that is, nets, cells or pins below the top level of the design), then when **reset_design** deletes the objects, they are also removed from the `dc_shell` variable. Thus a variable that contains instance-specific objects no longer contains them after **reset_design** is executed.

For more information on this side effect, and a suggested workaround, refer to the **EXAMPLES** section.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example resets the current design:

```
prompt> reset_design
Resetting current design 'test'
```

The following example illustrates how the **reset_design** command can change the values of dc_shell variables:

```
prompt> a = [get_cells I_2/U70*]
{"I_2/U709", "I_2/U708", "I_2/U707", "I_2/U706"}

prompt> reset_design
Resetting current design 'TOP'
1

prompt> list a
a = {}
1
```

The following example shows a workaround to the side effect that the **reset_design** command may change the values of dc_shell variables. The **string** variables are not affected by **reset_design**, so to preserve the *object_list* variable, you can save it as a string variable before executing **reset_design**. Then, after executing **reset_design**, you can restore the contents of the original *object_list* variable using the appropriate **get_*** command and the string variable.

```
prompt> list_var = [get_cells I_2/U70*]
{"I_2/U709", "I_2/U708", "I_2/U707", "I_2/U706"}
(NOTE: Variable list_var is a list of design objects.)

prompt> string_var = ""
Warning: Defining new variable 'string_var'. (EQN-10)
""
(NOTE: Assigning an empty string to a new variable
assures that it is a string variable.)

prompt> string_var = list_var
{"I_2/U709", "I_2/U708", "I_2/U707", "I_2/U706"}
(NOTE: string_var now contains the names of all the
design objects in list_var.)

prompt> reset_design
Resetting current design 'TOP'
1

prompt> list list_var
list_var = {}
1

prompt> list string_var
string_var = {"I_2/U709", "I_2/U708", "I_2/U707", "I_2/U706"}
1
(NOTE: string_var has not been changed by the
reset_design command.)

prompt> list_var = string_var
{"I_2/U709", "I_2/U708", "I_2/U707", "I_2/U706"}
(NOTE: list_var now contains a list of strings,
```

each of which is the name of one of the design objects previously stored in `list_var`.)

```
prompt> list_var = [get_cells list_var]  
{ "I_2/U709", "I_2/U708", "I_2/U707", "I_2/U706" }  
(NOTE: list_var is now restored to the value it had before  
the execution of reset_design; that is, a list  
of design objects.)
```

SEE ALSO

```
current_design(2)  
remove_attribute(2)  
remove_clock(2)  
reset_path(2)
```

reset_fp_clock_plan_options

Removes clock planning options which you set by using the **set_fp_clock_plan_options** command.

SYNTAX

```
status reset_fp_clock_plan_options
[-output_directory]
[-no_feeds_plan_group]
[-clock_nets]
[-anchor_cell]
[-route_mode]
[-keep_block_tree]
[-update_clock_latency]
[-all]
```

ARGUMENTS

- output_directory
Reset the directory specified to write log files. Restore the output directory to default value "cp_output".
- no_feeds_plan_group
Removes the list of plan groups in which feedthroughs are not allowed. By default, feedthroughs are allowed in all plan groups.
- clock_nets
Removes the list of clock nets on which clock planning will be performed. By default, all clock nets will be processed in clock planning.
- anchor_cell
Removes the list of lib cells to be used as anchor cell during clock planning. However, you must specify at least one lib cell to perform clock planning.
- route_mode
Resets the route mode to the default value "none", with which clock planning performs no interface nets routing.
- keep_block_tree
Resets the switch value to the default "false", so that the block level clock tree will be removed when clock planning finishes.
- update_clock_latency
Resets the switch value to default "false", with which clock planning does not update clock latency before finishing.
- all
Removes all clock planning options set in **set_fp_clock_plan_options**. -all is equivalent to the combination of all the other options.

DESCRIPTION

This command removes/resets the database entry for the clock planning options define in **set_fp_clock_plan_options** command.

Multicorner-Multimode Support

This command is not supported in a multi-scenario design flow.

EXAMPLE

The following example resets all clock planning options.

```
prompt> reset_fp_clock_plan_options -all
```

SEE ALSO

```
set_fp_clock_plan_options(2)  
report_fp_clock_plan_options(2)  
compile_fp_clock_plan(2)
```

reset_latency_adjustment_options

Resets the options for I/O latency adjustment, defined by using the **set_latency_adjustment_options** command.

SYNTAX

```
status reset_latency_adjustment_options  
[-latency]  
[-exclude_clock]  
[-from_clock]
```

DESCRIPTION

This command resets the I/O latency adjustment options previously set by using the **set_latency_adjustment_options** command.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example resets the I/O latency adjustment options.

```
prompt> reset_latency_adjustment_options
```

SEE ALSO

set_latency_adjustment_options(2)

reset_mode

Resets the modes of the specified instances.

SYNTAX

```
int reset_mode  
[instance_list]
```

Data Types

instance_list list

ARGUMENTS

instance_list
Specifies a list of instances for which modes are to be reset. If using DCL, all modes except for the default are disabled. If using .lib, all modes of these instances are enabled. No error occurs if you specify to reset the modes on a cell that has no modes.

DESCRIPTION

The **reset_mode** command resets the modes of the specified instances. If no instances are specified, all instances have their modes reset. If using DCL, all modes except for the default are disabled. If using .lib, all modes of these instances are enabled. This is the default behavior when no modes are specified with command **set_mode**.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example resets the modes of all instances whose name matches the description RAM*. You can use the command **report_mode** to report the active mode of each instance that have modes. Before executing **reset_mode**, instance Uram1/core was only in read mode, and instance Uram2/core was only in write mode. After executing **reset_mode**, the read and write modes are active for both instances, as shown by the **report_mode** command.

```
prompt> report_mode Uram*
```

```
*****  
Report : mode  
Design : top_design  
*****
```

Cell	Mode (Group)	Status

```
reset_mode  
2090
```


Uram1/core(RAM2_core)	read(rw)	ENABLED
	write(rw)	disabled
Uram2/core(RAM2_core)	read(rw)	disabled
	write(rw)	ENABLED

```
prompt> reset_mode RAM*
prompt> report_mode RAM*
```

```
*****
Report : mode
Design : top_design
*****
```

Cell	Mode (Group)	Status
Uram1/core(RAM2_core)	read(rw)	ENABLED
	write(rw)	ENABLED
Uram2/core(RAM2_core)	read(rw)	ENABLED
	write(rw)	ENABLED

SEE ALSO

```
report_mode(2)
set_mode(2)
```

reset_path

Resets specified paths to single cycle timing.

SYNTAX

```
int reset_path
[-setup | -hold]
[-rise | -fall]
[-from from_list
  | -rise_from rise_from_list
  | -fall_from fall_from_list]
[-through through_list]
[-rise_through rise_through_list]
[-fall_through fall_through_list]
[-to to_list
  | -rise_to rise_to_list
  | -fall_to fall_to_list]
```

Data Types

<i>from_list</i>	list
<i>rise_from_list</i>	list
<i>fall_from_list</i>	list
<i>through_list</i>	list
<i>rise_through_list</i>	list
<i>fall_through_list</i>	list
<i>to_list</i>	list
<i>rise_to_list</i>	list
<i>fall_to_list</i>	list

ARGUMENTS

- setup**
Specifies that the setup checking (maximum delay) is reset to single cycle behavior.
- hold**
Specifies that the hold checking (minimum delay) is reset to single cycle behavior.
- rise**
Specifies that the rising path delays are reset to single cycle behavior. The default is that both rising and falling delays are affected.
- fall**
Specifies that falling path delays are set to single cycle behavior. The default is that both rising and falling delays are affected.
- from *from_list***
Specifies names of clocks, pins, or cells to use to find path startpoints. If a specified object is a clock, all flip-flops, latches, and primary inputs related to that clock are used as path startpoints.

-rise_from *rise_from_list*

Same as the **-from** option, except that the path must rise from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by rising edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of **-from**, **-rise_from**, and **-fall_from**.

-fall_from *fall_from_list*

Same as the **-from** option, except that the path must fall from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of **-from**, **-rise_from**, and **-fall_from**.

-through *through_list*

A list of path throughpoints (port, pin, or leaf cell names) of the current design. The **reset_path** applies only to paths that pass through one of the points in the *through_list*. If more than one object is included, the objects must be enclosed either in quotes or in '{}' braces. If you specify the **-through** option multiple times, the **reset_path** applies to paths that pass through a member of each *through_list* in the order the lists were given. In other words, the path must first pass through a member of the first *through_list*, then through a member of the second list, and so on for every through list specified. If you use the **-through** option in combination with the **-from** or **-to** options, the **reset_path** applies only if the **-from** or **-to** conditions are satisfied and the **-through** conditions are satisfied.

-rise_through *rise_through_list*

Same as the **-through** option, but applies only to paths with a rising transition at the specified objects. You can specify **-rise_through** more than once in a single command invocation as with the **-through** option.

-fall_through *fall_through_list*

Same as the **-through** option, but applies only to paths with a falling transition at the specified objects. You can specify **-fall_through** more than once in a single command invocation as with the **-through** option.

-to *to_list*

Specifies names of clocks, pins, or cells to use to find path endpoints. If a specified object is a clock, all flip-flops, latches, and primary outputs related to that clock are used as path endpoints.

-rise_to *rise_to_list*

Same as the **-to** option, but applies only to paths rising at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths captured by rising edge of the clock at clock source, taking into account any logical inversions along the clock path. You can use only one of **-to**, **-rise_to**, and **-fall_to**.

-fall_to *fall_to_list*

Same as the **-to** option, but applies only to paths falling at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock

path. You can use only one of **-to**, **-rise_to**, and **-fall_to**.

DESCRIPTION

Specifies that designated timing paths in the current design are restored to the default single cycle behavior. **reset_path** is used to undo the effect of **set_multicycle_path**, **set_false_path**, **set_max_delay**, and **set_min_delay**. Remove the attributes set by those commands with **reset_path**. Note that a general **reset_path** command removes the effect of matching general or specific point-to-point exception commands.

If you don't specify **-setup** or **-hold**, both are restored to the default behavior. The default is a setup relation of one cycle and a hold relation of zero cycles. A hold value of zero means that hold is checked one active edge before the setup data at the destination register.

If you specify **-setup**, only setup (maximum delay) checking is reset to the default behavior. If you specify **-hold**, only hold (minimum delay) checking is reset to the default behavior.

There is separate setup and hold information for rise and fall. In most cases, these are reset together. The **-rise** or **-fall** options are used to reset only one or the other.

To disable setup or hold calculations for paths, use **set_false_path**.

To see the nondefault path requirements on the design, use **report_timing_requirements**.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example resets the timing relation between ff1/CP and ff2/D to the default single cycle.

```
prompt> reset_path -from {ff1/CP} -to {ff2/D}
```

This example resets the rising delay to single cycle for all paths ending at latch2d.

```
prompt> reset_path -rise -to {latch2d}
```

This example shows how a specific and general point-to-point exception is set, then removed.

```
prompt> set_false_path 2 -from A -to Z
prompt> set_max_delay 15 -to Z
prompt> reset_path -to Z /* removes all exceptions to Z */
```

SEE ALSO

`current_design(2)`
`report_constraint(2)`
`report_timing_requirements(2)`
`reset_design(2)`
`set_false_path(2)`
`set_max_delay(2)`
`set_min_delay(2)`
`set_multicycle_path(2)`

reset_split_clock_gates_options

Specifies to reset the options of **split_clock_gates** to the default settings. The set option command is **set_split_clock_gates_options**.

SYNTAX

status **reset_split_clock_gates_options**

ARGUMENTS

The **reset_split_clock_gates_options** command has no arguments.

DESCRIPTION

This command resets the options of **split_clock_gates** to the default settings. The default values are:

```
f  -honor_dont_touch false
f  -honor_size_only false
f  -slack_margin 0.0
```

SEE ALSO

```
split_clock_gates(2)
set_split_clock_gates_options(2)
report_split_clock_gates_options(2)
```

reset_switching_activity

Removes the toggle rate and static probability attributes, or the maximum toggle rate attribute, from nets, pins, cells, and ports of the current design.

SYNTAX

```
integer reset_switching_activity  
-switching_activity | -max_toggle_rate | -all  
[-verbose]  
[object_list]
```

Data Types

object_list list

ARGUMENTS

-switching_activity | -max_toggle_rate | -all
Specifies the attributes to remove from annotated design objects. The **-switching_activity** option (the default) removes only the toggle rate and static probability attributes. The **-max_toggle_rate** option removes only the maximum toggle rate attribute. Specifying **-all** removes all 3 attributes.

-verbose
Prints more information messages than are printed by default.

object_list
Specifies the objects from which to remove the specified attributes. By default, the attributes are removed from all annotated nets, pins, cells, and ports in the current design.

DESCRIPTION

The command enables you to remove the toggle rate, static probability, and maximum toggle rate attributes throughout the entire design. Executing the **reset_switching_activity** command without any arguments removes the toggle rate and static probability attributes.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example removes all toggle rate and static probability attributes from the *TOP_MULT* design:

```
prompt> current_design TOP_MULT
```

```
prompt> reset_switching_activity
```

The following example removes all maximum toggle rate attributes from the *TOP_MULT* design:

```
prompt> current_design TOP_MULT  
prompt> reset_switching_activity -max_toggle_rate -verbose
```

The following example removes all 3 switching activity attributes from the entire design:

```
prompt> current_design TOP_MULT  
prompt> reset_switching_activity -all -verbose
```

SEE ALSO

```
read_saif(2)  
set_switching_activity(2)
```


reset_timing_derate

Removes all derate factors set on the current design or libraries.

SYNTAX

string **reset_timing_derate**

ARGUMENTS

The **reset_timing_derate** command has no arguments.

DESCRIPTION

The **reset_timing_derate** command removes all timing derates from the current design or libraries. Once the command is run, the result is the same as if timing derates were never set.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example removes derates for all of the objects in the current design or libraries.

```
prompt> reset_timing_derate
```

SEE ALSO

```
set_timing_derate(2)  
report_timing_derate(2)  
report_timing(2)
```

reset_upf

Removes all UPF constraints and UPF data-dependent constraints from the current design. This command is supported only in UPF mode.

SYNTAX

```
status reset_upf
```

ARGUMENTS

The **reset_upf** command has no arguments.

DESCRIPTION

The **reset_upf** command is used to clean up all UPF constraints in the current design. This command returns a 1 if successful and returns a 0 otherwise.

The **reset_upf** command also removes UPF data-dependent constraints, which include the following commands:

```
set_voltage  
set_related_supply_net  
set_always_on_strategy
```

After **reset_upf** is executed and new UPF constraints are reloaded using the **load_upf** command, run the above 3 commands again with respect to the new UPF constraints.

Warning: Executing **reset_upf** has extremely wide-ranging consequences. This command removes all UPF constraints in the current design, with a result that is often equivalent to starting the UPF flow. This command removes UPF constraints without touching the netlist. The command does not check if the netlist has been processed by prior UPF constraints. All commands that might change the netlist after the **load_upf** command will not be backed out by the **reset_upf** command. Remember to verify that the UPF constraints are consistent.

You can choose to use **remove_power_domain**, **remove_supply_net**, and so on to remove a single UPF object.

EXAMPLES

The following example shows a typical use of the *reset_upf* command:

```
prompt> load_upf upf_file1  
prompt> reset_upf  
prompt> load_upf upf_file2  
1
```

The following example shows one of the risks of using the **reset_upf** command. After running the **reset_upf** command, the power domain is removed along with the

```
reset_upf  
2100
```

relationship of the power domain with its related voltage area. So even after reloading the UPF file, you cannot get related power domain information on those voltage areas. In this case, remove the related voltage areas and recreate them with the power domain.

```
prompt> get_voltage_area VA1
VA1
prompt> get_power_domain VA1
VA1
prompt> reset_upf
prompt> load_upf upf_file
prompt> set_fp_rail_voltage_area_constraints \
-voltage_area VA1 -voltage_supply 1.08
Error: Cannot get supply net from UPF in power domain VA1. (PNA-052)
```

SEE ALSO

```
load_upf(2)
save_upf(2)
remove_power_domain(2)
remove_power_switch(2)
remove_supply_net(2)
remove_supply_port(2)
```

resize_objects

Resizes one or more objects.

SYNTAX

```
status resize_objects
{-bbox bounding_box
 | -delta offset
 | -scale scale_factor
 | -width object_width
 | -height object_height
 | -utilization util_factor
 | -aspect aspect
 | -area area_name}
[-keep_placement]
[-keep_pad_to_core_distance]
[-ignore_fixed]
objects
```

Data Types

<i>bounding_box</i>	rectangle
<i>offset</i>	rectangle
<i>scale_factor</i>	point
<i>object_width</i>	real
<i>object_height</i>	real
<i>util_factor</i>	real
<i>aspect</i>	real
<i>area_name</i>	real
<i>objects</i>	collection

ARGUMENTS

-bbox *bounding_box*

Specifies the new bounding box of the object. The *bounding_box* argument can be in either of the following formats: {x1 y1 x2 y2} or {x1 y1} {x2 y2}. The **-bbox** option cannot be used with any of the following options: **-delta**, **-scale**, **-width**, **-height**, **-utilization**, and **-aspect**.

-delta *offset*

Specifies the offset of the current bounding box of the object. The *offset* argument can be in either of the following formats: {x1 y1 x2 y2} or {x1 y1} {x2 y2}. The **-delta** option cannot be used with any of the following options: **-bbox**, **-scale**, **-width**, **-height**, **-utilization**, and **-aspect**.

-scale *scale_factor*

Specifies an x-axis and y-axis scale factor. All objects will be resized by this factor. The **-scale** option cannot be used with any of the following options: **-bbox**, **-delta**, **-width**, **-height**, **-utilization**, and **-aspect**. The scale factor must be positive and non-zero.

`-width object_width`

Specifies the width for a given object (or collection of objects). All objects are resized so that their bounding box has this width; however, the origin will not change. The **-width** option cannot be used with any of the following options: **-bbox**, **-delta**, **-scale**, **-utilization**, and **-aspect**. The **-width** and **-height** options can be used together.

`-height object_height`

Specifies the height for the given object (or collection of objects). All objects are resized so that their bounding box has this height; however, the origin will not change. The **-height** option cannot be used with any of the following options: **-bbox**, **-delta**, **-scale**, **-utilization**, and **-aspect**. The **-width** and **-height** options can be used together.

`-utilization util_factor`

Specifies the required utilization for the object. The object changes size based on *util_factor* value, but keeps the lower-left corner fixed and retains the original aspect ratio. The **-utilization** option cannot be used with any of the following options: **-bbox**, **-delta**, **-width**, **-height**, and **-aspect**. The utilization factor must be positive and non-zero.

Note: this option is usable only for soft macros and movebounds.

`-aspect aspect`

Specifies the required aspect for the object. The object changes size based on the *aspect* value, but keeps the lower-left corner fixed and retains the original area. The **-aspect** option cannot be used with any of the following options: **-bbox**, **-delta**, **-width**, **-height**, and **-utilization**.

`-area area_name`

Specifies a new area.

`-keep_placement`

Keeps existing placement if possible. This option is effective only when applied to the core or the die; it is ignored if applied on any other object. Cells in the core and I/O pad cells are re-placed if necessary. The relative positions of the cells are not guaranteed to be preserved, although an attempt is made to do that when possible. The output of the re-placement is not legalized. The you must explicitly legalize the placement by using the **legalize_placement** command. The default is **false**.

`-keep_pad_to_core_distance`

Maintains the distance between the core and the pad cells; and, by implication, maintains the distance between the core and the die, thus forcing changes to the core and the die to be made in together. If pad cells are absent then the core-to-die distance is still maintained. This option is effective only when applied to the core or the die; it is ignored if applied on any other object. The default is **false**.

`-ignore_fixed`

Resizes fixed objects. By default, fixed objects are not resized.

`objects`

Specifies a list of objects to be resized.

DESCRIPTION

This command resizes a list of objects to either an explicit bounding box, a delta from the current bounding box, an x and y scale, an explicit width and height, a size that gives a required utilization given a constant aspect ratio, or a size that gives a specified aspect ratio given a constant area. Only resizable objects can be used with this command. See the **get_edit_property** man page for a list of resizable objects.

Notes

Fixed objects are not moved unless you use the **-ignore_fixed** option.

Snapping is done automatically using global snap settings.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example resizes all selected objects so that they have an aspect ratio of 2.5.

```
prompt> resize_objects [get_selection]\  
-aspect 2.5
```

The following example resizes all specified objects so that their lower-left corner is at the point {0 0} and they are 100 units wide and 100 units high.

```
prompt> resize_objects [get_selection]\  
-bbox {0 0 100 100}
```

SEE ALSO

```
align_objects(2)  
distribute_objects(2)  
expand_objects(2)  
flip_objects(2)  
get_object_snap_type(2)  
move_objects(2)  
remove_objects(2)  
rotate_objects(2)  
set_object_snap_type(2)
```

resize_polygon

Pushes the edges of the specified polygon inward or outward by the specified distance.

SYNTAX

```
list resize_polygon  
polygon  
-size size
```

Data Types

<i>polygon</i>	list of points
<i>size</i>	double

ARGUMENTS

polygon

Specifies the polygon whose edges are pushed inward or outward by the specified distance. You specify a polygon in the following format: `{{x1 y1} {x2 y2} ... {xN yN} {x1 y1}}`. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point. The coordinate unit is specified in the technology file; typically it is microns.

-size *size*

Specifies the distance used to adjust the edges of the polygon. The coordinate unit is specified in the technology file; typically it is microns.

DESCRIPTION

This command pushes the edges of the specified polygon inward or outward by the specified distance and returns the list of polygons whose edges have been moved. Each polygon in the list is represented as a list of points.

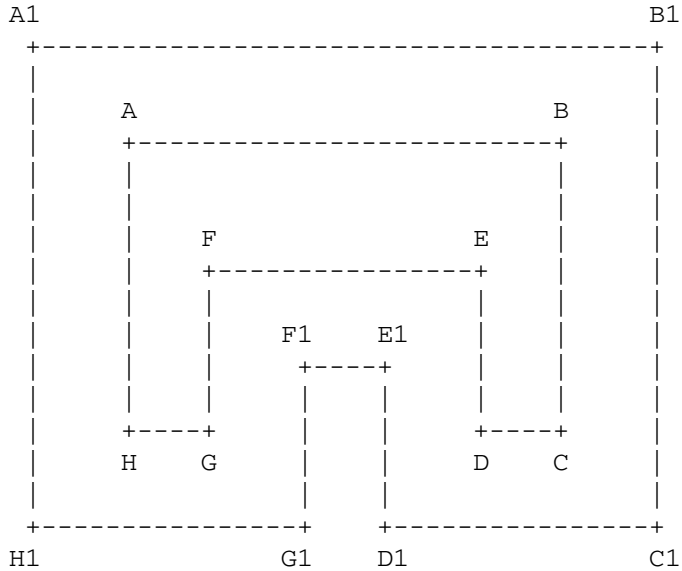
If you specify a positive value for the **-size** option, the edges of the polygon are pushed outward to increase the size of the polygon, which is referred to as oversizing the polygon. The tool fills any gaps that are less than two times the distance unit. When you increase the polygon's size, the command returns a single polygon. You can directly pass this result to another polygon command, such as **convert_from_polygons**, **get_polygon_area**, **resize_polygon**, or **compute_polygons**.

If you specify a negative value for the **-size** option, the edges of the polygon are pushed inward and the size of the polygon is reduced, which is referred to as undersizing the polygon. When you decrease the polygon's size, the result can be a disjoint rectilinear region that is represented by more than one polygon. Because the polygon commands take a single polygon as input, you cannot directly pass the result as a parameter to another polygon command. Instead, you must use a Tcl list command, such as the **foreach** or **lindex** command, to extract each polygon from the returned list, and then pass each polygon to the polygon command.

You must open the Milkyway design library before using this command.

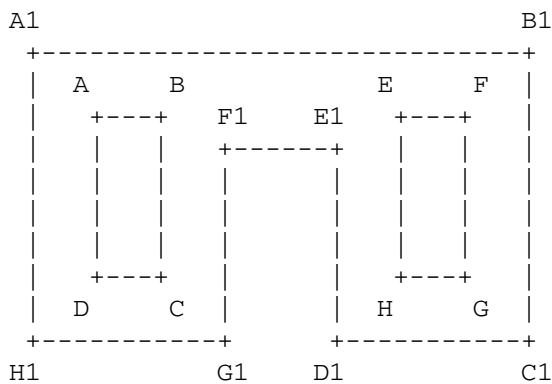
EXAMPLES

The following example returns the A1-B1-C1-D1-E1-F1-G1-H1-A1 polygon, which is oversized from the A-B-C-D-E-F-G-H-A polygon by 10 units.



```
prompt> resize_polygon {{10 40} {60 40} {60 10} {50 10} {50 30} \
{20 30} {20 10} {10 10} {10 40}} -size 10
{{70 50} {70 0} {40} {40 20} {30 20} {30 0} {0 0} {0 50} {70 50}}
```

The following example returns two polygons, A-B-C-D-A and E-F-G-H-E, after undersizing the A1-B1-C1-D1-E1-F1-G1-H1-A1 polygon by 10 units.



```
prompt> resize_polygon {{0 45} {80 45} {80 0} {50 0} {50 30} \
{30 30} {30 0} {0 0} {0 45}} -size -10
{{10 35} {20 35} {20 10} {10 10} {10 35}} {{60 35} {70 35} {70 10}
{60 10} {60 35}}
```

The following example passes the result of the **resize_polygon** command to another **resize_polygon** command.


```
prompt> resize_polygon -size -10 \  
[resize_polygon {{10 40} {50 40} {50 10} \  
{40 10} {40 30} {20 30} {20 10} {10 10} {10 40}} -size 10]  
{10 40} {50 40} {50 10} {10 10} {10 40}
```

SEE ALSO

```
convert_to_polygon(2)  
convert_from_polygon(2)  
compute_polygons(2)
```

restore_design_settings

Restores the saved design settings.

SYNTAX

```
status restore_design_settings  
-library
```

ARGUMENTS

```
-library  
    Restores design settings that were saved to the current library.
```

DESCRIPTION

This command restores the design settings that were previously saved with the **save_design_settings** command. The design settings stored in the current library are applied to the current session. Before you restore the saved design settings, you can use the **write_design_settings** command to see which variables will be overwritten (and their new values).

EXAMPLES

The following example restores the design settings saved in the current library.

```
prompt> restore_design_settings -library  
1
```

SEE ALSO

```
save_design_settings(2)  
write_design_settings(2)
```

rotate_objects

Rotates one or more objects either geometrically or by setting the current orientation.

SYNTAX

```
new_objects rotate_objects
{-to orient
  | -by rotate
  | -mirror mirror}
[{-pivot point | -anchor anchor_point}]
[-ignore_fixed]
objects
```

Data Types

<i>orient</i>	string
<i>rotate</i>	string
<i>mirror</i>	string
<i>anchor_point</i>	string
<i>objects</i>	collection

ARGUMENTS

-to orient

Specifies the orientation of the object using either Design Exchange Format (DEF) or Synopsys notation.

The valid values are as follows:

N, W, S, E, FN, FS, FE, FW

NW, NE, EN, ES, SE, SW, WN, WS

he following groups of values are synonymous:

N , NE

W , WN

S , SW

E , ES

FN, NW

FS, SE

FE, EN

FW, WS

The -to option, -by and -mirror options are mutually exclusive. You must include one of these options.

The default is **N**.

-by rotate

Specifies the change in orientation of the object.

The valid values are as follows:

90, 180, 270

CW90, CW180, CW270

CCW90, CCW180, CCW270

The following values are synonymous:

90 , CW270, CCW90

180, CW180, CCW180

270, CW90 , CCW270

The `-to` option, `-by` and `-mirror` options are mutually exclusive. You must include one of these options.

`-mirror mirror`

Specifies which direction to mirror the object:

X - Mirror X

Y - Mirror Y

The `-to` option, `-by` and `-mirror` options are mutually exclusive. You must include one of these options.

`-pivot point`

Specifies the point for pivoting.

The `-pivot` option and `-anchor` option are mutually exclusive. You can specify only one of them. If both are omitted, the default uses the center of the bounding box of the object or collection of objects.

This option only affects the `-by` option.

`-anchor anchor_point`

Specifies the position of the pivot point in relation to the objects being rotated. One of:

ll - Lower Left

ur - Upper Right

center - Center

The `-pivot` option and `-anchor` option are mutually exclusive. You can specify only one of them. If both are omitted, the default uses the center of the bounding box of the object or collection of objects.

This option only affects the `-by` option.

The default is **center**.

`-ignore_fixed`

Specifies that fixed edit objects are rotated as well as the usual non-fixed edit ones.

`objects`

Specifies the list of objects to be rotated.

RETURNS

`new_objects`

Contains the list of rotated objects.

Note: In most cases this list is the same as the input list of objects but in some cases, e.g. when a vertical wire is rotated to become a new horizontal wire, the list may be different.

DESCRIPTION

This command rotates the specified objects around a pivot point by the specified absolute (`-to` option) or delta (`-by` option) orientation.

Transformable objects have their current orientation updated. Other resizable objects have their boundary rotated and reset.

`rotate_objects`

2110

See man page for `get_edit_property` command for a list of which objects can be transformed and which can have their boundary set.

Note: Snapping is done automatically using global snap settings.

DESCRIPTION

This command rotates and orients a list of objects rigidly about a given pivot point or computed point.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example rotates cell `nand23` by 90 degrees clockwise:

```
prompt> rotate_objects -by CW90 [get_cells nand23]
```

SEE ALSO

`move_objects(2)`
`remove_objects(2)`
`resize_objects(2)`
`align_objects(2)`
`distribute_objects(2)`
`expand_objects(2)`
`flip_objects(2)`
`set_object_snap_type(2)`
`get_object_snap_type(2)`
`set_object_fixed_edit(2)`
`get_object_fixed_edit(2)`

route_area

Performs detail routing in specified area of the design.

SYNTAX

```
status route_area
-within {{llx1 lly1} {urx1 ury1} ....}
[-mode (initial_route | search_and_repair | optimize)]
[-loop num]
[-run_time_limit time]
```

Data Types

<i>llx1</i>	integer
<i>lly1</i>	integer
<i>urx1</i>	integer
<i>ury1</i>	integer
<i>num</i>	integer
<i>time</i>	integer

ARGUMENTS

-within {{*llx1 lly1*} {*urx1 ury1*}}
Specifies either an area or areas to perform detail route.

-mode (initial_route | search_and_repair | optimize)
Runs initial detail route or search & repair or optimization. Default is initial_route.

-loop *num*
Specifies the number of search & repair loops. Default value is 20. Range is 0 to 200. This option works with -mode values search_and_repair and optimize.

-run_time_limit *time*
Specifies the runtime limit for detail route in minutes. The default value is -1 (it means no limit). Range of **time** is -1 to 500.

DESCRIPTION

This command is used to perform detail routing in a specified area or areas in the design. The option **-within** is mandatory.

Prerequisites

Global route the design using **route_global** command.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following command performs area routing using the **-within**, **-mode**, **-loop**, and **-run_time_limit** options.

```
prompt> route_area -within {{100 100} {200 200}}\  
      -mode search_and_repair -loop 5 -run_time_limit 50
```

SEE ALSO

`route_auto(2)`
`set_route_options(2)`

route_auto

Performs global routing, track assignment, detail routing, and search and repair in one step.

SYNTAX

```
status route_auto
[-no_global]
[-no_track]
[-no_detail]
[-search_repair_loop num]
[-effort (minimum | low | medium | high)]
[-save_after_global_route]
[-save_after_track]
[-save_after_detail_route]
[-num_cpus num]
```

Data Types

num integer

ARGUMENTS

-no_global
Skips the global routing step.

-no_track
Skips the track assignment step.

-no_detail
Skips the detail routing step.

-search_repair_loop *num*
Specifies the number of search and repair loops. The default is 0. The range is 0 to 500.

-effort (minimum | low | medium | high)
Specifies the global route effort. The default is medium.

-save_after_global_route
Saves the design after global route.

-save_after_track
Saves the design after track assignment.

-save_after_detail_route
Saves the design after detail route.

-num_cpus *num*
Specifies the number of CPUs for distributed routing. The default is 0. The range is 0 to 500.

DESCRIPTION

This command is used to run global route, track assignment, detail route and search & repair in one step.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following command runs auto route using the **-search_repair_loop**, **-effort**, **-save_after_global_route**, **-save_after_track**, **-save_after_detail_route**, and **-num_cpus** options.

```
prompt> route_auto -search_repair_loop 5\  
-effort high -save_after_global_route\  
-save_after_track -save_after_detail_route\  
-num_cpus 1
```

SEE ALSO

```
route_detail(2)  
route_global(2)  
route_search_repair(2)  
route_track(2)  
set_route_options(2)
```

route_detail

Performs detail routing on the design.

SYNTAX

```
status route_detail
[-track_assign (auto | skip)]
[-search_repair_loop num]
[-run_time_limit num]
[-num_cpus num]
```

Data Types

num integer

ARGUMENTS

-track_assign (auto | skip)
Specifies if track assignment should be run. The default value is auto i.e., the tool decides if track assignment should be run.

-search_repair_loop *num*
Specifies the number of search and repair loops to be run. The default value is zero. Range is 0 to 500.

-run_time_limit *num*
Specifies the cpu runtime limit in minutes. The default is no limit. Range is -1 to 500.

-num_cpus *num*
Specifies the number of CPUs for distributed routing. Number must be >= 1.

DESCRIPTION

This command uses the general pathways suggested by the global routing and track assignment processes to route the nets (paths and contacts).

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example performs detail routing using the **-track_assign**, **-search_repair_loop**, **-run_time_limit**, and **-num_cpus** options.

```
prompt> route_detail -track_assign skip\  
-search_repair_loop 5 -run_time_limit 50\  
route_detail
```

-num_cpus 2

SEE ALSO

`route_auto(2)`
`set_route_options(2)`

route_differential

Routes the nets in the specified differential groups.

SYNTAX

```
status route_differential  
-groups group_names  
[-ignore_global]  
[-ignore_track_assign]  
[-ignore_detail]  
[-search_repair_loop int]
```

Data Types

<i>group_names</i>	list
<i>int</i>	integer

ARGUMENTS

-groups *group_names*
Specified the name(s) of groups to route.

-ignore_global
Do not run the global route on the specified group(s). Route the differential group(s) on default.

-ignore_track_assign
Do not run the track assignment on the specified group(s). Route the differential group(s) on default.

-ignore_detail
Do not run the detail route on the specified group(s). Route the differential group(s) on default.

-search_repair_loop *int*
Enter in the number of Search and Repair loops that you want to control.
Default is 5.

DESCRIPTION

Prerequisites : You need to define the differential group(s) with the `create_differential_group` command before using this command.

Route the nets in the given differential groups. This command uses the normal global route/track-assignment/detail route flow to route.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example is to route specified group, DGroup1, on all three routing phases (global route, track assignment, and detail route), and only try to search&repair once.

```
prompt> route_differential -groups DGroup1 -search_repair_loop 1
```

SEE ALSO

`create_differential_group(2)`

route_eco

Performs ECO routing on the design.

SYNTAX

```
status route_eco
[-no_global]
[-no_track]
[-no_detail]
[-auto]
[-region_based file]
[-distributed]
[-num_cpus num]
[-search_repair_loop num]
[-utilize_dangling_wires]
[-scope (global | local)]
[-reroute (modified_nets_only | modified_nets_first_then_others | any_nets)]
[-freeze_routing_on_layer name]
[-freeze_vias_on_frozen_metal]
```

Data Types

<i>file</i>	string
<i>num</i>	integer
<i>name</i>	string

ARGUMENTS

-no_global
Skips global routing.

-no_track
Skips track assignment.

-no_detail
Skips detail routing.

-auto
Decides which stages to run.

-region_based *file*
Performs region based ECO routing on nets specified in *file*. Requires **-auto**.

-distributed
Performs distributed region based ECO routing. Requires **-auto** and **-region_based**.

-num_cpus *num*
Specifies number of cpus for distributed routing. Number must be ≥ 1 .

-search_repair_loop *num*
Specifies the number of search and repair loops. Default is 20. Range is 0

to 500.

`-utilize_dangling_wires`

Utilizes dangling wires. Default option is to discard.

`-scope (global | local)`

The default value is **global**. Use **global** when the design has DRC violations. The router not only connects the ECO nets but also fixes the DRC violations in the design. Use **local** option when the design is almost DRC clean and only the ECO nets need to be reconnected.

`-reroute (modified_nets_only | modified_nets_first_then_others | any_nets)`

The default value is **any_nets**. For the option **modified_nets_only**, the router freezes all the fully connected nets and tries to finish the routing by modifying only the nets with open ECO changes. This might cause the router to fail when fixing some violations. It is only suitable for very minor ECO changes.

For the option **modified_nets_first_then_others**, the router first freezes all the fully connected nets and tries to finish the routing by only modifying the nets with open ECO changes. When there are violations remaining, the router tries to reroute those fully connected nets to resolve the violations. This reduces the layout changes for minor ECO operations but is not suitable for major ECO changes.

For the option **any_nets**, The router reroutes any nets freely to fix all the DRC violations.

`-freeze_routing_on_layer name`

Freezes routing on the layers whose names are specified, ex m1. Default is none.

`-freeze_vias_on_frozen_metal`

Freezes vias on frozen metal layers. Works with **-freeze_routing_on_layer** option.

DESCRIPTION

This command performs routing for the broken nets. It runs global route, track assignment and detail route.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example performs ECO routing using the **-auto**, **-search_repair_loop**, **-utilize_dangling_wires**, **-scope**, and **-reroute** options.

```
prompt> route_eco -auto -search_repair_loop 5\  
-utilize_dangling_wires -scope local\  
-reroute modified_nets_first_then_others
```

SEE ALSO

`set_distributed_route(2)`
`set_route_options(2)`

route_flip_chip

Routes flip-chip nets on a single metal layer.

SYNTAX

```
status route_flip_chip
[-nets | -nets_in_file nets_file]
[-route_by_input_net_order bool]
[-45_degree bool]
-routing_layer layer
[-terminal_names {list_of_terminal_names}]
```

Data Types

<i>collection_of_nets</i>	collection
<i>nets_file</i>	string
<i>layer</i>	string
<i>list_of_terminal_names</i>	list

DATA TYPES

<i>nets_file</i>	string
<i>collection_of_nets</i>	collection
<i>layer</i>	string
<i>list_of_terminal_names</i>	list

ARGUMENTS

-nets {*collection_of_nets*}
Specifies the flip-chip nets to route.

-nets_in_file *nets_file*
Specifies the name of the file that contains the list of flip-chip nets to route.
This option and the **-nets** option are mutually exclusive; you can specify only one. If you do not specify either option, all flip-chip nets are routed.

-route_by_input_net_order *bool*
Routes the nets in the order in which they are specified in the **-nets** or **-net_in_file** option. By default, the tool sorts the nets based on their locations.

-45_degree *bool*
The flip-chip router will route nets with 45 degree paths.

-routing_layer [*layer*]
Specifies the target layer for flip-chip routing. The range is m1 to m15.
This option is required.

-terminal_names {*list_of_terminal_names*}
Use this option for nets with multiple driver pins, also known as multi-

terminal nets, or for electrical equivalent (EEQ) pins. Sometimes a flip-chip net can be connected to several driver pins or EEQ pins. The **-terminal_names** option allows you to selectively route from the specified driver pins or EEQ pins. The syntax for terminal names is *instance_name/pin_name* for multi-terminal nets and *instance_name/port_name:pin_name* for EEQ pins. If you do not specify the **-nets** or **-nets_in_file** option, only the specified driver pins or EEQ pins will be routed.

DESCRIPTION

This command routes flip-chip nets using the flip-chip router. In addition, it can rip up and reroute Manhattan wires.

Flip-chip routing is a special application where a shape-based router routes nets on a single layer. It meets the following requirements:

- Provides a routing tool to route, rip-up and reroute automatically
- Optimizes routing patterns to eliminate jogs and shorten wire length
- Checks DRCs and opens for routed wires
- Routes flip-chip nets on a single layer

The suggested flow for flip-chip routing is as follows:

1. Generate a file listing the flip-chip nets in the design by running the **write_flip_chip_nets** command.
2. Set the flip-chip routing options by running the **set_route_flip_chip_options** command.
3. Create stacked vias on the driver cells by running the **create_stack_via_on_pad_pin** command.
4. Verify that the stacked vias do not violate routing DRCS by running the **verify_route** command if you are using the classic router or **verify_zrt_route** if you are using Zroute.
5. Perform flip-chip routing by running the **route_flip_chip** command two or three times.
6. Display the remaining open nets by running the **display_flip_chip_route_flylines** command.
7. Manually route the open nets, if there are any, by using the **remove_flip_chip_route**, **push_flip_chip_route** and **route_flip_chip** commands.
8. After all flip-chip nets are routed, improve and enhance the routed wires by running the **optimize_flip_chip_route** command.
9. Verify that the flip-chip nets do not violate routing DRCS by running the **verify_route** command if you are using the classic router or **verify_zrt_route** if you are using Zroute.

EXAMPLES

The following example routes the flip-chip nets specified in the RDLNetFile:

1. Route flip-chip nets recorded in a net file:
prompt> route_flip_chip -nets_in_file RDLNetFile -routing_layer m9
2. Route all flip-chip nets:

```
prompt> route_flip_chip
```

The following command specifies the `terminal_names` to route VSS net connected to EEQ pins "VSS", "VSS 1", and "VSS 5" of instance I_1.

```
prompt> route_flip_chip -routing_layer m9 -nets VSS -terminal_names {"I_1/  
VSS:VSS" "I_1/VSS:VSS 1" "I_1/VSS:VSS 5"}
```

SEE ALSO

```
set_route_flip_chip_options(2)  
remove_flip_chip_route(2)  
create_stack_via_on_pad_pin(2)  
write_flip_chip_nets(2)  
push_flip_chip_route(2)  
display_flip_chip_route_flylines(2)  
optimize_flip_chip_route(2)
```

route_fp_proto

Performs quick global routing on the design.

SYNTAX

```
status route_fp_proto
[-effort (low | medium)]
[-congestion_map_only]
[-track_assignment]
```

ARGUMENTS

-effort (low | medium)
Performs global routing. The default value of *effort* is low.

-congestion_map_only
Creates only the congestion map without creating glinks.

-track_assignment
Performs track assignment.

DESCRIPTION

This command maps general pathways through the design for each unrouted net (signal nets and clock nets). The global router uses a three-dimensional array of global routing cells to model the demand and capacity of global routing. The average height of the standard cells is used to create the height and width of each global routing cell.

During global routing, the tool assigns nets to the global routing cells through which they will pass. For each global routing cell, routing capacity is calculated according to the blockages, pins, and routing tracks inside the cell.

Although global routing does not assign the nets to the actual wire tracks, it notes the number of nets assigned to each global routing cell. The tool calculates the demand for wire tracks in each global routing cell and reports the overflows, which are the number of wire tracks still needed after the tool assigns nets to the available wire tracks in a global routing cell.

The tool might reduce overflows by detouring nets around congested areas and increasing the wire length. When a wire is prerouted, the tool checks to see whether it is completely connected. If the net is completely connected, the tool treats the net as a blockage; if the net is partially connected, the tool connects all parts of the net.

- * The tool treats power and ground nets as blockages.
- * The tool treats prerouted meshes and trunks as a single connection.

Make sure you examine the routing to see whether it is connected properly.

The **route_fp_proto** command uses the same algorithm as the **route_global** command but

internally sets some parameters to achieve a faster result, at some expense in quality. It is intended to give a quicker estimate of routability and congestion.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example performs proto routing using the **-effort** option.

```
prompt> route_fp_proto -effort medium
```

SEE ALSO

route_global(2)

route_global

Performs global routing on the design.

SYNTAX

```
status route_global
[-effort (minimum | low | medium | high)]
[-congestion_map_only]
```

ARGUMENTS

```
-effort (minimum | low | medium | high)
    Performs global routing. The default value of effort is medium.

-congestion_map_only
    Create only the congestion map without creating glinks.
```

DESCRIPTION

This command maps general pathways through the design for each unrouted net (signal nets and clock nets). The global router uses a three-dimensional array of global routing cells to model the demand and capacity of global routing. The average height of the standard cells is used to create the height and width of each global routing cell.

During global routing, the tool assigns nets to the global routing cells through which they will pass. For each global routing cell, routing capacity is calculated according to the blockages, pins, and routing tracks inside the cell. Although global routing does not assign the nets to the actual wire tracks, it notes the number of nets assigned to each global routing cell.

The tool calculates the demand for wire tracks in each global routing cell and reports the overflows, which are the number of wire tracks still needed after The tool assigns nets to the available wire tracks in a global routing cell. The tool might reduce overflows by detouring nets around congested areas and increasing the wire length. When a wire is prerouted, the tool checks to see whether it is completely connected. If the net is completely connected, the tool treats the net as a blockage; if the net is partially connected, the tool connects all parts of the net.

- * The tool treats power and ground nets as blockages.
- * The tool treats prerouted meshes and trunks as a single connection.

Make sure you examine the routing to see whether it is connected properly.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following command performs global routing using the **-effort** option.

```
prompt> route_global -effort high
```

SEE ALSO

route_auto(2)
set_route_options(2)

route_group

Performs routing on a group of nets in the design.

SYNTAX

```
status route_group
[-nets nets | -all_clock_nets]
[-no_global]
[-no_track]
[-no_detail]
[-search_repair_loop num]
[-utilize_dangling_wires]
[-trim_antenna_of_user_wires]
[-dont_optimize_route_pattern]
[-num_cpus num]
```

Data Types

<i>nets</i>	collection
<i>num</i>	integer

ARGUMENTS

-nets *nets*
Specifies the collection of nets to be routed. This option and the **-all_clock_nets** option are mutually exclusive; you can specify only one.

-all_clock_nets
Routes all clock nets. This option and the **-nets** option are mutually exclusive; you can specify only one.

-no_global
Skips global routing.

-no_track
Skips track assignment.

-no_detail
Skips detail routing.

-search_repair_loop *num*
Specifies the number of search and repair loops. The default is 5. The range is 0 to 500.

-utilize_dangling_wires
Utilizes dangling wires. If you do not specify this option, dangling wires are discarded.

-trim_antenna_of_user_wires
Trims wire segments of preroutes.

`-dont_optimize_route_pattern`
Skips optimizing the route pattern.

`-num_cpus num`
Specifies the number of CPUs for distributed routing. The default value is 0. The range is 0 to 63.

DESCRIPTION

This command is used to route a specified group of nets. It is useful for discovering timing problems. For example, you can route clock and bus pins to prerouted clock and bus wires, then perform timing analysis on these nets. If the timing results are satisfactory, you can route the remaining nets by using the global routing, track assignment, and detail routing commands. You should route critical clocks nets before routing other signal nets so that the clock nets have the most direct routing.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example performs routing for a group of nets using the following options: `-all_clock_nets`, `-search_repair_loop`, `-utilize_dangling_wires`, `-trim_antenna_of_user_wires`, `-dont_optimize_route_pattern`, and `-num_cpus`.

```
prompt> route_group -all_clock_nets \  
-search_repair_loop 6 -utilize_dangling_wires \  
-trim_antenna_of_user_wires \  
-dont_optimize_route_pattern -num_cpus 1
```

SEE ALSO

`route_detail(2)`
`route_global(2)`
`route_track(2)`
`route_search_repair(2)`
`set_route_options(2)`

route_htree

Creates routes of H, I, or T (rotated H, rotated I, or rotated T) shape on the specified nets.

SYNTAX

```
status route_htreefp
-nets list_of_nets
-layers {horizontal_layer vertical_layer}
[-orientation list_of_three_orientation_values]
[-allow_off_grid]
[-allow_violation]
[-root root_buffer_of_the_H/I_tree]
[-mesh_net mesh_net]
```

Data Types

<i>list_of_nets</i>	collection
<i>horizontal_layer</i>	string
<i>vertical_layer</i>	string
<i>list_of_three_orientation_values</i>	collection

ARGUMENTS

-nets *list_of_nets*
Indicates the list of nets to create routes. This is a required option.

-layers {*horizontal_layer vertical_layer*}
Indicates the numbers or names of horizontal and vertical layers to be used for the routing. This is a required option.

-orientation *list_of_three_orientation_values*
Specifies the preferred orientation to create the routes. You can provide up to three orientation values: H route, T route, and I route.

-allow_off_grid
Enables the allowance for tracks that might not align to the routing grids.

-allow_violation
Enables the allowance of DRC violations during routing.

DESCRIPTION

The route_htree command creates routes of the following shapes: H , H_90(rotated-H), I, I_90(rotated-I),T, T_90(rotated-T) routes, for example, T_90/T_180/T_270. The T shape is considered counter clockwise. For 5-pin nets (4 loads and 1 driver), the command creates routes of H or H_90 shape; for 3-pin nets, routes of I or I_90 shape; for 4-pin nets, routes of T, T_90, T_180, or T_270 shape. The 90 indication represents 90 degree counter clockwise rotation.

The /fBorientation/fP option specifies the default routing orientation for H, I, and

T routes. However, if the default does not produce good route, the command will try other possible orientations. If this option is not specified, the command determines a default orientation for the nets.

By default, the command selects tracks aligned to the routing grids during routing. If the command cannot find suitable on-grid tracks, it fails to Route. However, you can use the /fBallow_off_grid/fP option to select off-grid tracks in the proximity for routing.

The command does not create any routes when no DRC violation is found in creating the H, T, and I shaped routes. However, you can set the /fB-allow_violation/fP option for the command to create routes even with no DRC violation.

After creating route topology such as H/T/I/H_90/T_270, the comb router will connect pins to the nets using all available layers.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following shows an example of how to use **route_htree**:

```
prompt>route_htree -nets [get_nets {Htree_ck_gclkn_driver_x_L2* \  
Htree_ck_gclkn_driver_x_L3*}] -layers {M6 M7}\  
-orientation {H T_180 I_90}
```

For the nets specified in the above example, the **route_htree** will create H, inverted T, or rotated I shaped routes for 4-, 3- or 2-load-pin nets, respectively. The horizontal routes will be on M6 and vertical routes will be on M7. The routes will be aligned to tracks.

```
prompt>route_htree -nets [get_nets Htree_ck_gclkn_driver_x_L2*]\  
-layers {M6 M7}\  
-allow_off_grid
```

For the nets with the name prefix Htree_ck_gclkn_driver_x_L2 and Htree_ck_gclkn_driver_x_L3 in this example, appropriate routes such as H, T, or I shape will be created per the number of load pins. However, the default orientation is determined by the command. The routes might not be aligned with the tracks.

SEE ALSO

add_clock_drivers(2)

route_mesh_net

Routes the connections from the clock mesh wire grid to the associated drivers and loads.

SYNTAX

```
status route_mesh_net
-net net_name
[-mode comb | fishbone]
[-max_backbone_fanout -max_backbone_fanout_value]
[-max_span max_span_value]
[-backbone_dir horizontal | vertical | both]
[-max_rc_delay float]
```

Data Types

net_name string

ARGUMENTS

-net *net_name*
Specifies only one net to be routed.

-mode comb | fishbone
Specifies the topology to be employed when you run the routing command. The comb mode has a thicker segment with many small segments coming off perpendicularly. The fishbone mode builds fishbone structures including the backbones grown from the mesh stripes and many fingers connecting the drivers and loads to the backbones. The default is comb mode.

-max_backbone_fanout *-max_backbone_fanout_value*
The maximum number of fanouts on the backbone of one single fishbone structure. This option allows the user to control the number of loads on one single fishbone structure. This option only works with fishbone mode. By default, the max_backbone_fanout constraint is disabled.

-max_span *max_span_value*
The maximum span of one fishbone structure. This options allows the user to control the size of fishbone structures. The span of a fishbone structure is defined as the width of the bounding box for the whole fishbone in the direction perpendicular to the backbone. This option only works with fishbone mode. By default, the max_span constraint is disabled.

-backbone_dir horizontal | vertical | both
The direction of the backbones of fishbone structures. User can choose horizontal or vertical direction to route the backbone, or specify both to let the tool to decide the direction for each backbone. If both is specified, some backbones may be routed in horizontal direction and others may be routed in vertical direction. This option only works with fishbone mode. The default value of backbone_dir is both.

DESCRIPTION

To design a clock mesh, first you need to create a grid of relatively thick routing wires, which is called a mesh, and then allocate the buffers that drive the mesh. Use the **route_mesh_net** command to attach the drivers and loads to the mesh.

FILES

No file is required or generated by this command.

LIMITATIONS

In general, the purpose of mesh nets is to reduce clock skew at the load points. The **route_mesh_net** command operates at a local level and does not attempt to reduce skew. That is, the connections to the routing mesh might vary in length and delay. For effective mesh results, create a mesh with enough horizontal and vertical wires so that no load is far apart from the mesh.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command is invoked after the wire mesh and all the drivers have been inserted.

```
prompt> route_mesh_net -net clk1 -mode balanced
```

SEE ALSO

```
create_clock_mesh(2)
add_clock_drivers(2)
compile_premesh_tree(2)
analyze_subcircuit(2)
compile_clock_tree(2)
```

route_opt

Performs simultaneous routing and post-route optimization on the design.

SYNTAX

```
status route_opt
[-effort low | medium | high]
[-xtalk_reduction]
[-only_xtalk_reduction]
[-skip_initial_route]
[-stage global | track | detail]
[-power]
[-incremental]
[-size_only]
[-optimize_wire_via]
[-area_recovery]
[-wire_size]
[-only_wire_size]
[-only_hold_time]
[-only_design_rule]
[-only_power_recovery]
[-only_area_recovery]
[-num_cpus number]
[-initial_route_only]
```

ARGUMENTS

-effort low | medium | high
Specifies the effort level for postroute optimizations in the **route_opt** flow. This option does not change the routing steps performed by **route_opt**. Valid values are **low**, **medium**, and **high**. In **high** effort, the tool is more aggressive during optimization and runs three optimization loops. The default effort level is **medium**.

-xtalk_reduction
Runs routing-based crosstalk reduction signal integrity (SI) optimization. By default, this option is off. This switch is applicable only if you enable signal integrity mode before running **route_opt**. You can enable signal integrity mode by setting either the **-delta_delay** or **-static_noise** option to true in **set_si_options**. By default, these options are set to false in **set_si_options**. If you specify the **-xtalk_reduction** option without enabling signal integrity mode, the tool issues an error message and stops.

-only_xtalk_reduction
Runs only routing-based crosstalk reduction. By default, this option is off. Similar to **-xtalk_reduction**, this switch is applicable only if you enable signal integrity mode running **route_opt**. If you specify this option without enabling signal integrity mode or if you specify any other options with the **-only_xtalk_reduction** option, the tool issues an error message and stops.

-skip_initial_route
Runs the topology-based optimization steps without initial routing. By

default, this option is off. Using this switch on a non-detail-routed design might generate erroneous results; therefore, you should verify that the design is fully routed before using this option. This option can not be used with the **-initial_route_only**, **-incremental**, or **-stage** options.

-stage global | track | detail

Specifies the stage to run routing and topology-based optimization. If you do not specify this option, the tool first runs global routing, track assignment, and detail routing. When you specify the stage "global", the tool runs only global routing along with optimization. When you specify the stage "track", the tool removes existing routing and re-runs global routing and track assignment along with optimization. When you specify the stage "detail", the tool runs detail routing along with optimization. By default, this option is off.

-power

Enables topology-based leakage power optimization as part of the optimization loop in the `route_opt` flow. By default, this option is off.

-incremental

Disables running of the initial routing steps. Runs only topology-based incremental optimization and ECO routing. By default, this option is off.

-size_only

Performs only sizing of cells in topology-based optimization. By default, this option is off. You can use this option for designs very sensitive to postroute optimization changes. You can control the sizing option by setting effort. Effort low will use `preserve_footprint` sizing. Effort medium will use `in_place` sizing. Effort high will use regular density based sizing. And the default effort is medium.

-optimize_wire_via

Runs an extra wire length and via count optimization. This option cannot be used with `route_opt` in Zroute mode. By default, this option is off.

-area_recovery

Enables area recovery for cells that are not on timing critical paths. By default, this option is off.

-wire_size

Enables the use of an additional technique of sizing the width of route segments to fix setup time violations during topology-based optimization. Needs existing NDR, and it can change netlist. Touched nets will have NDRs applied, and the applied NDRs will not be removed after wire sizing. By default, this option is off.

-only_wire_size

Performs only sizing of wires during topology-based optimization. Touched nets will have NDRs applied, and the applied NDRs will not be removed after wire sizing. This option can be used only with the **-incremental** option. By default, this option is off.

-only_hold_time

Specifies that only hold time violations are fixed. Applicable only if hold fixing is enabled using the **set_fix_hold** command before running **route_opt**.

This switch can be used only with the **-incremental** option. By default, this option is off.

-only_design_rule

If **-only_design_rule** is specified, the tool performs only design rule fixing, and timing optimizations are not performed. This option can be used only with the **-incremental** option. By default, this option is off.

-only_power_recovery

Specifies that only leakage power optimizations are performed. This option can be used only with the **-incremental** option. By default, this option is off.

-only_area_recovery

Specifies that only area optimizations are performed. This option can be used only with the **-incremental** option. By default, this option is off.

-num_cpus *number*

Specifies the number of CPUs for detail routing and search and repair routing. You can use up to 63 CPUs. By default, this option is off. This option is applicable only if you set up multiple CPU mode by using the **set_distributed_route** command prior to running. This option cannot be used with **route_opt** in Zroute mode. **route_opt**.

-initial_route_only

Runs only the initial routing stage (it runs global routing, track assignment, detail routing, and search and repair). By default, this option is off.

DESCRIPTION

This command performs simultaneous routing and postroute optimization on the current design. The output of this command is a postroute optimized design. The **route_opt** flow runs differently depending on whether signal integrity mode is enabled.

A default (medium effort) signal-integrity-enabled **route_opt** flow includes:

1. Initial routing: Global routing, track assignment, detail routing, and search and repair
2. Topology-based optimization (adaptive) without considering signal integrity
3. Topology-based signal integrity optimization

A default (medium effort) signal-integrity-disabled **route_opt** flow includes:

1. Initial routing: Global routing, track assignment, detail routing, and search and repair
2. Topology-based optimization without considering signal integrity

You can add routing-based crosstalk reduction and wire length and via count optimization to the default flow by using the **-xtalk_reduction** and **-optimize_wire_via** options. The signal-integrity-enabled **route_opt** flow is then changed to:

1. Initial routing: Global routing, track assignment, detail routing, and search and repair

2. Optimization of wire length and via count
3. Topology-based optimization (adaptive) without considering signal integrity
4. Routing-based crosstalk reduction
5. Topology-based signal integrity optimization

Multicorner-Multimode Support

This command uses information from active scenarios only.

EXAMPLES

The following example runs initial routing and stops without running optimization:

```
prompt> route_opt -initial_route_only
```

The following example performs only global routing:

```
prompt> route_opt -initial_route_only -stage global
```

The following example performs only global routing and track assignment:

```
prompt> route_opt -initial_route_only -stage track
```

The following example runs only detail routing and search and repair:

```
prompt> route_opt -initial_route_only -stage detail
```

The following example runs only optimization and skips the initial routing steps:

```
prompt> route_opt -skip_initial_route
```

The following example runs only incremental optimization:

```
prompt> route_opt -incremental
```

The following example runs routing-based crosstalk reduction in addition to the default route_opt signal integrity flow:

```
prompt> route_opt -xtalk_reduction
```

SEE ALSO

clock_opt(2)
extract_rc(2)
place_opt(2)
set_route_opt_strategy(2)

route_rc_reduction

Optimizes the timing and crosstalk on a fully-routed design, based on the detail routing results. This command works only on completely detail-routed designs that include all the necessary timing information.

SYNTAX

```
status route_rc_reduction
[-slack_target slack_target_number]
[-setup_slack_effort optimize | preserve]
[-max_transition]
[-max_capacitance]
[-crosstalk_noise]
[-run_time_limit number]
[-opt_delay_loop number]
[-opt_delay_search_repair_loop number]
```

Data Types

<i>slack_target_number</i>	integer
<i>number</i>	integer

ARGUMENTS

-slack_target *slack_target_number*
Specifies the slack target of delay opt (default is 0.1).

-setup_slack_effort optimize | preserve
Specifies the effort level (default is optimize).

-max_transition
Considers maximum transition constraints during optimization.

-max_capacitance
Considers maximum capacitance constraints during optimization.

-crosstalk_noise
Considers crosstalk noise constraints during optimization.

-run_time_limit *number*
Specifies the run time limit in minutes (default is no limit).

-opt_delay_loop *number*
Specifies the total number of optimization loops (default is 10).

-opt_delay_search_repair_loop *number*
Specifies the number of search repair loops (default is 5).

DESCRIPTION

This command calls the timer to analyze the critical paths and then uses various

techniques to improve the design timing and minimize the crosstalk problems.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example optimizes the timing to 0.22 and only runs 3 opt loops.

```
prompt> route_rc_reduction\  
-slack_target 0.22 -opt_delay_loop 3
```

SEE ALSO

`set_si_options(2)`

route_search_repair

Performs a search and repair routing operation on the design.

SYNTAX

```
status route_search_repair
[-loop num]
[-run_time_limit num]
[-reset_width]
[-rerun_drc]
[-reset_min_max_layer]
[-trim_antenna_of_user_wires]
[-dont_connect_tie_off]
[-ignore_open_nets]
[-num_cpus num]
```

Data Types

num integer

ARGUMENTS

-loop *num*
Specifies the number of search and repair loops. The default is 50. The range is 0 to 1000.

-run_time_limit *num*
CPU runtime limit in minutes. The default is no limit. The range is -1 to 500.

-reset_width
Resets the default width.

-rerun_drc
Runs the design rule checker before starting search and repair.

-reset_min_max_layer
Resets the min-max layer setting.

-trim_antenna_of_user_wires
Trims wire segments of preroute wires.

-dont_connect_tie_off
Specifies not to connect pg or clock.

-ignore_open_nets
Specifies not to connect partially routed wire segments.

-num_cpus *num*
Specifies the number of cpus for distributed routing. It must be ≥ 1 . The range is 0 to 63.

DESCRIPTION

This command allows you to run incremental search and repair routing passes after detail routing. During these routing passes, the tool searches for design rule violations and reroutes wires in an effort to avoid violations.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example performs search and repair using the following options: **-loop**, **-run_time_limit**, **-reset_width**, **-rerun_drc**, **-reset_min_max_layer**, **-trim_antenna_of_user_wires**, **-dont_connect_tie_off**, **-ignore_open_nets**, and **-num_cpus**.

```
prompt> route_search_repair -loop 5\  
-run_time_limit 50 -reset_width\  
-rerun_drc -reset_min_max_layer  
-trim_antenna_of_user_wires\  
-dont_connect_tie_off\  
-ignore_open_nets -num_cpus 1
```

SEE ALSO

```
route_auto(2)  
route_detail(2)  
set_route_options(2)
```

route_spreadwires

Spreads wires in the current design to increase the average spacing between spacing between wires, which reduces critical area short faults and improves yield.

SYNTAX

```
status route_spreadwires
[-timing_driven [-setup_slack_threshold setup_time]]
[-search_repair_loop number_of_loops]
[-min_jog_length min_ratio]
[-num_cpus number_of_cpus]
```

Data Types

<i>setup_time</i>	float
<i>number_of_loops</i>	integer
<i>min_ratio</i>	integer
<i>number_of_cpus</i>	integer

ARGUMENTS

- timing_driven
Prevents wire spreading on critical nets. The tool determines the critical nets by identifying those nets whose setup slack is less than the setup slack threshold specified. By the default, the tool performs wire spreading on all nets.
- setup_slack_threshold *setup_time*
Specifies the setup slack threshold. By default, the value of the *setup_time* option is 0.0. The higher the threshold value, the more nets are set as critical. This option has an effect only when used with the **-timing_driving** option.
- search_repair_loop *number_of_loops*
Specifies the number of loops that search and repair performed after wire spreading. By default, the *number_of_loops* value is 10.
- min_jog_length *min_ratio*
Specifies the minimum value of the ratio of jog length to layer pitch. By default, the *min_ratio* value is 2.
- num_cpus *number_of_cpus*
Specifies the number of CPUs to use for distributed routing. The *number_of_cpus* value must be greater than or equal to 1. By default, the *number_of_cpus* value is 1.

DESCRIPTION

This command spreads wires in the current design to increase the average spacing between wires, which reduces critical area short faults and improves yield. When spreading wires, a piece of the original wire is broken and pushed away, and jog

wires are created at both ends to maintain a connection. After spreading the wires, the tool runs search and repair to fix any DRC violations introduced during wire spreading.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example performs wire spreading using the **search_repair_loop**, **timing_driven**, **setup_slack_threshold** options.

```
prompt> route_spreadwire -search_repair_loop 6 -timing_driven \  
                        -setup_slack_threshold 0.15
```

SEE ALSO

`route_search_repair(2)`

route_track

Assigns wires to tracks in the design.

SYNTAX

status **route_track**

DESCRIPTION

Before running the detail router, run **route_track** to specify which tracks within each global routing cell are to be used for each net. Track assignment operates on the entire design at once; it can make long routes straight and reduce the number of vias, whereas the detail router routes a small area at a time. After track assignment finishes, all nets are routed. There are many violations, particularly where the routing connects to pins. The detail router cleans up the violations.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following command performs track assignment:

```
prompt> route_track
```

SEE ALSO

set_route_options(2)
route_auto(2)

route_widen_wire

Performs wire widening for wires of signal nets.

SYNTAX

```
status route_widen_wire
[-search_repair_loop number_of_loops]
[-nonuniform_widening]
[-timing_driven]
[-setup_slack_threshold value_of_setup_slack_threshold]
[-hold_slack_threshold value_of_hold_slack_threshold]
```

Data Types

<i>number_of_loops</i>	integer
<i>value_of_setup_slack_threshold</i>	float
<i>value_of_hold_slack_threshold</i>	float

ARGUMENTS

- search_repair_loop *number_of_loops*
Specifies the number of search and repair loops for design rule checking (DRC) fixes after wires are widened ($0 \leq \text{number\_of\_loops} \leq 100$). By default, the value of *number_of_loops* is **10**.
- nonuniform_widening
Attempts to break each wire according to its neighbor wire patterns, and then tries to widen it with a different width for each piece of wire.
- timing_driven
Prevents widening of the wires of timing-driven nets. Widened wires impact both the setup and hold time of a net. Timing-critical nets should not be widened.
- setup_slack_threshold *value_of_setup_slack_threshold*
Sets the setup time threshold. Default value is 0.0. The higher the threshold, the more nets will be set as critical.
- hold_slack_threshold *value_of_hold_slack_threshold*
Sets the hold time threshold. Default value is 0.0. The higher the threshold, the more nets will be set as critical.

DESCRIPTION

This command attempts to widen the wires of all signal nets. The width of a wire is limited during the widening operation so that the wire does not exceed 1.5 times the default wire width or become a fat wire. A fat wire is one whose width exceeds the threshold of fat degree defined in the technology file.

The **-timing_driven** option freezes the nets whose setup time slack or hold time slack values are lower than their respective threshold values. Wires of frozen nets are

not widened. This can preserve timing results of timing-critical nets.

The **-setup_slack_threshold** and **-hold_slack_threshold** options must be used together with the **-timing_driven** option. These options provide the interface to adjust the range of timing-critical nets, which will be frozen from being widened.

After wire widening is done, the internal search and repair engine is triggered to fix DRC violations introduced by **route_widen_wire**.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example performs wire widening followed by 20 search and repair loops.

```
prompt> route_widen_wire -search_repair_loop 20
```

SEE ALSO

`route_spreadwires(2)`

route_zrt_auto

Performs global routing, track assignment, and detail routing in one step.

SYNTAX

```
status route_zrt_auto
[-max_detail_route_iterations num]
[-reuse_existing_global_route true | false]
[-stop_after_track_assignment true | false]
[-save_after_global_route true | false]
[-save_after_track_assignment true | false]
[-save_after_detail_route true | false]
[-save_cell_prefix name]
```

Data Types

<i>num</i>	integer
<i>name</i>	string

ARGUMENTS

-max_detail_route_iterations *num*
Specifies the maximum number of detail routing iterations. You can specify an integer between 1 and 1000.
By default, the maximum number of iterations for the detail routing command (40) is used.

-reuse_existing_global_route true | false
Enables or disables incremental global routing.
By default (false), the global router deletes all existing global routes from the design before routing starts.
If this option is set to true, the global router keeps and reuses existing global routes.

-stop_after_track_assignment true | false
Controls whether the command runs detail routing.
By default (false), the command runs global routing, track assignment, and detail routing.
If this option is true, the command skips detail routing and runs only global routing and track assignment.

-save_after_global_route true | false
Controls whether the design is saved after global routing.
The default value is false.

-save_after_track_assignment true | false
Controls whether the design is saved after track assignment.
The default value is false.

-save_after_detail_route true | false
Controls whether the design is saved after detail routing.
The default value is false.

`-save_cell_prefix name`

Specifies the prefix to use when saving the design. For example, if you specify "foo" as the prefix, the name of the design saved after global routing is foo_icGRt.

The default prefix is auto.

DESCRIPTION

The **route_zrt_auto** command runs global routing, track assignment, and detail routing in one step. This command runs faster than using the basic commands because it does not access the disk between the different stages. The results are comparable to the basic flow (it might not be identical because the shapes could be ordered differently when the design is saved to disk).

Multicorner-Multimode Support

This command uses information from active scenarios only.

EXAMPLES

The following command runs automatic routing using the **-save_after_global_route**, **-save_after_detail_route**, **-save_cell_prefix**, and **-max_detail_route_iterations** options:

```
prompt> route_zrt_auto \  
-save_after_global_route true \  
-save_after_detail_route true \  
-save_cell_prefix auto \  
-max_detail_route_iterations 15
```

SEE ALSO

route_zrt_global(2)
route_zrt_track(2)
route_zrt_detail(2)
set_route_zrt_common_options(2)
set_route_zrt_detail_options(2)
set_route_zrt_track_options(2)

route_zrt_clock_tree

Performs routing on the clock nets in the design using Zroute.

SYNTAX

```
status route_zrt_clock_tree  
[-utilize_dangling_wires false | true]  
[-reuse_existing_global_route false | true]  
[-max_detail_route_iterations int]
```

ARGUMENTS

There are no arguments to this command.

DESCRIPTION

The **route_zrt_clock_tree** command route the clock nets in the current design using Zroute.

Multicorner-Multimode Support

This command uses information from active scenarios only.

EXAMPLES

The following example shows the default command output.

```
prompt> route_zrt_clock_tree
```

SEE ALSO

```
set_route_mode_options(2)  
route_zrt_group(2)
```

route_zrt_detail

Performs detail routing on the design.

SYNTAX

```
status route_zrt_detail
[-max_number_iterations count]
[-coordinates {{llx1_lly1}_{urx1_ury1}_...}]
[-incremental true | false]
[-initial_drc_from_input true | false]
[-start_iteration int]
```

Data Types

<i>count</i>	integer
<i>int</i>	integer

ARGUMENTS

-max_number_iterations *count*
Specifies the maximum number of routing iterations. The detail router terminates after the maximum number of iterations or when it detects a lack of progress, whichever occurs first. Limiting the number of routing iterations can provide faster results; however, it might cause the detail router to terminate before the design rule constraint (DRC) violations are resolved. The range is from 1 to 1000. The default value is 40. To disable automatic termination of detail routing (and force Zroute to complete the maximum number of iterations), use the **set_route_zrt_detail_options -force_max_number_iterations true** command.

-coordinates {{*llx1_lly1*}_{*urx1_ury1*}_...}
Specifies the routing area. It supports multiple rectangle routing areas. The coordinates set should be in user unit. The -coordinates option is not set by default, in which case the full chip will be routed.

-incremental true | false
If true (default is false), the router performs incremental mode routing. By default the router starts from the iteration number that detail routing ends with last time the cell was routed. If -start_iteration is set, the router will start from the iteration number specified in this option. By default the router will re-check DRC in the beginning. If -initial_drc_from_input is set as true, the router will skip the initial DRC checking and start with the DRC information

stored in the cell.

`-initial_drc_from_input true | false`

When true, the detail router uses the DRC information in the cell as the initial DRC information. Use this option very carefully; set it to true only if you are absolutely sure that the the DRCs in the cell are up-to-date.
The default is false.

`-start_iteration int`

Specifies the start iteration for detail route. Detail route iterations will run from **start_iteration** number up to the iteration number specified by **max_number_iterations**.
The range is from 0 to 1000.
The default is 0.

DESCRIPTION

This command performs detail routing on a design. You run this command after running the **route_zrt_track** command.

By default, **route_zrt_detail** analyzes the design rule constraint (DRC) violations in the entire design, and then resolves those violations. To skip the analysis and use the violations stored in the database as the starting point, set the option **-initial_drc_from_input** to **true**. The user should do this only if the violations stored in the database are current and correct.

The behavior of this command is influenced by the options set by the **set_route_zrt_detail_options** and **set_route_zrt_common_options** commands.

Multicorner-Multimode Support

This command uses information from active scenarios only.

EXAMPLES

The following command performs detail routing:

```
prompt> route_zrt_detail
```

The following command performs detail routing with a maximum of 3 iterations:

```
prompt> route_zrt_detail -max_number_iterations 3
```

The following command performs detail routing in the rectangle areas [0 0 50.0 60.2] and [100.0 100.0 120.2 110.2]:

```
prompt> route_zrt_detail -  
coordinates {{0 0} {50.0 60.2} {100.0 100.0} {120.2 110.2}}
```

The following command performs incremental detail routing:

```
prompt> route_zrt_detail -incremental true
```

The following command uses drc information in the cell and starts with iteration 1:

```
prompt> route_zrt_detail -initial_drc_from_input true  
-start_iteration 1
```

SEE ALSO

```
set_route_zrt_detail_options(2)  
set_route_zrt_common_options(2)  
route_zrt_auto(2)  
route_zrt_global(2)  
route_zrt_track(2)
```


route_zrt_eco

Performs ECO routing on the design.

SYNTAX

```
status route_zrt_eco
[-max_detail_route_iterations count]
[-nets list]
[-open_net_driven true | false]
[-reroute modified_nets_only | modified_nets_first_then_others | any_nets]
[-reuse_existing_global_route true | false]
[-utilize_dangling_wires true | false]
```

Data Types

count integer
list list

ARGUMENTS

- max_detail_route_iterations** *count*
Specifies the maximum number of detail routing iterations. You can specify an integer between 1 and 1000.
By default, the maximum number of iterations for the detail routing command (40) is used.
- nets** *list*
Specifies the nets on which to perform ECO routing. The command first connects the open nets in this list, and then fixes design rule violations in their neighborhood. Since the violations are computed and fixed only in the neighborhood of the specified nets, the violations reported by the command are not accurate for the design. You need to run **verify_zrt_route** after this command to get the accurate picture.
When you specify the **-nets** option, the command always treats the **-open_net_driven** option as false, regardless of its actual setting.
By default, this command works on all nets in the design.
- open_net_driven** true | false
Controls whether ECO route fixes DRC violations for the whole design (default) or only in the neighborhood of the open nets.
By default (false), ECO route connects the open nets in the design and fixes design rule violations for the entire design.
When true, ECO route connects the open nets in the design and fixes DRC violations only in the neighborhood of the open nets. In this case, the user needs to run **verify_zrt_route** after this command to get an accurate picture of the violations in the design.
- reroute** modified_nets_only | modified_nets_first_then_others | any_nets
Controls which nets are rerouted.
The definition for each mode is
* **any_nets** (default)
The router reroutes any nets freely to fix all the DRC violations.

* **modified_nets_only**

The router freezes all the fully connected nets and tries to finish the routing by modifying only the nets with open ECO changes. This might cause the router to fail when fixing some violations. It is only suitable for very minor ECO changes.

* **modified_nets_first_then_others**

The router first freezes all the fully connected nets and tries to finish the routing by modifying only the nets with open ECO changes. When there are violations remaining, the router tries to reroute those fully connected nets to resolve the violations. This reduces the layout changes for minor ECO operations but is not suitable for major ECO changes.

`-reuse_existing_global_route true | false`

Enables or disables incremental global routing.

By default (false), the global router deletes all existing global routes from the design before routing starts.

If this option is set to true, the global router keeps and reuses existing global routes.

`-utilize_dangling_wires true | false`

Controls whether the router tries to reuse the dangling routes as much as possible to fix the opens.

The default value is true.

DESCRIPTION

The **route_zrt_eco** command performs ECO routing. It first connects the open nets and then fixes the DRC violations. The options control which nets are connected and on which areas of the chip the DRCs are fixed.

Multicorner-Multimode Support

This command uses information from active scenarios only.

EXAMPLES

The following command performs ECO routing using the **-nets**, **-utilize_dangling_wires**, and **-reroute** options.

```
prompt> route_zrt_eco -nets {n1 n2 n3} -utilize_dangling_wires true \  
-reroute modified_nets_first_then_others
```

SEE ALSO

`route_zrt_global(2)`
`route_zrt_track(2)`
`route_zrt_detail(2)`
`set_route_zrt_common_options(2)`
`set_route_zrt_detail_options(2)`

route_zrt_global

Performs global routing on the design.

SYNTAX

```
status route_zrt_global  
[-effort minimum | low | medium | high]  
[-congestion_map_only true | false]  
[-reuse_existing_global_route true | false]
```

ARGUMENTS

-effort minimum | low | medium | high
Specifies the effort used to perform global routing.
By default, the global router uses medium effort.

-congestion_map_only true | false
Creates the congestion map without creating glinks.
The default value is false.

-reuse_existing_global_route true | false
Enables or disables incremental global routing.
By default (false), the global router deletes all existing global routes from the design before routing starts.
If this option is set to true, the global router keeps and reuses existing global routes.

DESCRIPTION

The **route_zrt_global** command maps general pathways through the design for each unrouted net (signal nets and clock nets). The global router uses a three-dimensional array of global routing cells to model the demand and capacity of global routing. The average height of the standard cells is used to create the height and width of each global routing cell. During global routing, the tool assigns nets to the global routing cells through which they pass. For each global routing cell, routing capacity is calculated according to the blockages, pins, and routing tracks inside the cell. Although global routing does not assign the nets to the actual wire tracks, it notes the number of nets assigned to each global routing cell. The tool calculates the demand for wire tracks in each global routing cell and reports the overflows, which are the number of wire tracks still needed after the tool assigns nets to the available wire tracks in a global routing cell. The tool might reduce overflows by detouring nets around congested areas and increasing the wire length. When a wire is prerouted, the tool checks to see whether it is completely connected. If the net is completely connected, the tool treats the net as a blockage; if the net is partially connected, the tool connects all parts of the net.

- * The global router treats power and ground nets as blockages.
- * The global router treats prerouted meshes and trunks as a single connection. Make sure you examine the routing to see whether they are connected properly.

Multicorner-Multimode Support

This command uses information from active scenarios only.

EXAMPLES

The following command performs global routing using high effort:

```
prompt> route_zrt_global -effort high
```

SEE ALSO

```
set_route_zrt_global_options(2)  
route_zrt_auto(2)
```

route_zrt_group

Performs routing on a group of nets in the design.

SYNTAX

```
status route_zrt_group
[-max_detail_route_iterations num]
[-nets nets | -all_clock_nets]
[-stop_after_global_route true | false]
[-reuse_existing_global_route true | false]
[-utilize_dangling_wires true | false]
-from_file file_name
```

Data Types

<i>num</i>	integer
<i>nets</i>	collection

ARGUMENTS

-max_detail_route_iterations *num*

Specifies the maximum number of detail routing iterations. You can specify an integer between 1 and 1000.

By default, the maximum number of iterations for the detail routing command (40) is used.

-nets *nets*

Specifies the nets to be routed.

You must specify the nets on which to run this command by specifying at least one of the following options: **-nets**, **-all_clock_nets**, and **-from_file**. The **-nets** and **-all_clock_nets** options are mutually exclusive; you can specify only one. You can specify the **-from_file** option in addition to the **-nets** option.

-all_clock_nets

Routes all clock nets.

You must specify the nets on which to run this command by specifying at least one of the following options: **-nets**, **-all_clock_nets**, and **-from_file**. If you specify the **-all_clock_nets** option, you cannot specify the **-nets** or **-from_file** option.

-stop_after_global_route true | false

Controls whether the router stops after global route or continues all the way to detail route.

The default value is false which means it will continue all the way to detail route.

-reuse_existing_global_route true | false

Enables or disables incremental global routing.

By default (false), the global router deletes all existing global routes from the design before routing starts.

If this option is set to true, the global router keeps and reuses existing global routes.

`-utilize_dangling_wires true | false`

Controls whether the router tries to reuse the dangling routes as much as possible.

The default value is true.

`-from_file file_name`

Routes the nets specified by name in the file.

If the specified file either does not exist or does not contain any valid nets, the tool generates an error and indicates the source of the problem. You must specify the nets on which to run this command by specifying at least one of the following options: **-nets**, **-all_clock_nets**, and **-from_file**. You can specify the **-from_file** option in addition to the **-nets**; however, the **-from_file** option cannot be used with the **-all_clock_nets** option.

DESCRIPTION

The **route_zrt_group** command is used to route a specified group of nets.

Multicorner-Multimode Support

This command uses information from active scenarios only.

EXAMPLES

The following command performs routing for all clock nets using the **-max_detail_route_iterations** and **-utilize_dangling_wires** options.

```
prompt> route_zrt_group -all_clock_nets \  
-max_detail_route_iterations 6 \  
-utilize_dangling_wires true
```

SEE ALSO

`route_zrt_global(2)`

`route_zrt_track(2)`

`route_zrt_detail(2)`

route_zrt_track

Performs track assignment on the design.

SYNTAX

status **route_zrt_track**

ARGUMENTS

This command has no arguments.

DESCRIPTION

This command performs track assignment on a global routed design. Run this command after running **route_zrt_global** and before running **route_zrt_detail** to assign wires to the routing tracks defined in the technology file. Track assignment operates on the entire design at once; whereas the detail router routes a small area at a time. Track assignment can make long routes straight and reduce the number of vias. After track assignment finishes, all nets are routed, but there are many violations, particularly where the routing connects to pins. The detail router fixes the violations.

The behavior of this command is influenced by options set by the **set_route_zrt_track_options** and **set_route_zrt_common_options** commands.

Multicorner-Multimode Support

This command uses information from active scenarios only.

EXAMPLES

The following command performs track assignment.

```
prompt> route_zrt_track
```

SEE ALSO

```
set_route_zrt_track_options(2)
set_route_zrt_common_options(2)
route_zrt_auto(2)
route_zrt_global(2)
route_zrt_detail(2)
```

route_zrt_with_route_guidance

Fixes complex DRC violations or lithography hotspots based on the route guidance specified in a route guidance file.

SYNTAX

```
status route_zrt_with_route_guidance  
[-max_detail_route_iterations count]
```

Data Types

count integer

ARGUMENTS

-max_detail_route_iterations *count*
Specifies the maximum number of detail routing iterations to run after making the changes specified in the route guidance file. This is mainly for cleaning up any DRC violations generated during fixing. The default value is 5.

DESCRIPTION

The **route_zrt_with_route_guidance** command fixes complex DRC violations or lithography hotspots based on the route guidance specified in a route guidance file. Before you run this command, the design should have little or no DRC violations; otherwise, there is no guarantee of DRC convergence.

You must run the **read_zrt_route_guidance** command to import a route guidance file before running this command, as the violation and route guidance information comes directly from this file.

By default, the router runs five detail routing iterations after fixing the violations from the route guidance file to fix any DRC violations that might have been introduced. You can change the number of detail routing iterations by using the **-max_detail_route_iterations** option.

EXAMPLES

The following commands import DRC violation and lithography hotspot information from the out00.xml route guidance file and use the route guidance contained in that file to fix the violations. The router runs three detail routing iterations after fixing the violations.

```
prompt> read_zrt_route_guidance out00.xml  
prompt> route_zrt_with_route_guidance -max_detail_route_iterations 3
```

SEE ALSO

read_zrt_route_guidance(2)

rp_group_inclusions

Returns a collection of relative placement groups that are directly included in the specified relative placement groups.

SYNTAX

```
collection rp_group_inclusions  
[rp_groups]
```

Data Types

rp_groups list or collection

ARGUMENTS

rp_groups
Specifies the relative placement groups in which to search for directly included relative placement groups.
If you do not specify this argument, the tool searches all relative placement groups for directly included relative placement groups.

DESCRIPTION

The **rp_group_inclusions** command returns a collection containing all relative placement groups that are directly included in the relative placement groups specified in *rp_groups*. If you do not specify *rp_groups*, the command returns a collection containing all included relative placement groups. This command is supported only for designs that do not contain multiply-instantiated designs.

A group directly includes another group if the group was added by using the **add_to_rp_group -hierarchy** command without also using the **-instance** option.

If no groups are found, an empty string is returned.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses the **rp_group_inclusions** command:

```
prompt> get_rp_groups  
{b::top a0::a0_group b::u_top/a1_group  
  b::u_top/a1_group_include b::u_nxt/a1_group  
  b::a1_group_include a1::a1_group}  
  
prompt> rp_group_inclusions
```

```
{b::u_top/a1_group b::u_nxt/a1_group}

prompt> rp_group_inclusions a0::a0_group

prompt> rp_group_inclusions top
{b::u_top/a1_group b::u_nxt/a1_group}

prompt> remove_rp_groups -all -quiet
1

prompt> rp_group_inclusions
```

SEE ALSO

```
add_to_rp_group(2)
all_rp_groups(2)
all_rp_hierarchicals(2)
all_rp_inclusions(2)
all_rp_instantiations(2)
all_rp_references(2)
create_rp_group(2)
remove_rp_groups(2)
rp_group_instantiations(2)
rp_group_references(2)
```

rp_group_instantiations

Returns a collection of relative placement groups that are instantiated in any of the specified relative placement groups.

SYNTAX

```
collection rp_group_instantiations  
[rp_groups]
```

Data Types

rp_groups list or collection

ARGUMENTS

rp_groups
Specifies the relative placement groups in which to search for instantiated relative placement groups.
If you do not specify this argument, the tool searches all relative placement groups for instantiated relative placement groups.

DESCRIPTION

The **rp_group_instantiations** command returns a collection containing all relative placement groups that are instantiated in the relative placement groups specified in *rp_groups*. If you do not specify *rp_groups*, the command returns a collection containing all instantiated relative placement groups.

This command is supported only for designs that do not contain multiply-instantiated designs.

A group instantiates another group if the group was added by using the **add_to_rp_group -hierarchy -instance** command.

If no groups are found, an empty string is returned.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses the **rp_group_instantiations** command:

```
prompt> get_rp_groups  
{b::top a0::a0_group b::u_top/a1_group  
  b::u_top/a1_group_include b::u_nxt/a1_group  
  b::a1_group_include a1::a1_group}
```

```
prompt> rp_group_instantiations
{a0::a0_group}

prompt> rp_group_instantiations b::a1_group_include
{a1::a1_group}

prompt> remove_rp_groups -all -quiet
1

prompt> rp_group_instantiations
```

SEE ALSO

```
add_to_rp_group(2)
all_rp_groups(2)
all_rp_hierarchicals(2)
all_rp_inclusions(2)
all_rp_instantiations(2)
all_rp_references(2)
create_rp_group(2)
remove_rp_groups(2)
rp_group_inclusions(2)
rp_group_references(2)
```

rp_group_references

Returns a collection of cells that are directly included by the specified relative placement groups.

SYNTAX

```
collection rp_group_references  
[rp_groups]  
[-leaf | -instance]
```

Data Types

rp_groups list or collection

ARGUMENTS

rp_groups
Specifies the relative placement groups in which to find directly included cells.

-leaf | -instance
Specifies whether only leaf cells or only hierarchical cells are returned. By default, both leaf and hierarchical cells are returned.

DESCRIPTION

The **rp_group_references** command returns a collection containing all cells that are directly included in the relative placement groups specified in *rp_groups*. If you do not specify *rp_groups*, the command returns a collection containing all cells that are directly included in any relative placement group.

This command is supported only for designs that do not contain multiply-instantiated designs.

When you specify the **-leaf** option, only leaf cells are returned. When you specify the **-instance** option, only hierarchical cells are returned. An included relative placement group, unlike an instantiated relative placement group, does not have a hierarchical cell associated with it, so the **rp_group_references** command does not report included hierarchical relative placement groups.

A group directly includes a leaf cell if the cell was added to that group using the **add_to_rp_group -leaf** command.

A group directly includes a hierarchical cell if the cell was used to hierarchically instantiate another group using the **add_to_rp_group -instance** command.

If no cells are found, an empty string is returned.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses the **rp_group_references** command:

```
prompt> get_rp_groups
{b::top a0::a0_group b::u_top/a1_group
  b::u_top/a1_group_include b::u_nxt/a1_group
  b::a1_group_include a1::a1_group}

prompt> rp_group_references
{u_mid u_bot}

prompt> all_rp_references u_0 -design a0
{a0::a0_group}

prompt> rp_group_references a0::a0_group
{u_bot/u_1 u_bot/u_0}

prompt> remove_rp_groups -all -quiet
1

prompt> rp_group_references
```

SEE ALSO

```
add_to_rp_group(2)
all_rp_groups(2)
all_rp_hierarchicals(2)
all_rp_inclusions(2)
all_rp_instantiations(2)
all_rp_references(2)
create_rp_group(2)
remove_rp_groups(2)
rp_group_inclusions(2)
rp_group_instantiations(2)
```

run_parallel_jobs

Runs parallel jobs controlled by the tool. It is typically used to invoke the tool on a group of subblocks in a hierarchical design.

SYNTAX for Submitting a Job

status **run_parallel_jobs**

```
-batch_script script_path
-exec exec_path
[-parallel num_jobs]
[-monitor]
[-job_title job_title]
[-stage_names stage_name_list]]
[-submit_timeout timeout]
[-run_timeout timeout
 | -run_timeout_factor factor]
[-queue queue_name]
[-var_list name_value_pairs]
[-temp_dir directory]
[-setup_for_run_only]
[-job_script fsubmission_command_path]
[-verbose]
[-kill]
[-check]
job_name_list
```

Data Types for Submitting a Job

<i>script_path</i>	string
<i>exec_path</i>	string
<i>num_jobs</i>	int or "all"
<i>job_title</i>	string
<i>stage_name_list</i>	string list
<i>timeout</i>	time
<i>factor</i>	int
<i>queue_name</i>	string
<i>name_value_pairs</i>	string list
<i>job_name_list</i>	job_name list

ARGUMENTS

-batch_script *script_path*

Specifies the path to the batch script which is the script that is provided to the child jobs. The batch script can contain any commands and is passed to the executable on each parallel job by using **-f** *script_path*.

-exec *exec_path*

Specifies the path to the executable that is run in parallel. The variables that are available to the executable are passed through the UNIX environment defined in the **-var_list** option. The batch script defined in the **-batch_script**

option, is passed to the executable on the command line by using **-f** *script_path*.

-parallel *num_jobs*

Controls the level of parallelism used for job submission.

It is used to restrict the number of active jobs to a specified number. By default all jobs are launched in parallel.

-monitor

Opens a GUI panel that shows the job status for the running jobs and enables interactive job control. This option requires that the GUI is already running. With this monitor you have the ability to check jobs and also to kill all active jobs. This ability is only available via the monitor.

-job_title *job_title*

Specifies the title of the job set. This title is used for the job set within the queuing system and for the title bar if you open the GUI.

-stage_names *stage_name_list*

Specifies the stage names for which status information is returned from the child processes. The stages are displayed in the GUI monitor. If no stages are defined, a single stage called "Job Status" is defined for each running job and is used to collect updated status information.

-submit_timeout *timeout*

Specifies the timeout value in the following format: [[HH:]MM:]SS[.SS]. For example, if you specify **-submit_timeout 30**, this means 30 seconds. If you specify **-submit_timeout 2:30**, this means 2 minutes and 30 seconds. If you specify hours, the maximum number of minutes is 59; otherwise, you can specify any number of minutes. If you specify minutes, the maximum number of seconds is 59; otherwise, you can specify any number of seconds.

If the submitted jobs do not become running jobs within the specified timeout, they are killed. This is to prevent hanging jobs in the queuing system.

If you do not specify this option, the system waits as long as needed for jobs to start.

-run_timeout *timeout*

Specifies the maximum amount of elapsed time that a job can run in the following format: [[HH:]MM:]SS[.SS]. For example, if you specify **-run_timeout 30**, this means 30 seconds. If you specify **-run_timeout 2:30**, this means 2 minutes and 30 seconds. If you specify hours, the maximum number of minutes is 59; otherwise, you can specify any number of minutes. If you specify minutes, the maximum number of seconds is 59; otherwise, you can specify any number of seconds.

If the elapsed time exceeds this amount, the job is killed. You can use this option to prevent stuck or runaway jobs that will prevent a flow from running to completion.

This option and the **-run_timeout_factor** option are mutually exclusive; you can specify only one. If you specify neither, there is no time limit.

-run_timeout_factor *factor*

Specifies a factor used to determine the maximum elapsed time that a job can run. The factor value must be greater than one. The factor relates to the average elapsed time of completed jobs. When one or more jobs have finished successfully, an average elapsed time of completion is computed. When a job's

elapsed time rises above this average times the *factor*, the job is killed. The expectation is that in most cases, the parallel jobs are arranged so that they run for approximately the same time. By setting a *factor* in the range of 5-10, you can effectively kill runaway jobs without fear of killing a long-running job that will ultimately succeed.

This option and the **-run_timeout** option are mutually exclusive; you can specify only one. If you specify neither, there is no time limit.

-queue *queue_name*

Specifies the queue name in which to run the jobs. This name has meaning only to a given installation using the job script specified by the **-job_script** option.

In the default job script (which ties to LSF), the default LSF queue will be used.

-var_list *name_value_pairs*

Communicates information from the parent to the child processes. The values in this list are specified as pairs of the form *name=value*. The *name* values become UNIX environment variables that are passed through the queuing system to the child jobs.

In addition to the variables and values specified in this list, there are several predefined variables managed by the command. The *HostName* and *PortNum* variables are set to the parent host and the number of the port on which the parent will be listening for child status information, which is communicated back to the parent through the **send_flow_status** command. The *JobName* variable contains the name of the job and is available to the child processes. You must use this option to pass through any additional data that is needed by the batch script given to the executable.

-temp_dir *directory*

This option sets up the working directory for the command. By default the working directory will be *./sub_blocks*.

-setup_for_run_only

This option causes the command to set up all the scripts for the parallel run (in *temp_dir*), but not actually do the parallel run. This is useful for looking at the scripts that will be run to see that they are correct. Also you can use the scripts for running jobs locally if needed for debugging.

-job_script *fsubmission_command_path*

Specifies the program and script to execute for job submission. This typically involves an executable like */usr/bin/perl* plus a job submission script written in perl.

The default job submission script is located at *\$root/auxx/syn/chipoft/job_run.pl*. This default job submission script interfaces to LSF and provides a set of capabilities that is sufficient for many, if not most, users. You can also construct your own job submission script. The script can be in any language, as long as it is directly executable by the system. Using this mechanism, you can interface to any job-control system, proprietary or commercial.

The job submission script must use the following syntax:

job_id_list your_script_here

-submit

[-queue *queue_name*]

[-wait]

```
[-verbose]
[-var_list name_value_pairs]
-script script_path
-exec exec_path
-temp_dir temp_dir
job_name_list
```

With the exception of the return value, `job_id_list`, the definitions for the options and arguments is the same as for the **run_parallel_jobs** command. The `job_id_list` is a single string listing the submitted jobs in the form "JOB_NAME_1:JOB_ID_1 JOB_NAME_2:JOB_ID_2 ...", where the JOB_NAME values come from the `job_name_list`, and the JOB_ID values come from the queuing system. The JOB_ID values are unique values returned as the result of the job submission. These values are used to identify jobs when using the kill or check features of the GUI monitor (see **-monitor** option)

-verbose

Prints additional status information. This option is usually used while debugging the logistics of running a new flow.

-kill

Kills the specified jobs that are still running. Because the batch version of **run_parallel_jobs** blocks until the children are finished, this option can be used only when the the command is run with GUI monitor.

-check

Checks the specified jobs that are still running. By default, the **run_parallel_jobs** command checks on the running children once a minute. You can use this option to accelerate this checking. Because the batch version of **run_parallel_jobs** blocks until the children are finished, this option can be used only when the command is run with the GUI monitor.

`job_name_list`

Specifies a list of names for the jobs. These names have meaning only when taken together with the batch script and executable that are launched in parallel. For many cases, they are the names of subblocks in a hierarchical design, but they can refer to any task that can benefit from parallel execution.

If dependencies among jobs exist, you can specify them by using a special syntax in the `job_name_list` argument. For each job, you can specify a comma-separated list of jobs that must complete before this job is launched. A semicolon separates the job name from its associated dependency list. For example, to run jobs A and B in parallel and then run job C only when both jobs A and B have successfully completed, enter the following command: `is: prompt> run_parallel_jobs {A B C:A,B}` In this example, if either job A or B fails, job C is skipped.

If you do not specify any dependencies, none are assumed and the jobs run in parallel up to the level of parallelism currently in effect.

DESCRIPTION

This command allow you to submit jobs to an Load Sharing Facility (LSF). To submit a job, you must specify at least a **-batch_script**, which is the script that is provided to the child jobs; an **-exec**, which is the path to the executable that is run for the child jobs; and a list of named jobs. In this case, a job is launched for each named

job of the specified executable using the specified batch script. The status is collected about the progress of each job, and the **run_parallel_jobs** command completes after the final job has finished.

From this basic syntax, there are many increasing levels of complexity. You can invoke a GUI monitor and define a set of stages for the child flows. The GUI visually records the progress of the child jobs through the stages from status information communicated back from the children. You can add a job name to this run, which labels the information and keeps it separate from other jobs that might be run in the same parent session. You can set up a series of timeouts that prevent the command from hanging in the event that the queuing system hangs or if one or more of the child jobs hangs or is stuck in a single stage. You can provide a customized job script to control the interface to the queuing system. You can pass to the child jobs an arbitrary set of variable data values that can customize the behavior of the batch script.

By design, this command is independent of the batch script or the child-job executable. The definitions of stages, definitions of variables, the contents of the batch script, the choice of executable, and the names for the jobs are entirely up to you. You decide what the meaning of the pieces is based upon your own conventions and needs. This makes **run_parallel_jobs** a general-purpose command that can be used anywhere a set of tasks can be run in parallel.

EXAMPLES

This example launches jobs to implement subblocks in a hierarchical design:

```
prompt> run_parallel_jobs -batch_script block_flow.tcl -exec icc \  
-monitor -stage_names "Init PlaceOpt ClockOpt RouteOpt ModelCreation" \  
-var_list MWLibName=orca.mw {BLENDER_0 BLENDER_1 BLENDER_2 BLENDER_3 BLENDER_4}
```

This example sets a variable MWLibName to orca.mw. This variable gets passed to the five jobs for the blocks BLENDER_0 through BLENDER_4 to the icc executable using the batch flow script block_flow.tcl. This script includes **send_flow_status** commands to send status back to the parent process. Because the example uses the **-monitor** option, there is a GUI display of the status of each stage for each of the five jobs while they are executing. After the last job has finished, the command returns. It returns a true status if all jobs have succeeded and a false status if one or more of the child jobs have failed.

SEE ALSO

[send_flow_status\(2\)](#)

run_signoff

Performs sign-off analysis by invoking PrimeTime and StarRC.

SYNTAX

```
status run_signoff
[-check_only]
[-star_only]
[-pt_only]
[-incremental]
[-signoff_analysis true | false | sleep | wakeup]
[-aocvm]
[-path_based_analysis]
[-variation]
[-ignore_library_mismatch]
[-keep_license]
[-snapshot base_name]
[-correlation list]
```

ARGUMENTS

- check_only**
Performs consistency checking only without actually invoking StarRC or PrimeTime.
- star_only**
Invokes only StarRC during sign-off analysis and continues to use IC Compiler for other components, such as using the IC Compiler timer, not PrimeTime, for timing analysis.
- pt_only**
Invokes only PrimeTime during sign-off analysis and continues to use IC Compiler for other components, such as using IC Compiler extraction, not StarRC, for extraction.
- incremental**
Performs incremental sign-off analysis instead of full signoff analysis. To perform incremental signoff analysis, a full sign-off analysis must have been done already.
- signoff_analysis true | false | sleep | wakeup**
Enables or disables sign-off analysis mode with true|false. The default is true. If you are in sign-off mode, use **sleep** to enter sleep mode by releasing all PrimeTime processes. If you are in sleep mode, use **wakeup** to reacquire all PrimeTime processes.
- aocvm**
Indicates that the timing paths are to be adjusted using advanced on-chip variation (OCV) modeling information. Advanced OCV modeling computes the effects of OCV on timing by using variable derating factors that consider the location and the logic depth of each path being analyzed. This approach reduces pessimism compared to global derating, thereby minimizing unnecessary

overdesign.

-path_based_analysis

Enables the path-based analysis capability in PrimeTime.

-variation

Indicates that StarRC VX and PrimeTime VX are invoked to perform variation-aware analysis.

-ignore_library_mismatch

Ignores library mismatches between IC Compiler and PrimeTime saved sessions.

-keep_license

Retains IC Compiler licenses while running the sign-off tools.

-snapshot *base_name*

Enables snapshots. When snapshots are enabled, the **run_signoff** command saves the design after each invocation. The saved design is named by using the specified *base_name* appended with an index. The **run_signoff** command also saves a session from PrimeTime. This saved session is also named by using the specified *base_name* appended with an index.

-correlation *list*

Performs correlation checking between IC Compiler and StarRC or between IC Compiler and PrimeTime. You can specify one or more of the following keywords in the list:

- **setup**

Checks the IC Compiler, PrimeTime, and StarRC settings and reports any inconsistencies. It also generates a script called ICC.tcl which can be used to correct the inconsistent settings. For a multicorner-multimode design, this setting generates reports and scripts for each scenario. Before running the sign-off-driven design closure flow, you should exit sign-off mode by running the **run_signoff -signoff_analysis false** command and source the generated script.

- **spef_out**

Generates two parasitic files in SPEF format, one from IC Compiler per corner and one from StarRC per scenario. A sample script called compare_spef.csh is also generated. This script contains a utility to compare the SPEF files and check the parasitic correlation between IC Compiler and StarRC. You should check the sample script and modify it to match the design technology. All files generated during correlation are saved in a directory named *correlation_report.time_stamp*, such as *correlation_report.02_18_20_31_24*. This report directory is created automatically in each correlation run.

DESCRIPTION

The **run_signoff** command performs sign-off analysis by invoking sign-off tools: PrimeTime and StarRC. StarRC provides sign-off quality parasitic extraction and PrimeTime provides sign-off quality timing and signal integrity analysis. You can use the sign-off costing from these tools to perform optimization on the design. Use the **set_primetime_options** and **set_starrcxt_options** commands to configure the **run_signoff** command.

To enable signal integrity analysis, use the **set_si_options** command.

The **run_signoff** command does not perform optimization. Use the **signoff_opt** command to perform sign-off costing-based optimization.

Multicorner-Multimode Support

This command uses information from active scenarios only.

EXAMPLES

The following example runs the **run_signoff** command. The **report_timing** command run after the **run_signoff** command reports PrimeTime timing based on StarRC extraction.

```
prompt> open_mw_cel -lib my_lib my_cel
prompt> set_primetime_options -exec_dir /PrimeTime/linux/syn/bin
prompt> set_starrcxt_options -exec_dir /StarRCXT/linux/bin \
-max_nxtgrd_file /design/max.nxtgrd -map_file /design/map_file
prompt> run_signoff
prompt> report_timing
```

The following example runs the **run_signoff** command with the **-correlation** option. A correlation report and a script called ICC.tcl are generated.

```
prompt> run_signoff -correlation {setup}
```

SEE ALSO

```
signoff_opt(2)
set_primetime_options(2)
set_starrcxt_options(2)
report_primetime_options(2)
report_starrcxt_options(2)
set_si_options(2)
```

save_design_settings

Saves the design settings.

SYNTAX

```
status save_design_settings  
-library  
[-input file]
```

Data Types

file string

ARGUMENTS

-library
 Saves the design settings into the current library.

-input *file*
 Saves the design settings found in the given input file instead of the current session's non-default valued application variables.

DESCRIPTION

By default, this command saves only the session's application variables that have been changed from their default values, like the default behavior of the **write_app_var** command. However, if an input file is specified with the **-input** option, the contents of this input file are saved instead.

In this release, design settings can be saved only to the current library. If design settings already exist in the current library, the new design settings will overwrite the existing ones. Saving the design settings does not apply the settings to the current session. The saved design settings are applied only when the **restore_design_settings** command is used.

EXAMPLES

The following example saves the design settings into the current library.

```
prompt> save_design_settings -library  
1
```

SEE ALSO

```
write_app_var(2)  
write_design_settings(2)  
restore_design_settings(2)
```

save_mw_cel

Saves the specified design in Milkyway format.

SYNTAX

```
status save_mw_cel
[-as name]
[-increase_version]
[-scenarios scenario_list]
[-hierarchy]
[-previous]
[-check_only]
design_name
[-overwrite]
```

Data Types

<i>name</i>	string
<i>scenario_list</i>	list
<i>design_name</i>	string

ARGUMENTS

-as *name*

Saves the specified design as a Milkyway design with the specified name. This can be a brand new Milkyway design or an existing Milkyway design. If it is an existing Milkyway design, it can be similar to the name of the design to be saved and the data will be saved to the same design, or else it must not open yet.

The name can be formatted as A[.B], in which the string A represents the base name of the design and the string B represents the view name of the design. Currently, 4 kinds of views are supported: CEL, FRAM, err, and FILL. You can not specify the mw_cel of other views through the option. Please refer to the man page of mw_cel_collection for how to specify a valid Milkyway design name. save_mw_cel runs database consistency checker internally. If it fails, _CORRUPT will be appended to the Milkyway design name. For example. If you specify the name as "abc.CEL" and CEL-checker fails before saving the design, the Milkyway design name is "abc_CORRUPT" and is created in the CEL view.

-increase_version

Increases the version of the Milkyway design to be saved. If this option is not specified, it will overwrite current version of specified Milkyway design. If used with **-as** option, it means to increase the version of Milkyway design specified in as option.

If database consistency checker fails, _CORRUPT will be appended to the Milkyway design name.

-scenarios *scenario_list*

Specifies a list of scenarios whose constraints are saved with the Milkyway design. By default, the command includes all scenarios.

-hierarchy

Saves the modified top-level Milkyway design, its modified subdesigns, and all attached files. The saved data will overwrite the current version of the designs.

The subdesigns must be in the same design library as the top-level Milkyway design. Subdesigns (modified or unmodified) that are instantiated from the reference libraries are not touched.

Note: This option can not be used with the **-increase_version** option.

-previous

If current cell's data model version is the latest, convert it back to previous version and save. If the cell has been modified after opening, the user must use "save_mw_cel" to save the cell to disk first, then use "save_mw_cel -previous -as" to save it to the previous version. Otherwise, "save_mw_cel -previous -as" will not be allowed. After saving, current cell will be closed. If the user wants to continue operating on the cell, the user must re-open it. This option must be used together with **-as** or **-increase_version** option to avoid overwriting.

-check_only

Prints the list of Milkyway designs to be saved, but does not execute the save operation. If you use this option, you must also use the **-hierarchy** option.

design_name

Specifies the Milkyway design to be saved. By default, the command saves the current Milkyway design. If database consistency checker fails, **_CORRUPT** will be appended to the Milkyway design name.

DESCRIPTION

This command saves the specified design in Milkyway format. The open status of all Milkyway designs in current session are not affected.

When use the **-as** option, this command saves the specified design as a new design and copies the attach files for the new design. The associated views will be copied in overwrite mode, when the view name of the saved design is **CEL** and the character . (period) is not specified in the name of the new design.

The **-increase_version** option increases the version of the design to be saved as and copies all the attach files. Combine the **-as** option with **-increase_version**, the command increases the versions of the associated views, when the view name of the saved design is **CEL** and the character . (period) is not specified in the name of the design to be saved as.

Database consistency checker will be run before saving the Milkyway design. If it fails, **_CORRUPT** will be appended to the Milkyway design name.

To save the Milkyway design hierarchically, use the **-hierarchy** option. When the command finishes saving the Milkyway design, it prints the names of the saved Milkyway designs.

The command saves the top-level Milkyway design or subdesigns when they have been modified. When the top-level Milkyway design is un-modified while subdesigns are

modified, the commands saves the modified subdesigns and updates the time stamp of the top-level Milkyway design.

The **-hierarchy** option does not close any Milkyway designs. Thus, any open Milkyway designs remain open after the hierarchical save. If you do another hierarchical save, the whole process is applied again for the specified top-level Milkyway design.

The **-previous** option can only convert current cell from the latest schema version to the previous one, it cannot convert cell with older schema version. And if current cell has child referenced cells, this option cannot convert these child cells, instead it will error out.

When you use the **-check_only** option together with the **-hierarchy** option, the command prints the list of Milkyway designs to be saved, but does not save the Milkyway designs.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example saves the current design as a Milkyway design named *new*.

```
prompt> save_mw_cel -as new
Information: Performing CEL netlist consistency check. (MWDC-118)
Information: CEL consistency check PASSED. (MWDC-119)
Information: Saved design named new. (UIG-5)
1
```

The following example saves the current design as a Milkyway design named *new_CORRUPT*.

```
prompt> save_mw_cel -as new
Information: Performing CEL netlist consistency check. (MWDC-118)
Error: Cell UPC_BLK (0xa0b) reachable from parent hierarchy alu (0xa01) multiple times. (MWDC-131)
Error: CEL consistency check FAILED (MWDC-100)
Information: Saved design named new_CORRUPT . (UIG-5)
1
```

The following example saves the *ORCA* design and the designs in its hierarchy.

```
prompt> save_mw_cel -hierarchy ORCA
Starting hierarchical save ...
Saving the cell: BLENDER_4.CEL;1 ...
Saving the cell: BLENDER_1.CEL;1 ...
Saving the cell: BLENDER_3.CEL;1 ...
Saving the cell: BLENDER_0.CEL;1 ...
```

```
Saving the cell: ORCA.CEL;1 ...
Information: Saved design named ORCA.CEL;1. (UIG-5)
1
```

The following example checks and finds the sub-design *BLENDER_0* is modified, then saves the *BLENDER_0* design and its top-level design *ORCA*.

```
prompt> save_mw_cel -hierarchy -check
Starting hierarchical save with check only enabled ...
check only -- will save the cell: BLENDER_0.CEL;1 ...
Information: Saved design named ORCA.CEL;1. (UIG-5)
1
prompt> save_mw_cel -hierarchy
Starting hierarchical save ...
Saving the cell: BLENDER_0.CEL;1 ...
Saving the cell: ORCA.CEL;1 ...
Information: Saved design named ORCA.CEL;1. (UIG-5)
1
```

SEE ALSO

```
close_mw_cel(2)
current_mw_cel(2)
open_mw_cel(2)
mw_cel_collection(2)
```

save_qtm_model

Saves the current Quick Timing Model (QTM) description.

SYNTAX

string **save_qtm_model**

ARGUMENTS

The **save_qtm_model** command has no arguments.

DESCRIPTION

This command creates an in-memory DB timing model for the QTM model and indicates to ICC Design Planning that the model being defined is completed. All QTM commands must be placed between a **create_qtm_model** and a **save_qtm_model** command.

This command creates an in-memory Synopsys .db representation to be used by the ICC Timer. This command runs faster than **write_qtm_model** and doesn't require disk I/O. This command may be called multiple times in an ICC session, if the user tweaks the QTM model.

To display information about the current QTM model, use the **report_qtm_model** command.

For a basic description about QTM, see the **create_qtm_model** manual page. For a more detailed description about QTM, see the *ICC Design Planning User Guide*.

The command line options "*-format format_list*", "*-output file_name*" and "*-library_cell*" for **save_qtm_model** command are redundant and maintained for backward compatibility with JupiterXT and PrimeTime.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example creates in-memory DB timing model for QTM model.

```
prompt> save_qtm_model -format {db lib}
```

The following example creates in-memory DB timing model for QTM model.

```
prompt> save_qtm_model
```

SEE ALSO

`create_qtm_model(2)`
`report_qtm_model(2)`

save_upf

Writes out the UPF commands in the specified file. This command is supported only in UPF mode.

SYNTAX

```
collection save_upf  
upf_file_name
```

Data Types

```
upf_file_name      string
```

ARGUMENTS

```
upf_file_name  
    Specifies the name of the file to which UPF commands are to be written out.
```

DESCRIPTION

This command writes out the UPF that is currently part of the design to the specified file. This UPF is basically what you have loaded by using the **load_upf** command or the command line and the changes made to it during optimization and UPF commands specified at the command prompt in that session. For example if the optimization has changed the name of the cell that drives and isolation signal, it will modify the set_isolation_control to reflect the new isolation_signal name.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

```
prompt> save_upf
```

SEE ALSO

select_mim_master_instance

SYNTAX

```
status select_mim_master_instance  
[-clear]  
object
```

Data Types

object collection of plan groups or soft macro instances

ARGUMENTS

object Collection of plan groups and/or soft macros to identify as the "master" plan group or soft macro among MIM groups of plan groups and/or soft macros.

DESCRIPTION

This command specifies a plan group or a soft macro instance as the "master" plan group or soft macro among a set of plan groups, soft macros or black boxes that belong to a group of multiply-instantiated modules. Only one instance from among a MIM group of instances (plan group/soft macro/black box) can have this "master" instance property. This "master" instance represents a "physical" master instance for the purposes of the **commit_fp_plan_group** and **place_fp_pins** commands, which use a particular physical "master" to make certain decisions about commit and pin placement.

For the **commit_fp_plan_groups** command, when a MIM group of plan groups, soft macros or black boxes exists, the **commit_fp_plan_groups** command must identify one plan group among this MIM group of block instances as the "master" block instance. You can then specify which plan group among a MIM group is the "master" plan group prior to commit using the **select_mim_master_instance** command. This command allows you to specify which plan group within a MIM group that the **commit_fp_plan_groups** command should treat as the physical "master" plan group.

After **commit_fp_plan_groups** has committed the physical master plan group or black box, then by default, the cell instance that results from the commit process will inherit the physical "master" property. After this, any other plan groups or black boxes which are part of the same MIM group that are subsequently committed will not generate new .CEL views but will have instances created only in the top cell. When there are more than 2 soft macro or plan group instances of a MIM group which exist after commit, then this command can be used to transfer the physical "master" property to one of the other cell instances for the purpose of impacting how pin placement is performed.

The -clear option can be used to remove the physical "master" property from the possessing property.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example specifies a particular plan group as the "master" of its MIM group:

```
prompt> select_mim_master_instance [get_cells {planGroupA}]
```

```
prompt> select_mim_master_instance -clear [get_cells smB]
```

SEE ALSO

```
uniquify_fp_mw_cel(2)  
commit_fp_plan_groups(2)
```


send_flow_status

Sends flow status from a running job to its parent. It is used exclusively with the **run_parallel_jobs** command.

SYNTAX

```
status send_flow_status
[-job_name job_name]
[-host host_name]
[-port port_number]
-stage_name stage_name
-status current_status
[-eof]
[-verbose]
```

Data Types

<i>job_name</i>	string
<i>host_name</i>	string
<i>port_number</i>	int
<i>stage_name</i>	string
<i>current_status</i>	one of Running, Succeeded, or Failed

ARGUMENTS

- job_name *job_name***
Specifies the name for the current job.
If you do not specify this option, the command checks the JobName UNIX environment variable. If neither is set, the command fails.
- host *host_name***
Specifies the host name for the parent of the current job.
If you do not specify this option, the command checks the HostName UNIX environment variable. If neither is set, the command fails.
- port *port_number***
Specifies the port number for the parent of the current job.
If you do not specify this option, the command checks the PortNum UNIX environment variable. If neither is set, the command fails.
- stage_name *stage_name***
Specifies the name of the current stage. This is a required option.
Stage names are arbitrary and have meaning only to the **run_parallel_jobs** command and the child jobs that use **send_flow_status** to communicate flow status back to the parent. The stage names **must** be consistent between the child and parent. If a stage name is used in a **send_flow_status** call that is not known to the parent job, a warning message is generated in the parent. If a stage name is used by the parent but not used in the child, the named stage will appear to be "skipped" in the parent (since no information will ever be reported on that stage by the child).

`-status current_status`

Reports the status back to the parent. This is a required option.

There are many status values, but the only ones that make sense coming from a child are **Running** (when the stage is starting) and then one of **Succeeded** or **Failed** (when the stage is over).

`-eof`

Indicates to the parent that this job has completed and will be exiting soon. This option is used to improve the latency in detecting job completion. This option should **only** be used right before an exit. If any processing takes place after this option is used but before the child process exits, it creates a flow that is intermittently error-prone.

`-verbose`

Generates detailed information about communication with the parent.

This option is used only for debugging when setting up a flow. After a flow is working correctly, it is not needed.

DESCRIPTION

This command sends status information back from a child job launched through the **run_parallel_jobs** command back to the parent job. The command must have the job name for the child and the host name and parent number of the parent process. This information can be specified on the command line or it can come from the UNIX environment. Because the **run_parallel_jobs** command sets these values, you should never need to set them. The only information you need to provide is the current stage name (set up by convention between the **run_parallel_jobs** command in the parent and the batch script being used in the child) and the actual status. Status values are predefined, and the only ones that make sense in this context are **Running**, **Succeeded**, and **Failed**. The `-eof` option is provided to let the parent know that the child process has finished its flow and will be exiting (to reduce the latency in the parent process in determining when the child exits). The `-verbose` option is provided for use when setting up and debugging a flow.

EXAMPLES

The following example sends status in the briefest form:

```
prompt> send_flow_status -stage PlaceOpt -status Succeeded
1
```

The following example sends status using all the options:

```
prompt> send_flow_status -job_name BLOCK_1 -host_name ada \
-port_num 4523 -stage RouteOpt -status Succeeded -eof
1
```

SEE ALSO

`run_parallel_jobs(2)`

set_active_scenarios

Specifies which scenarios are to be active.

SYNTAX

```
int set_active_scenarios  
scenario_list | -all
```

Data Types

scenario_list list of strings

ARGUMENTS

scenario_list
Specifies which scenarios are active.
This argument and the -all option are mutually exclusive; specify only one.

-all
Makes all scenarios active.
This option and the *scenario_list* argument are mutually exclusive; specify only one.

DESCRIPTION

Specifies which scenarios are active.

To report timing results with back-annotated delays for newly activated scenarios, run **extract_rc -estimate** before issuing **report_timing** command.

Multicorner-Multimode Support

This command uses information from both active and inactive scenarios.

EXAMPLES

The following example uses **set_active_scenarios** to set only one scenario active.

```
prompt> create_scenario MODE1  
prompt> create_scenario MODE2  
prompt> set_active_scenarios {MODE1}  
prompt> all_scenarios  
MODE1 MODE2  
prompt> all_active_scenarios  
MODE1
```

SEE ALSO

`all_scenarios(2)`

```
all_active_scenarios(2)
current_scenario(2)
remove_scenario(2)
create_scenario(2)
extract_rc(2)
```

set_ahfs_options

Specifies the options to be used when running automatic high-fanout synthesis (AHFS).

SYNTAX

```
status set_ahfs_options
[-optimize_buffer_trees true | false]
[-enable_port_punching true | false]
[-no_port_punching cells]
[-default_reference references]
[-port_map_file file_name]
[-preserve_boundary_phase true | false]
[-constant_nets true | false]
[-hf_threshold integer]
[-mf_threshold integer]
[-remove_effort none | medium | high]
[-default]
```

Data Types

<i>cells</i>	string
<i>references</i>	string
<i>file_name</i>	string

ARGUMENTS

-optimize_buffer_trees true | false

Controls whether automatic high-fanout synthesis uses the enhanced optimization algorithm to create buffer trees.

By default, this option is true and automatic high-fanout synthesis optimizes the buffer trees, including automatic analysis and removal of some buffer and inverter trees and resynthesis, as needed for better QoR. The enhanced algorithm is also aware of more design constraints, which often leads to reduced congestion better timing QoR.

When **-optimize_buffer_trees** is true, the following options are ignored: **-hf_threshold**, **-mf_threshold**, **-remove_effort**.

When set to false, automatic high-fanout synthesis uses the original algorithm to insert buffer or inverter trees as needed for high-fanout synthesis. The removal of buffer trees is controlled separately by the **-remove_effort** option and the **remove_buffer_tree** command.

This option prevents modification of the specified buffer trees during **create_buffer_tree**, **remove_buffer_tree**, and the high-fanout synthesis phase of **place_opt**. It has no effect on the optimization phases of other commands, such as **psynopt**.

-enable_port_punching true | false

Enables or disables port punching in automatic high-fanout synthesis optimization. The default is true.

When this option is true, automatic high-fanout synthesis can insert buffers or inverters across hierarchy boundaries to get better QoR.

You can use the **-no_port_punching** option to prevent port punching on specific

hierarchical cells, even when **-enable_port_punching** is true.

-no_port_punching cells

Specifies the hierarchical cells for which automatic high-fanout synthesis cannot add or remove any ports. The cells are specified as a string of hierarchical cell instance names separated by commands, such as "sub/mod1 sub/block2".

You can use this option to limit the port punching on specific hierarchical cell instances when **-enable_port_punching** is true (the default). If **-enable_port_punching** is false, there is no need for this option.

During the removal phase of automatic high-fanout synthesis, inverters might be moved across hierarchical boundaries to cancel "inverter chains" for better QoR. If you do not want to allow any phase changes on hierarchical boundaries, use the **-preserve_boundary_phase true** option instead of using this option.

-default_reference references

Specifies a list of buffers or a list of inverters for automatic high-fanout synthesis to use when constructing new buffer or inverter trees. The list must be either entirely buffers or entirely inverters, not a mixture of buffers and inverters.

You can specify the reference cells with or without the library prefix.

If you do not specify this option, or if the specified references are not active buffer cells, automatic high-fanout synthesis tries to use all active buffer or inverter cells defined in the library.

When **-optimize_buffer_trees** is true and the list contains buffers, automatic high-fanout synthesis automatically uses inverters as needed to construct logically correct buffer and inverter trees.

-port_map_file file_name

Specifies the name of the port mapping file. This option is used only when **-enable_port_punching** is set to true.

If you do not specify this option and **-enable_port_punching** is set to true, the default name of the port mapping file is <design_name>_port_map.<version>. The initial file is created with a version of 0. If the specified port mapping file already exists, a new version of the file is created with the version number increased by one, such as <design_name>_port_map.1.

-preserve_boundary_phase true | false

Specifies whether automatic high-fanout synthesis is allowed to move inverters across hierarchical boundaries so that the logical phase of the boundary changes.

The default is false, meaning that automatic high-fanout synthesis can freely move inverters across hierarchical boundaries. This default behavior often leads to better QoR because automatic high-fanout synthesis can move inverters and cancel inverter chains as needed.

-constant_nets true | false

Specifies that automatic high-fanout synthesis should work on constant nets as well as signal nets.

By default, automatic high-fanout synthesis ignores constant nets.

-hf_threshold integer

Specifies the fanout threshold for high-fanout optimization during automatic

high-fanout synthesis.

The default value is 100.

The tool performs maximum-fanout-driven high-fanout optimization on the nets in your design that have fanout greater than or equal to the threshold value you set for this option, after coarse placement and before normal optimization. If you set the threshold for this option as less than or equal to 0, the tool does not process any of the nets.

When **-optimize_buffer_trees** is true, this option is ignored.

-mf_threshold *integer*

Specifies the fanout threshold for medium-fanout optimization during automatic high-fanout synthesis. The default value is -1.

The tool performs timing-driven medium-fanout optimization on the nets in your design that have fanout greater than or equal to the threshold value you set for this option, after coarse placement and before normal optimization. If you set the threshold for this option as less than or equal to 0, the tool does not process any of the nets.

If you set the threshold for this option equal to or more than the value you specified for the **-hf_threshold** option, this optimization is not enabled.

When **-optimize_buffer_trees** is true, this option is ignored.

-remove_effort none | medium | high

Specifies the effort level for removing existing buffer trees or inverter trees during automatic high-fanout synthesis.

The default value is none.

Automatic high-fanout synthesis removes existing buffer trees or inverter trees in your design, after coarse placement and before normal optimization, guided by the value you set for this option.

The accepted values are as follows:

* **none** (the default)

Indicates that the the tool does not remove any buffer trees or inverter trees.

* **medium**

Indicates that the tool removes buffer trees or inverter trees that have negative maximum delay slack.

* **high**

Indicates that the tool removes all buffer trees and inverter trees.

When **-optimize_buffer_trees** is true, this option is ignored.

-default

Resets all of the options set by previous **set_ahfs_options** commands to their default values.

DESCRIPTION

This command defines the constraints used while performing automatic high-fanout optimization.

You can report the settings by running the **report_ahfs_options** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows how to set automatic high-fanout synthesis options for current design. It sets the high-fanout optimization threshold to 50, turns off the medium-fanout optimization, enables port punching, sets the buffer removal effort to remove only buffer tree with a negative maximum delay slack, and uses cells of type BUFX3M or ss_0v80_125c/DLY4X4M during high-fanout synthesis. The **-preserve_boundary_phase false** option indicates that inverters should not move across hierarchical boundaries. The **-no_port_punching "sub/mod1 sub/mod2"** option indicates that the sub/mod1 and sub/mod2 hierarchical cells should have no changes to their ports during automatic high-fanout synthesis.

```
prompt> set_ahfs_options \  
-optimize_buffer_trees false \  
-hf_threshold 50 \  
-mf_threshold -1 \  
-enable_port_punching true \  
-remove_effort medium \  
-no_port_punching "sub/mod1 sub/mod2" \  
-preserve_boundary_phase false \  
-default_reference "BUFX3M ss_0v80_125c/DLY4X4M"  
prompt> report_ahfs_options  
Report AHFS options:  
*****  
AHFS options for the design:  
  Enable port punching: ON  
  Default Reference : BUFX3M ss_0v80_125c/DLY4X4M  
  Port Map File :  
  Preserve Boundary Phase : OFF  
  No Port Punching on these hier cells : "sub/mod1 sub/mod2"  
  Constant Nets allowed during AHFS optimization : OFF  
  -optimize_buffer_trees FALSE  
    High-fanout(HF) threshold: 50  
    Medium-fanout(MF) threshold: -1  
    Remove effort: medium
```

The following example shows how to use the **-optimize_buffer_trees** option to activate optimization of buffer and inverter trees, including automatic analysis and removal of some trees, and resynthesis to achieve better QoR. The report indicates that the **-hf_threshold**, **-mf_threshold**, and **-remove_effort** options are ignored when **-optimize_buffer_trees** is true. The buffer tree driven by the u_start/gte_107/U151/Y leaf pin is protected from automatic high-fanout synthesis optimization during **create_buffer_tree** and **remove_buffer_tree**.

```
prompt> set_ahfs_options \  
-optimize_buffer_trees true \  
prompt> report_ahfs_options  
Report AHFS options:  
*****  
AHFS options for the design:
```



```
Enable port punching: ON
Default Reference : BUFX3M ss_0v80_125c/DLY4X4M
Port Map File :
Preserve Boundary Phase : OFF
No Port Punching on these hier cells : ""
Constant Nets allowed during AHFS optimization : OFF
-optimize_buffer_trees TRUE
    Ignoring: High-fanout(HF) threshold: 50
    Ignoring: Medium-fanout(MF) threshold: -1
    Ignoring: Remove effort: medium
```

SEE ALSO

```
create_buffer_tree(2)
remove_buffer_tree(2)
report_ahfs_options(2)
```

set_always_on_cell

On-The-Fly setting of library always_on cell.

SYNTAX

Boolean **set_always_on_cell**
cell_name
[-reset]

Data Types

cell_name string

ARGUMENTS

cell_name
Specify the library cell as always_on cell.

-reset
reset the always_on cell attribute.

DESCRIPTION

The **set_always_on_cell** command set all the cells with the same name specified as always_on cell. Always on cell can be only buffer or inverter library cell.

If -reset is issued, the always_on cell attribute will be cleaned up.

If -reset is combined with other options, reset happens first, other options will be set on the cell after reset.

EXAMPLES

```
prompt> set_always_on_cell INV_A0
```

SEE ALSO

report_lib(2)
check_library(2)

set_always_on_strategy

Sets the always-on strategy for shutdown power domains.

SYNTAX

```
status set_always_on_strategy  
-object_list list_of_domains  
-cell_type single_power | dual_power
```

Data Types

list_of_domains collection

ARGUMENTS

```
-object_list list_of_domains  
    Specifies one or more power domains for which to define the always on  
    strategy.  
  
-cell_type single_power | dual_power  
    Specifies the type of always_on cells to be used; either single power cells  
    or dual power cells with backup power. The default is dual_power.
```

DESCRIPTION

This command is used to set the always_on strategy for shut-down power domains. The strategy indicates the type of cells that are to be used for always-on buffering. You can choose to use standard cells with single power cells as AO or dual power cells with backup supply. This information drives the cell selection of the optimizations.

Use dual power cells for always-on if you want to route backup power to these cells. Use single power cells as always-on if you are reserving special sites for these cells in the floorplan.

This command ensures that the specified *list_of_domains* consists of power domains that are defined as power down.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLE

The following example sets the always-on strategy on "switchable_domain" to use single power standard cells.

```
prompt> set_always_on_strategy\
```

```
-object_list switchable_domain\  
-cell_type single_power  
1
```

SEE ALSO

```
create_power_domain(2)  
remove_power_domain(2)  
report_power_domain(2)  
report_power_guide(2)  
set_power_guide(2)  
unset_power_guide(2)
```

set_annotated_check

Sets the setup, hold, recovery, or removal timing check value between two pins.

SYNTAX

```
int set_annotated_check  
check_value  
-from from_pins  
-to to_pins  
-setup  
  | -hold  
  | -recovery  
  | -removal  
  | -nochange_high  
  | -nochange_low  
[-rise | -fall]  
[-clock clock_check]  
[-worst]
```

Data Types

<i>check_value</i>	float
<i>from_pins</i>	list
<i>to_pins</i>	list

ARGUMENTS

check_value
Specifies the timing check value between pins on the same cell. You must express *check_value* in units consistent with the technology library used during optimization. For example, if the technology library specifies delay values in nanoseconds, *check_value* must be expressed in nanoseconds: *check_value* can be negative.

-from *from_pins*
Specifies a list of leaf cell clock pins that are the startpoints of the timing arcs for which checks are to be annotated.

-to *to_pins*
Specifies a list of leaf cell data pins that are the endpoints of the timing arcs for which checks are to be annotated.

-setup | -hold | -recovery | -removal | -nochange_high | -nochange_low
Specifies the type of the timing check. There must be a corresponding timing arc between the *from* and *to* pins.
The **-setup** option indicates that data must be stable for the amount of time specified by *check_value* before the closing clock edge.
The **-hold** indicates that data must be stable for the amount of time specified by *check_value* after the closing clock edge.
The **-recovery** option indicates that an asynchronous set or clear inactive edge cannot occur within *check_value* before the closing clock edge. This is essentially a setup requirement on an asynchronous pin, but only the inactive

transition is considered.

The **-removal** option indicates that an asynchronous set or clear inactive edge cannot occur until *check_value* time after the closing clock edge. This is essentially a hold requirement on an asynchronous pin, but only the inactive transition is considered.

The **-nochange_high** option provides a timing check that indicates that the data must not rise within *check_value* time before the clock becomes active and must not fall until *check_value* time after the clock becomes inactive.

The **-nochange_low** option provides a timing check that indicates that the data must not fall within *check_value* time before the clock becomes active and must not rise until *check_value* time after the clock becomes inactive.

-rise | -fall

Specifies whether the check is for data rise or fall transition. If you do not specify either **-rise** or **-fall**, both values are set.

The **-rise** option specifies a rise data transition that corresponds to the check before the activating clock edge. A rise data transition corresponds to the check after the deactivating clock edge. A rise clock transition indicates that the clock is active-high.

The **-fall** option specifies a fall data transition that corresponds to the check after the deactivating clock edge. A fall data transition corresponds to the check before the activating clock edge. A fall clock transition indicates that the clock is active-low.

-clock *clock_check*

Specifies whether the check is for clock rising or falling. Valid values for *clock_check* are **rise** and **fall**. By default, checks for both clock rise and fall are set.

-worst

Specifies that the check is to overwrite the previously-annotated check only if the *check_value* specified is worse than the check value previously annotated. By default, *check_value* overwrites any previously-annotated value.

DESCRIPTION

Annotates a timing check between two or more pins on a cell or a net in the *current design*. This might be appropriate after place and route, for technologies where the timing check value varies for different instances and the library timing check values do not provide sufficient accuracy.

If the design is not already linked, **set_annotated_check** links it automatically.

The command can be used for pins at lower levels of the design hierarchy. Pins are specified as "INSTANCE1/INSTANCE2/PIN_NAME."

To list annotated timing check values, use the **report_annotated_check** command. To remove annotated timing check values from a design, use the **remove_annotated_check** or **reset_design** command. To see the affect of **set_annotated_check** for a specific instance, use the **report_timing** command.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example annotates a setup time of 2.1 units between clock pin *CP* of cell instance *u1/ff12* and data pin *D* of the same cell instance.

```
prompt> set_annotated_check -setup 2.1 -from u1/ff12/CP \  
-to u1/ff12/D
```

The following example annotates a nochange check between data low and clock low. The nochange margin before the clock is 1.3 and the nochange margin after the clock is 2.4.

```
prompt> set_annotated_check -nochange_low 1.3 -clock fall \  
-fall -from u1/CP -to u1/EN  
prompt> set_annotated_check -nochange_low 2.4 -clock fall \  
-rise -from u1/CP -to u1/EN
```

SEE ALSO

```
current_design(2)  
link(2)  
remove_annotated_check(2)  
report_annotated_check(2)  
report_timing(2)  
reset_design(2)
```

set_annotated_delay

Sets the net or cell delay value between two pins.

SYNTAX

```
int set_annotated_delay  
-net | -cell  
[-load_delay load_delay_type]  
[-rise | -fall] [-min] [-max] delay_value  
-from from_pins -to to_pins [-worst]
```

Data Types

<i>delay_value</i>	float
<i>from_pins</i>	list
<i>to_pins</i>	list

ARGUMENTS

-net | -cell

Specifies whether the delay annotated is a net or cell delay. This is a required argument.

-load_delay *load_delay_type*

Specifies whether to include load delay as part of annotated net delays or as part of annotated cell delays. *load_delay_type* must be the value of net or cell. Load delay is the portion of cell delay arising from the capacitive load of the net the cell is driving. Set *location_name* to either *net* or *cell*. All timing arcs of the same net must be annotated with the same *location_name*; and all timing arcs of the same cell must be annotated with the same *location_name*.

-rise | -fall

Specifies whether the delay is for data rise or fall transition. If you don't specify **-rise** or **-fall**, both values are set.

-min -max

Specifies whether the delay is to be used for minimum delay analysis or maximum delay analysis. By default, the delay is used for maximum and minimum delay analysis. If a delay value is annotated for only minimum or maximum delay analysis, that value is used for both maximum and minimum delays. It is possible to annotate different delay values for minimum and maximum analysis, but it is not possible to annotate only for minimum or annotate only for maximum.

delay_value

Specifies the delay value between pins on the same cell or net. The *delay_value* must be expressed in units consistent with the technology library used during optimization. For example, if the technology library specifies delay values in nanoseconds, *delay_value* must be expressed in nanoseconds.

-from *from_pins*

Specifies a list of leaf cell pins or top-level ports that are the startpoints of the timing arcs for which delays are to be annotated.

-to *to_pins*

Specifies a list of leaf cell pins or top-level ports that are the endpoints of the timing arcs for which delays are to be annotated.

-worst

Indicates that the delay is to overwrite the previously annotated delay only if the *delay_value* specified is worse than the delay value previously annotated. By default, *delay_value* overwrites any previously annotated value.

DESCRIPTION

Annotates the cell delay between two or more pins on a cell in the current design or the net delay between two or more pins on the same net. With the **-net** option, pins in *from_pins* must be cell output or inout pins, and pins in *to_pins* must be cell input or inout pins. With the **-cell** option, both **-from** and **-to** options are required; pins in *from_pins* must be cell input or inout pins, and pins in *to_pins* must be cell output or inout pins. To verify the accuracy of the back-annotation done by **set_annotated_delay**, run **update_timing**.

If the current design is hierarchical, you must link it using **link -all** before using **set_annotated_delay**.

Load delay, also known as extra source gate delay, is the portion of cell delay caused by the capacitive load of the net being driven. Some delay calculators consider load delay part of the net delay and others consider it part of the cell delay. If your annotated delay value (for either cell or net) assumes that load delay is part of the cell delay, use **-load_delay cell**. If your delay value assumes that load delay is included in the net delay, use **-load_delay net**. By default, load delays are assumed to be in cell delays; and so they appear in **report_timing** path listings.

The specified delay value overrides the internally-estimated cell and net delay value. If the specified pins are not in the same cell or on the same net, then when the timing is updated an error message is generated and *delay_value* is discarded for those pins.

You can use **set_annotated_delay** for pins at lower levels of the design hierarchy. Pins are specified as "INSTANCE1/INSTANCE2/PIN_NAME."

To list annotated delay values, use **report_annotated_delay**. To remove the annotated cell or net delay values from a design, use **remove_annotated_delay** or **reset_design**.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example annotates a cell delay of 20 units between input pin "A" of cell instance "U1/U2/U3" and output pin "Z" of the same cell instance. The delay value of 20 includes the load delay.

```
prompt> set_annotated_delay -cell -load_delay cell 20 -from U1/U2/U3/A -to  
U1/U2/U3/Z
```

The following example annotates a rise net delay of 1.4 units between output pin "U1/Z" and input pin "U2/A". The delay value for this net does not include load delay.

```
prompt> set_annotated_delay -net -rise -load_delay cell 1.4 -from U1/Z -to  
U2/A
```

The following example annotates a rise net delay of 12.3 units for minimum delay analysis between the same output pins. In this case the net delay value does include load delay.

```
prompt> set_annotated_delay -net -rise -min -load_delay net 12.3 -from U1/Z  
-to U2/A
```

SEE ALSO

current_design(2)
link(2)
report_timing(2)
remove_annotated_delay(2)
report_annotated_delay(2)
reset_design(2)
set_input_delay(2)
target_library(3)

set_annotated_transition

Sets the transition time at a given pin.

SYNTAX

```
int set_annotated_transition  
[-rise | -fall] [-min] [-max] transition  
port_pin_list
```

Data Types

<i>transition</i>	float
<i>port_pin_list</i>	list

ARGUMENTS

-rise | -fall
Specifies whether the transition time is for data rise or data fall transition. If you do not specify **-rise** or **-fall**, both values are set.

-min -max
Specifies whether the transition is to be used for minimum delay analysis or maximum delay analysis. By default, the transition value is used for maximum and minimum delay analysis. If a transition value is annotated for only minimum or maximum delay analysis, that value is used for both maximum and minimum delays. It is possible to annotate different transition values for minimum and maximum analysis, but it is not possible to annotate only for minimum or annotate only for maximum.

transition
Specifies the transition value at the pins supplied with the *port_pin_list* argument. The transition value must be expressed in units consistent with the technology library used during optimization. For example, if the technology library specifies transition values in nanoseconds, the transition values must be expressed in nanoseconds.

port_pin_list
Specifies a list of leaf cell pins or top-level ports that are the endpoints of the timing arcs for which delays are to be annotated.

DESCRIPTION

To list annotated transition values, use **report_annotated_transition**. To remove the annotated transition values from a design, use **remove_annotated_transition** or **reset_design**.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example annotates a transition time of 20 units at input pin "A" of cell instance "U1/U2/U3".

```
prompt> set_annotated_transition 20 U1/U2/U3/A
```

The following example annotates a rise transition of 1.4 units at the input pin "U5/A".

```
prompt> set_annotated_transition -rise 1.4 U5/A
```

The following example annotates a rise transition of 12.3 units for minimum delay analysis at U5/A.

```
prompt> set_annotated_transition -rise -min 12.3 U5/A
```

SEE ALSO

```
current_design(2)  
link(2)  
report_timing(2)  
remove_annotated_transition(2)  
report_annotated_transition(2)  
reset_design(2)  
set_input_transition(2)  
target_library(3)
```

set_aocvm_coefficient

Sets advanced on-chip variation (OCV) random coefficients on cells and library cells for use during an advanced OCV analysis.

SYNTAX

```
status set_aocvm_coefficient  
rnd_coeff  
object_list
```

Data Types

```
rnd_coeff          float  
object_list       list
```

ARGUMENTS

rnd_coeff
Specifies the random coefficient applied to the objects in the *object_list*. The *rnd_coeff* setting should be a floating-point value greater than zero.

object_list
Specifies a list of library cells or leaf cells on which the coefficient is set.

DESCRIPTION

Sets random coefficients on library cells or leaf cells for advanced on-chip variation (OCV) analysis. These are user-specified depth adjustment factors (default of 1.0) that you can apply at the library cell or cell instance level. These coefficients are not required for an advanced OCV analysis. However, their application may increase the accuracy of the analysis.

If different coefficients have been annotated on a library cell and an instance of that cell, the coefficient annotated on the cell instance takes precedence.

Random coefficients are used to calculate cell path depths. The default increment of path depth for any cell is 1.0. The path depth increment for a cell if defined is *rnd_coeff*.

To display the advanced OCV coefficients that have been set, use the **report_aocvm** command.

To remove advanced OCV coefficients from a cell, use the **remove_aocvm -coefficient** or **reset_design** command.

EXAMPLES

The following example sets an advanced OCV random coefficient on a library cell named **lib/bufA**.

```
prompt> set_aocvm_coefficient 1.2 [get_lib_cells lib/bufA]
```

SEE ALSO

```
read_aocvm(2)  
remove_aocvm(2)  
report_aocvm(2)  
timing_aocvm_enable_analysis(3)
```

set_attribute

Sets an attribute to a specified value on the specified list of objects.

SYNTAX

```
collection set_attribute
[-class class_name]
object_list
attribute_name
attribute_value
[-quiet]
```

Data Types

<i>class_name</i>	string
<i>object_list</i>	list
<i>attribute_name</i>	string
<i>attribute_value</i>	string

ARGUMENTS

-class *class_name*
Specifies the class name for the object specified in *object_list*, if the element of *object_list* is a name. Valid classes are design, port, cell, net, etc.

object_list
A list of objects on which the attribute is to be set. Each element in the list is either a collection or a pattern that is combined with the *class_name* to find the objects.

attribute_name
Specifies the name of the attribute to be set.

attribute_value
Specifies the value of the attribute. The data type must be the same as that of the attribute.

-quiet
Turns off the warning message that would otherwise be issued if the attribute or objects are not found.

DESCRIPTION

This command sets the value of an attribute on an object. For a complete list of attributes, see the **attributes** man page.

This command creates a collection of objects that have the specified attribute value set. A returned empty string indicates that no object has been set.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example defines an attribute named X for cells, then sets the value on all cells in this level of the hierarchy:

```
prompt> define_user_attribute -type int -class cell X  
cell  
prompt> set_attribute [get_cells *] X 30  
{ "U1" }
```

SEE ALSO

```
collections(2)  
define_user_attribute(2)  
get_attribute(2)  
list_attributes(2)  
remove_attribute(2)
```


set_auto_disable_drc_nets

Sets the **auto_disable_drc_net** attribute on the current design, causing the specified networks to be have DRC disabled. This command was previously called **set_auto_ideal_nets**.

SYNTAX

```
int set_auto_disable_drc_nets
[-default]
[-none]
[-all]
[-clock true | false]
[-constant true | false]
[-scan true | false]
```

ARGUMENTS

-default

Disables design rule checking (DRC) on nets in clock trees and constant nets in the *current design*; equivalent to **-clock true** and **-constant true**. This is also the default behavior if no option is selected or if the command is not executed. You cannot use **-default** with any other option.

-none

Enables DRC for all networks in the *current design*, including nets in clock trees. You cannot use **-none** with any other option.

-all

Disables DRC on all applicable nets in the *current design*. It is equivalent to setting **-clock true**, **-constant true**, and **-scan true**.

-clock true | false

Disables or enables DRC on clock networks in the *current design*. When **true** (the default), it automatically disables DRC on only clock networks: When **false**, it enables DRC on them. By default, Design Compiler treats clock networks as ideal nets even if this command is not executed. To disable this default behavior and prevent clock networks from being treated as ideal nets, set this option to **false**.

-constant true | false

Disables or enables DRC on constant nets in the *current design*. When **true** (the default), it automatically disables DRC on constant nets: When **false**, it enables DRC on them.

-scan true | false

Disables or enables DRC on scan nets in the *current design*. When **true**, it automatically disables DRC on scan nets: When **false** (the default), it enables DRC on them. This option does not prevent Design Compiler from automatically disabling DRC on clock networks and constant nets, unless **-clock false** and **-constant false** is also used.

DESCRIPTION

Disables design rule checking for specified networks in the *current design*. If the command is executed without options, or if the command is not executed at all, by default, Design Compiler disables DRC for all clock nets and constant nets. The command options are additive each time a new **set_auto_disable_drc_nets** command is executed.

To prevent clock nets from having DRC disabled, use the **-clock false** or **-none** options.

To prevent constant nets from having DRC disabled, use the **-constant false** or **-none** options.

DRC disabled nets are a network of nets that are free from the max_capacitance, max_fanout, and max_transition design rule constraints. They are useful for reducing DRC violations caused by clock trees, because these networks usually have high max_capacitance and max_fanout violations.

You can add the **dont_touch** attribute to the DRC disabled nets by using the **set_dont_touch_network** or the **set_dont_touch** command.

Note that auto_disable_drc nets are different from ideal_nets and ideal_networks, since these also use ideal timing and set dont_touch on the nets (and cells, in the case of ideal_networks).

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example disables DRC on clock trees and constant nets in the *current design*.

```
prompt> set_auto_disable_drc_nets -clock true -constant true
```

The following example disables DRC on only the constant nets in the *current design*.

```
prompt> set_auto_disable_drc_nets -constant true
```

The following example re-enables DRC for all relevant nets, including those on clock networks, from the *current design*.

```
prompt> set_auto_disable_drc_nets -none
```

SEE ALSO

```
report_attribute(2)
report_design(2)
reset_design(2)
set_dont_touch(2)
set_dont_touch_network(2)
```

```
set_auto_disable_drc_nets
2212
```

```
set_ideal_net(2)
set_ideal_network(2)
```

set_buffer_opt_strategy

Invokes new buffering strategy.

SYNTAX

```
int set_buffer_opt_strategy  
[-effort string]
```

ARGUMENTS

-effort string

Specifies the effort level for the new buffering strategy. The different effort levels dictate the amount of time spent in the new buffering strategy. Typically, a higher effort level will result in better reduction in buffer and inverter counts. The **none** effort level can be used to turn off the new buffering strategy.

If fbset_buffer_opt_strategyfp is used without any arguments, the **effort** value of new buffering strategy is **medium**.

If fbset_buffer_opt_strategyfp is not used, the **effort** value of new buffering strategy is **none**.

DESCRIPTION

The fbset_buffer_opt_strategy command invokes a new buffering strategy, that reduces buffer and inverter counts, without significantly affecting the quality of results (QoR).

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example specifies high-effort buffering strategy.

```
prompt> set_buffer_opt_strategy -effort high
```

The following example turns off the new buffering strategy. prompt>

```
set_buffer_opt_strategy -effort none
```

set_case_analysis

Specifies that a port or pin is at a constant logic value 1 or 0, or is considered with a rising or falling transition..

SYNTAX

```
string set_case_analysis  
value  
port_or_pin_list
```

Data Types

```
port_or_pin_list    list
```

ARGUMENTS

value

Specifies the constant logic value or the transition to assign to the specified pin or port. The valid constant values are **0**, **1**, **zero**, or **one**. The valid transition values are **rising**, **falling**, **rise**, and **fall**.

port_or_pin_list

Lists ports or pins to which the case analysis is assigned. The command performs no backward constant propagation.

DESCRIPTION

Specifies that a port or pin is at a constant logic value 1 or 0.

Case analysis is a way to specify a given mode of the design without altering the netlist structure. For the current timing analysis session, you can specify either that some signals are at a constant value (1 or 0) or that only one type of transition (rising or falling) is to be examined. When you specify case analysis to a constant value, the constant value is propagated through the network as long as a controlling value for the traversed logic is at the constant value.

For example, if you specify that one of the inputs of a NAND gate is a constant value 0, it is propagated to the NAND output, which is now considered at a logic constant 1. This propagated constant value, itself, is propagated to all cells driven by this signal.

In the event of case analysis on transition, the given pin or port is only considered for timing analysis with the specified transition. The other transition is disabled.

All analysis commands use case analysis information, including the false-path detection algorithm used by the **report_timing** command with the **-true** option and the **report_timing** command with the **-justify** option.

You can use case analysis (in addition to the mode commands) to fully specify the mode of a design. For example, you can use the **set_mode** command to specify a design

that instantiates models with a TESTMODE that is disabled during the timing analysis session. In addition, if a TESTMODE signal exists on the design, it is specified to a constant logic value, so that all test logic that the TESTMODE signal controls is disabled.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example specifies that a port named *IN1* is at a constant logic value 0.

```
prompt> set_case_analysis 0 IN1
```

The following example specifies how to disable the TESTMODE of a design for which instances are models having a TESTMODE mode. The design also has a *TEST_PORT* port set to a constant logic value 0.

```
prompt> remove_mode TESTMODE U1/U2  
prompt> set_case_analysis 0 TEST_PORT
```

The following example specifies that the pins U1/U2/A is only considered for a rising transition. The falling transition on these pins are disabled.

```
prompt> set_case_analysis rising U1/U2/A
```

SEE ALSO

```
remove_case_analysis(2)  
report_case_analysis(2)  
set_mode(2)
```

set_cbt_options

Sets options used by the **create_buffer_tree** command.

SYNTAX

```
int set_cbt_options  
[-threshold threshold_value]  
[-references list_of_references]  
[-default]
```

Data Types

threshold_value int

ARGUMENTS

- threshold *threshold_value*
Specifies a fanout threshold used by the **create_buffer_tree** command to search for high fanout nets. The default value is 100.
- references *list_of_references*
Specifies a list of library cells (buffers and inverters) the **create_buffer_tree** command can use. If not specified, all library buffers and inverters can be used.
- default
Resets all the options to their default values. This option cannot be combined with any other option.

DESCRIPTION

The **set_cbt_options** command sets options used by the **create_buffer_tree** command. These options take effect only when **set_ahfs_options -optimize_buffer_trees false** is specified.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

```
set_cbt_options -threshold 1000 -references {BUFFLV3 INVLV3}
```

SEE ALSO

```
create_buffer_tree(2)  
report_cbt_options(2)
```

set_cell_degradation

Sets the **cell_degradation** attribute to a specified value on specified ports or designs.

SYNTAX

```
int set_cell_degradation  
cell_degradation_value  
object_list
```

Data Types

<i>cell_degradation_value</i>	float
<i>object_list</i>	list

ARGUMENTS

cell_degradation_value
Specifies a capacitance value for setting the **cell_degradation** attribute. You must express *cell_degradation_value* in capacitance units consistent with that used by the technology library during optimization. For example, if the library specifies capacitance values in picofarads, you must express *cell_degradation_value* in picofarads.

object_list
Specifies a list of names of input ports for setting the **cell_degradation** attribute.

DESCRIPTION

Sets the **cell_degradation** attribute to *cell_degradation_value* on the specified ports or designs. During optimization, the tool attempts to ensure that the capacitance value for a net is less than the *cell_degradation_value* if the value of the variable **compile_fix_cell_degradation** is *true*.

If cell degradation tables are already specified in a technology library (implicit constraints), **compile** automatically tries to meet them if the value of the variable **compile_fix_cell_degradation** is *true*. The cell degradation tables give the maximum capacitance that can be driven by a cell as a function of the transition times at the inputs of the cell.

By default, a port has no cell degradation constraint.

The **cell_degradation** attribute and the **max_capacitance**, **max_fanout**, and **max_transition** attributes are design rule constraints; the **max_delay** and **max_area** attributes are optimization constraints. Design rule constraints reflect those technology-specific restrictions that MUST be met for a design to function correctly. Optimization constraints reflect desirable, but not crucial goals and restrictions for the operation of a design. The tool attempts to meet all constraints placed on a design, but gives priority to design rule constraints in the optimization process. Therefore, optimization gives preference to **cell_degradation**

and other design rule constraints, even if they adversely affect optimization constraints on a design.

To get information about optimization and design rule constraints, use **report_constraint**. To remove the **cell_degradation** attribute from a port, use **remove_attribute**.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example sets a maximum capacitance value of 2.0 units on the port named `late_riser`:

```
prompt> set_cell_degradation 2.0 late_riser
```

SEE ALSO

```
all_inputs(2)  
all_outputs(2)  
characterize(2)  
remove_attribute(2)  
report_constraint(2)  
set_max_capacitance(2)
```

set_cell_internal_power

Sets or removes the **power_value** attribute on the specified pins. The value represents the power consumption for a single toggle of each pin.

SYNTAX

```
int set_cell_internal_power  
[-delete_all]  
pin_list  
[power_value [unit]]
```

Data Types

<i>pin_list</i>	object list
<i>power_value</i>	float
<i>unit</i>	string

ARGUMENTS

-delete_all

Deletes all of the internal power annotations from all of the pins in the design.

This option cannot be used with *pin_list*, *power_value*, or *unit*.

pin_list

Specifies a list of pins on which to either set the **power_value** attribute, or remove previously-annotated **power_value** attributes.

This option cannot be used with the **-delete_all** option.

power_value

Specifies the value with which the **power_value** attribute is to be set on the specified pins. If *power_value* is not specified, any existing **power_value** attributes are removed from the specified pins.

This option cannot be used with the **-delete_all** option.

unit

Specifies the unit of the power value. Allowed values for *unit* are as follows:

GW	MW	KW	W	mW
uW	nW	pW	fW	aW

If *unit* is not specified, the tool uses the library power unit. If the library does not have any units, the tool displays an error message.

This option cannot be used with the **-delete_all** option.

DESCRIPTION

This command sets or removes the **power_value** attribute on specified pins. The *power_value* is the value with which to set the **power_value** attribute, and represents the power consumption for a single toggle of the pin. If a cell has at least one such annotated pin, its internal power is calculated as the sum of the pin power values, where each pin power value is calculated by multiplying the annotated power value on that pin by the pin toggle rate.

If this command is issued without the *power_value* argument, the tool removes any existing **power_value** attributes from the specified pins. If the *power_value* argument is specified without *unit*, the tool uses the power unit of the library. If the library does not have any units, the tool generates an error message.

Use this command to override a cell's library power characterization in situations where that characterization does not apply. A common use is when you manually replace an entire cloud of logic with a single cell and you want the single cell's power consumption to represent that of the cloud of logic. For example, if you replace a clock tree by a single buffer cell, you can set the **power_value** attribute on the output pin of the buffer cell with the value of the power consumption for one clock toggle of the entire clock tree. Although the buffer cell might have been power-characterized in the library, its power consumption is now calculated using the value of the **power_value** attribute set by the **set_cell_internal_power** command.

Do not use this command as a routine part of the power characterization flow. It is considered best practice to use a cell's library power characterization unless there is a reason to override it, as described above.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example shows how an instance-specific cell is power-characterized using the **set_cell_internal_power** command:

```
prompt> set_cell_internal_power [get_pins U123/A] 1.234 uW
prompt> set_cell_internal_power [get_pins U123/B] 1.567 uW
prompt> set_cell_internal_power [get_pins U123/*] -1.000 nW
prompt> set_cell_internal_power -delete_all
```

SEE ALSO

`report_power(2)`

set_cell_location

Specifies the physical location for leaf cells.

SYNTAX

```
int set_cell_location  
object_list  
-coordinates {X Y}  
[-ignore_fixed]
```

ARGUMENTS

object_list
Specifies the list of cells object, each of which is a leaf cell. Typically, this command uses a single cell as the argument.

-coordinates {X Y}
Specifies the lower left coordinates of the specified cell(s). The numbers are in microns relative to the chip origin.

DESCRIPTION

The **set_cell_location** command specifies the physical location for leaf cells. It overwrites the existing information, but not the CLUSTER move_bounds.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets the location of the cell named INST_1 to the coordinates {100 100}.

```
prompt> set_cell_location -coordinates {100 100} INST_1
```

SEE ALSO

```
create_placement(2)  
set_dont_touch_placement(2)  
write_physical_constraints(2)
```

set_cell_row_type

Binds the logical hierarchical module to the specified row type attribute.

SYNTAX

```
int set_cell_row_type  
-type row_type  
cell_list
```

Data Types

```
row_type      int  
cell_list     list
```

ARGUMENTS

```
-type row_type  
    Specifies the row type identifier that is set on hierarchical cells.  
  
cell_list  
    Specifies one or more submodules in the design.
```

DESCRIPTION

The **set_cell_row_type** command sets a row type for a specified list of cells. The cells must be logical hierarchies and should not already have a row type attribute set.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows how to set the row type identifier to 1 for the cells "I_ALU" and "I_CONTROL".

```
prompt> set_cell_row_type -type 1 I_ALU I_CONTROL
```

SEE ALSO

set_row_type(2)

set_cell_type

Changes the **cell_type** attribute on any library reference cell so you can set or unset any cell as a macro cell.

SYNTAX

```
status set_cell_type  
-type BLOCK | COVER | RING | PAD | CORE  
list_of_library_reference_cells
```

Data Types

```
list_of_library_reference_cells      string
```

ARGUMENTS

```
-type BLOCK | COVER | RING | PAD | CORE  
    Specifies the cell_type attribute.
```

```
list_of_library_reference_cells  
    Specifies the list of library reference cells on which to change the cell_type  
    attribute.
```

DESCRIPTION

This command changes the **cell_type** attribute on the specified library reference cells. You specify the library reference cells in the input list, and the command changes the **cell_type** attribute on each one. Setting the library reference **cell_type** attribute to one of the valid types gives you the control to set or unset any cell as a macro cell. Valid cell types are BLOCK, COVER, RING, PAD, and CORE.

After you change any library reference cell to one of the valid cell types, all netlist cells instantiated from these library cells are treated like the specified cell type.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets the cell type of the *libram512* library reference cell to **block**.

```
prompt> set_cell_type -type BLOCK libram512
```

Note that every library cell already has a *cell_type* attribute and these attributes are overwritten in the current session after running this

command.

You can also specify the `cell_type` as **CORE** or **PAD** to remove the macro cell attributes.

A **cell_type** attribute set by using the **set_cell_type** command is not a permanent attribute set on library cells.

These attributes last only for the current session, unless you invoke a **write_lib** command in the current session.

After invoking a **write_lib** command, these attributes are written to a new library and then are persistent.

SEE ALSO

`all_macro_cells(2)`

`write_lib(2)`

set_cell_vt_type

Sets the threshold voltage type of a library cell or of all cells of a library. The threshold voltage type is used for mixed threshold voltage filler cell insertion.

SYNTAX

```
status set_cell_vt_type  
-library library_name | -lib_cell cell_name  
-vt_type vt_type_name
```

Data Types

<i>library_name</i>	string
<i>cell_name</i>	string
<i>vt_type_name</i>	string

ARGUMENTS

Use this option to set the specified library cell to be the indicated vt type.

```
-library library_name | -lib_cell cell_name  
Use -library to specify a library name or -lib_cell to specify a cell name.
```

```
-vt_type vt_type_name  
Specifies the name of the vt type to be set to the library cell or the cells  
of a library
```

DESCRIPTION

Use this command to set vt type string of a library cell or cells of a library.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets vt type VtType1 to library cell MasterA

```
prompt> set_cell_vt_type -lib_cell "MasterA" -vt_type "VtType1"
```

SEE ALSO

```
insert_stdcell_filler(2)  
set_vt_filler_rule(2)  
remove_cell_vt_type(2)  
report_cell_vt_type(2)
```


set_check_library_options

Sets specific options for the **check_library** command for logic library versus logic library checking, logic library versus physical library checking, and for checks between physical libraries.

SYNTAX

```
status set_check_library_options
[-cell_area]
[-cell_footprint]
[-bus_delimiter]
[-tech_consistency]
[-view_comparison]
[-same_name_cell]
[-signal_em]
[-antenna]
[-rectilinear_cell]
[-physical_only_cell]
[-phys_property {property_list}]
[-routeability]
[-tech]
[-drc]
[-scaling {scaling_types}]
[-mcmm]
[-upf]
[-compare {construct | attribute | value}]
[-tolerance {type relative_tolerance absolute_tolerance}]
[-leq]
[-physical]
[-logic_vs_physical]
[-logic]
[-reset]
[-all]
[-analyze {nominal_vs_sigma table_trend}]
[-validate {timing}]
[-group_attribute {group_name/attribute_name}]
[-criteria {std_error=val_err slope=val_sl trend=['/' | '' | '^' | 'V']}]
[-report_format {csv[=csv_dir] nosplit sort_by_cell | sort_by_group_type}]
```

Data Types

<i>property_list</i>	list
<i>scaling_types</i>	list
<i>type</i>	string
<i>relative_tolerance</i>	float
<i>absolute_tolerance</i>	float

ARGUMENTS

-cell_area
Compares the area of the cells in the logic library (set by the area attribute) against the area set by the cell PRBoundary in the physical

library. In the FRAM view, the ratios of the PRBoundary for standard cells in a rectangle, or the ratios of the CellBoundary for macros in a polygon, are compared against the cell area in the .db file. If the ratio is the same for all the cells in the library, the cell areas are considered to be consistent. If the ratio for a cell deviates from the normal or average ratio for this library by a margin of 5 percent, it is counted as an inconsistent area.

There is no area check for pad cells. A pad cell is a special cell at the chip boundary in a logic cell. Pad cells are not checked because they are not used as internal gates and always have an area of 0.

The default logic versus physical checks are also enabled with this option.

-cell_footprint

Checks that the cell PR boundary in the physical library is consistent among a class of cells when the same **cell_footprint** attribute is specified in the logic library. This option reports cell names and their PR boundaries grouped by the **cell_footprint** string.

The default logic versus physical checks are also enabled with this option.

-bus_delimiter

Reports bus delimiters in logic and physical libraries. If there is no bus delimiter in the library, the field is blank in the report.

The default logic versus physical checks are also enabled with this option.

-tech_consistency

Checks for technology data consistency between the main library or design library and each associated reference library. The option checks the physical libraries for missing layer data and mismatched technology data.

-view_comparison

Checks for the existence of CEL and FRAM views in the library and for mismatched time stamps between these views. In the report table for missing views, the CEL and FRAM columns list the cell name and version number. An X marks the view that is missing.

In the report table for mismatched views, the CEL version and FRAM version columns list the version numbers, and the CEL and FRAM modified time column lists the time when the view was last modified. If a cell has an earlier FRAM view than its CEL view, it is marked as mismatched.

The time that is checked is the internal view creation or modification time in the Milkyway database, not the UNIX time. No mismatch check is performed on the cell content. If you read FRAM views from the LEF and then stream in CEL views, the views are shown as mismatched. You can ignore this report in this case.

-same_name_cell

Checks for cells with identical names among the specified main or design library and all physical reference libraries linked to the specified library. If there are cells with the same name in multiple reference libraries, the tool uses the first cell in the reference control file and ignores the remaining cells. This option does not check naming among the logic libraries.

-signal_em

Checks for the signal electromigration rule (current model and model type) for each routing layer.

-antenna

Checks for missing antenna properties for cells and antenna rules in the layers in the specified main library.

In the missing antenna property table, it lists the cell names and pin names that are missing the antenna property and the missing property type. If an input pin is missing the gate size, it is counted as missing the property. If an output pin is missing diode protection, it is counted as missing the property. If a macro is missing the hierarchical antenna property, it is counted as missing the property. You should specify hierarchical antenna properties for macros.

This option reports the mode, diode mode, default metal ratio, default cut ratio and maximum ratio for each antenna rule.

-rectilinear_cell

Checks and reports the cell names, types, and coordinates of cells with rectilinear boundaries.

-physical_only_cell

Checks and reports the cell names, types, and properties of physical-only cells.

Physical-only cells are filler cells with and without metal, corner pad cells, tap cells, chip cells, I/O pad cells, cover cells and cells that only have power and ground pins.

-phys_property {property_list}

Specifies a list of physical properties to check. You can specify one or more of the following values:

- **place** reports the following placement properties for each standard cell:
 - * PR boundary as represented by its lower-left and upper-right coordinates.
 - * Cell height relative to the unit tile height, such as 1xH for single height.
 - * Coordinate, the bottom-left location for a single-height cell or left locations at each unit height for a multiple-height cell.
 - * Tile pattern representing possible cell orientations that can have combinations of R0, R90, R160, R0_MX, R0_MY, 90_MX, and R90_MY.
 - * Remarks noting that the PR boundary mismatches the tile pattern. If mismatched, use the **cmSetMultiHeightProperty** command to set the tile patterns for multiple-height cells.
 - * For macros, the option reports the cell boundary and height.

When you specify the **place** argument, it also lists the main library and its reference library names with paths, if any, and the unit tiles names and sizes. The unit tile size is reported in the format of width x height. If no tile size is reported, the library has no unit tile and the unit tile in the main library is used in the design.

- **route** checks and reports the routing properties for each routing layer including the preferred direction, track direction, offset, pitch, and remarks. The remarks column shows OK if the offset is 0 or half pitch; otherwise, it marks pitch=0 or offset!=0.5*pitch.

- **cell** reports the cell and pin properties for each cell including pin name, direction, and type. It also reports the number of cells with multiple power and ground pins.

- **metal_density** checks the macro metal density data for the cells specified

by the **check_library** command with the **-cells** option. The option checks if a cell has at least one of the following errors:

- * The metal density data in the CEL view and FRAM view are inconsistent.
- * The metal density windows do not cover the entire area on a layer.

The option lists all of the cells with metal density errors in a table with columns showing the cell name, data inconsistency between the CEL view and FRAM view, and on which layer the density windows do not cover the entire area.

-routeability

Checks the physical pin on-track accessibility and quality of defined wire tracks. The option reports the total number of pins without optimal on-track routability and lists the pin names, directions, layers, and tracks for each cell with this issue.

In the track column, an H denotes that the pin is not accessible on a horizontal track, a V denotes the pin is not accessible on a vertical track, and H&V denotes that the pin is not accessible on both horizontal and vertical tracks.

If a pin is reported as not accessible on-track, the pin might be routed by an off-track wire during detail routing. If too many pins do not have any on-track routability, adjust the offset values of the wire track (0 or half pitch is preferred) and rerun the **create_lib_track** command to reduce the number of pins without optimal accessibility.

If routability checking is not successful, the cause might be no unit tile or incorrect technology data, such as an illegal fat wire threshold for some layers, illegal vias or cut layers, an illegal layer number, and undefined rules.

It is possible that a track intersection is inside a via region but there is not optimal routability. This can happen because the tool considers the worst case when checking the routability, for example if a via is already dropped in a neighboring pin's via region, making the via spot that is from the via region of the pin being checked smaller.

-tech

Checks for the technology data in the specified library. This check is different from the checking done by **-tech_consistency** in that this option checks the technology data in the single library. The messages of the checking are similar to those during library creation or technology data replacement. This option requires that the directory where the library resides has write permission. If the library is not writable, it is copied to the local library first, and then the local library is checked.

-drc

Specifies DRC checks for the routing (FRAM) view of cells in the specified library. It checks the following design rules:

```
wire minWidth
minEdgeLength
minEnclosedArea
minSpacing
cutWidth
cutSpacing
minEnclosure
sameNetMinSpacing
maxStackLevel
```

fatTblSpacing

metal-via spacing (minSpacing in DesignRule)

It lists the cells with DRC violations in a table with columns showing the cell name, type, and error cell. Check the error cells for details of DRC violations. This option requires that the directory where the library resides has write permission.

-scaling {*scaling_types*}

Specifies a list of logic library consistency checks for various types of scaling.

You can specify one or more of the following values:

- **timing** specifies logic library consistency checking for CCS timing scaling.
- **noise** specifies logic library consistency checking for CCS noise scaling.
- **power** specifies logic library consistency checking for CCS or NLDM power scaling (driver and receiver models, load indices, base curve_x, waveform, noise, and power models.)

-mcmm

Specifies logic library consistency checking for multicorners-multimode (power_down_function and power data).

-upf

Specifies logic library consistency checking for multivoltage and UPF (power special cells, pg_pin, voltage_names, power_down_function, and power data).

-compare {construct | attribute | value}

Compares all groups, subgroups, and attributes between the logic libraries. You can specify one of the following values:

- **-compare {construct}** checks if constructs or attributes are missing.
- **-compare {attribute}** checks if constructs and attributes are missing and if attribute values are inconsistent. It checks all groups and subgroups and all attribute values except characterization values. It is a superset of **-compare {construct}**.
- **-compare {value}** compares values of each group and attributes between the libraries. It is a superset of the **-compare {attribute}**. In comparing characterization values, if the source is not from .lib files, the values are not reported, such as values in vectors and capacitances. The comparison of values are controlled by tolerances specified by **-tolerance** or by default values if **-tolerance** is unspecified. For minimum and maximum library checking and scaling group library checking, you should specify **-compare {attribute}** to check all library contents except characterizations. When you specify this option, the following default checks are also performed: missing cells, missing and mismatched pins, and timing arcs.

-tolerance {*type relative_tolerance absolute_tolerance*}

Specifies a relative tolerance and an absolute tolerance for characterization value comparison, such as delay values. The format is

-tolerance {*type rel_tol abs_tol*}

where the valid values for the *type* argument are delay, slew, constraint, slew_index, load_index, time, power, current, energy and capacitance. If the type is not specified, the values are for output load capacitance. To specify an absolute tolerance, do not include the unit. The unit in the first library is used. For example, to specify tolerances for delay and slew, use:

-tolerance {delay 0.1 0.2 slew 0.3 0.4}

If you do not specify tolerances for a type, default values are used. The default values are set as follows in compliance with PrimeTime and the Library Quality Assurance System:

Relative tolerance for load index	: 0.01
Absolute tolerance for load index	: 0.001pF
Relative tolerance for time	: 0.04
Absolute tolerance for time	: 0.015ns
Relative tolerance for power	: 0.04
Absolute tolerance for power	: 5pW

To specify monotonous decreasing (\), use {trend = ''}. In the expressions of std_error and slope, the sign can also be <. or =. For trend, inequality can be specified by !=. The VA analysis reports the results that meet any of the specified criteria (expressions).

-leq

Checks logical equivalence for *when* conditions. For example, if you set the **-leq** option, the following expressions are reported as identical:

when : A+B
when : B+A

By default, logical equivalence is disabled and string comparisons are used.

-physical

Includes the following physical checking options:

- tech_consistency
- view_comparison
- same_name_cell
- signal_em
- antenna
- rectilinear_cell
- physical_only_cell
- phys_property {place route}
- routeability
- tech
- drc

-logic_vs_physical

Checks all logic versus physical library checking including the default checks for missing cells and missing or mismatched pins in the logic or physical library and the following checks:

- cell_area
- cell_footprint
- bus_delimiter

-logic

Includes all of the following logic library checking options:

-scaling

-mcm

-upf

-reset

Resets the options to the default, meaning that missing cells and missing or mismatched pins are checked between logic and physical libraries. All other options are ignored if used with the **-reset** option.

-all

Checks all of the following for the library:

-tech_consistency

-view_comparison

-same_name_cell

-signal_em

-antenna

-rectilinear_cell

-physical_only_cell

-phys_property {place route cell metal_density}

-routeability

-tech

-drc

-logic_vs_physical

-logic

-analyze {*nominal_vs_sigma table_trend*}

Specifies what is analyzed. You can specify one of the following values:

- **nominal_vs_sigma** analyzes multiple characterization tables. In the case of variation-aware models, you usually analyze nominal, -sigma, and +sigma tables to see the trend and standard deviation at each slew and load index grid on a linear model by *va_parameters*.

Specify the *va_parameters* in the **-group_attribute** option. If you do not specify the *va_parameters*, the variation-aware models for all *va_parameters* are analyzed. Wildcards apply to *va_parameters*.

- **table_trend** analyzes a single characterization table. It analyzes the trend within a single table to see if the values change monotonously as either slew or load indices.

-validate {timing}

Specifies logic library consistency checking for library characterization between different timing models.

-group_attribute {*group_name/attribute_name*}

Checks or compares specific groups and attributes only.

You can specify one or more groups or attributes. Each item can be a group only, an attribute only, a group and an attribute together with a value, or a group and an attribute together without a value. Use a space to separate each item in the list.

Use a forward slash (/) to combine a group name and attribute name that you

are specifying together. For example, {pin/direction} combines the pin group and the direction attribute under the pin group. In this case, the command checks the direction in the pin group only.

Use curly braces ({}) or double quotation marks ("") to enclose an expression delimited by spaces. The following example checks only the pin group whose name is A:

-group/attribute {"pin = A"}

The following wildcard characters are available for pattern matching:

- An asterisk (*) matches any string, including a null string.

- A question mark (?) matches any single character.

For example:

- **-group/attribute {pin}** checks the pin group only.

- **-group/attribute {direction}** checks the direction attribute only.

- **-group/attribute {pin/direction}** checks the pin's direction only.

- **-group/attribute {pin/direction=input}** checks the pin direction only when it is input.

- **-group/attribute {cell_rise cell_fall output_current_*}** checks and validates NLDM and CCS timing models only.

- **-group/attribute {va_* compact_ccs_*}** checks and validates variation-aware timing models only.

If you do not specify the **-group/attribute** option, the **check_library** command checks all groups and attributes that apply to the specified mode.

-criteria {std_error=val_err slope=val_sl trend=['/' | '\' | '^' | 'V']}

Specifies analysis criteria for variation analysis (multiple tables) or non-variation-analysis (single table) in the following format: {std_error>m.m slope>n.n trend=trend_symbol}. The std_error and slope are for linear regression models and the trend_symbol includes the following eight symbols:

/ monotonously increasing

\ monotonously decreasing

^ non-monotonous up

V non-monotonous down

- flat

M Multiple peaks

W Multiple troughs

N >=2 peaks with 2nd not turning low

To specify monotonous decreasing (\), please use {trend = '\'} or {trend = \\} or "trend = \\", not {trend = \}. In the expressions of std_error and slope, the sign can also be <. or =. For trend, inequality can be specified by !=. The analysis reports the results that meet any of the specified criteria (expressions).

This option is used with the **-analyze** option.

`-report_format {csv[=csv_file_dir] nosplit`

`sort_by_cell | sort_by_group_type}`

Specifies the format for the generated report. The syntax of the specification is as follows:

`[csv[=csv_file_dir]] [nosplit] [sort_by_cell | sort_by_group_type]`

You must specify at least one of the following values in the specification:

- **csv[=*csv_file_dir*]**

Selects CSV as the report format. By default, the CSV files are stored in the current working directory. Specify the directory to store the CSV files using the *csv_file_dir* argument.

If you perform multiple runs with the same setting, the directory immediately prior to the current run is backed up to a directory named *csv_file_dir_bak*. If you do not specify this value, the **check_library** command generates an ASCII report.

- **nosplit**

Prevents prevents line-splitting and facilitates writing software to extract information from the report output. Most of the information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

- **sort_by_cell | sort_by_group_type"**

Specifies the sorting method. Use **sort_by_cell** to sort the validation and analysis tables by cell names. Use **sort_by_group_type** to sort the validation and analysis tables by group types, such as delay, slew, constraints, and receiver capacitance.

By default, the validation and analysis tables are sorted by group type.

If you do not specify this option, the **check_library** command generates an ASCII report that is sorted by group type.

DESCRIPTION

The **set_check_library_options** command sets specific options for the **check_library** command for logic library versus logic library checking, logic library versus physical library checking, and for checks between physical libraries.

Run the **set_check_library_options** command before running **check_library**. If you do not specify options with **set_check_library_options**, by default the command checks missing cells and pins and mismatched pins, including *pg_pins* versus power and ground pins.

A status indicating success or failure is returned.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

In the following example, the **-cell_footprint** and **-bus_delimiter** options are set, the lib1.db and lib2.db logic libraries are checked against the phys_lib physical libraries for missing cells and pins, mismatched pins, the cell footprints are compared, and the bus delimiters are checked:

```
prompt> set_check_library_options -cell_footprint \
        -bus_delimiter
1
prompt> check_library -mw_library_name {phys_lib} \
        -logic_library_name {lib1.db lib2.db}
1

#BEGIN_XCHECK_LIBRARY

Logic Library:      lib1.db
                   lib2.db
Physical Library:   phys_lib
check_library options: -cell_footprint -bus_delimiter
Version:            A-2007.12
Check date and time: Thu Jul 26 16:59:54 2007

#BEGIN_XCHECK_LOGICCELLS

Number of cells missing in logic library:      3 (out of 3348)

                List of cells missing in logic library
-----
Cell name                Cell type                Physical library
-----
AND2                     Core                     phys_lib
NOR1                     Core                     phys_lib
XOR3                     Core                     phys_lib
-----

                List of physical only cells
-----
Cell name                Cell type                Physical library
-----
GFILL                    Filler                    phys_lib
GFILL10                  Filler                    phys_lib
FILL8                    Filler                    phys_lib
-----

#END_XCHECK_LOGICCELLS

#BEGIN_XCHECK_PHYSICALCELLS

Number of cells missing in physical library:    0 (out of 846)

#END_XCHECK_PHYSICALCELLS

#BEGIN_XCHECK_PINS
```

Number of cells with missing or mismatched pins in libraries: 0

#END_XCHECK_PINS

#BEGIN_XCHECK_BUS

List of bus naming styles

Library name	Library type	Bus naming style
phys_lib	Physical library	_%d>
lib1.db	Logic library	

#END_XCHECK_BUS

#BEGIN_XCHECK_FOOTPRINT

Number of footprints: 1

List of cells with cell_footprint attribute

Footprint	Logic library name	Cell name	PR boundary
TIEH	lib1.db	GTIEH	(0,0) (0.8,1.8)
	lib1.db	TIEH	(0,0) (0.6,1.8)

#END_XCHECK_FOOTPRINT

Cross check summary:

Number of cells missing in logic library: 3
Logic library is INCONSISTENT with physical library.

#END_XCHECK_LIBRARY

1

SEE ALSO

check_library(2)
report_check_library_options(2)

set_checkpoint_strategy

Set the checkpoint strategy for optimization commands

SYNTAX

```
int set_checkpoint_strategy  
[-enable | -disable]  
[-overwrite]  
[-prefix prefix]
```

ARGUMENTS

`[-enable | -disable]`

These exclusive options will either enable checkpoint creation or disable checkpoint creation during optimization. Checkpoint creation is disabled by default.

`-overwrite`

If this option is used, then optimization will use the same design name during checkpoint creation. By default, optimization will save each checkpoint design with a different design name. See Section **Description** for more details of the design names.

`-prefix string`

This option causes optimization to add a prefix to the design names being saved.

DESCRIPTION

By enabling checkpointing using the `set_checkpoint_strategy` command, optimization will save the design (snapshot) at periodic intervals. This allows the user to examine the intermediate results while optimization is still proceeding.

By default, each checkpoint design will be given a different design name (cell name). The design names (cell name) are of the form:

```
<command>_checkpoint_<design_name>_<timestamp>_<number>
```

where `<command>` is "place_opt", "clock_opt" or "route_opt", `<design_name>` is the name of the design (the internal name seen by the tool), `<timestamp>` is the timestamp when `<command>` is run, and `<number>` is a sequential number starting from "1" and is incremented each time a checkpoint is created.

Since a user may want to make multiple runs, the timestamps allow a user to know which designs relate to which run. The timestamp is of the form `yyymmdd_hhmmss`.

If the `-prefix` option is used, then the design names are of the form:

```
<prefix>_<command>_checkpoint_<design_name>_<timestamp>_<number>
```

If the `-overwrite` option is used, then the design name becomes

<command>_single_checkpoint_<design_name>_<timestamp>_<number>

where the same name is being used, unless it is currently being accessed.

Each time a checkpoint is to be created, the tool will generate a checkpoint design name, and check if the design is currently being used (created from previous runs). If the design is not currently being used, then the tool will use the same name and replace the old design with the new checkpoint design. If the design is currently being used, then the tool will NOT overwrite the design, but will generate a new checkpoint design name.

EXAMPLES

We use the place_opt command as our examples below.

Suppose, prior to place_opt command, we run the command :

```
prompt> set_checkpoint_strategy -enable
```

Then, during place_opt, the tool creates the following checkpoint designs:

```
place_opt_checkpoint_MYDESIGN_080901_120402_1
place_opt_checkpoint_MYDESIGN_080901_120402_2
place_opt_checkpoint_MYDESIGN_080901_120402_3
..., etc.
```

where MYDESIGN is the name of the design, and the timestamp is Sept 1st, 2008, at time 12:04:02.

If, during this run, some of the designs are being accessed, then the next available number will be used. For example, if place_opt_checkpoint_MYDESIGN_080901_120402_2 is currently being accessed, then the checkpoint designs created will be:

```
place_opt_checkpoint_MYDESIGN_080901_120402_1
place_opt_checkpoint_MYDESIGN_080901_120402_3
place_opt_checkpoint_MYDESIGN_080901_120402_4
..., etc.
```

If the user instead had used:

```
prompt> set_checkpoint_strategy -enable -overwrite
```

then the checkpoint designs will be:

```
place_opt_checkpoint_MYDESIGN_080901_120402_1
```

Only this name is used, unless this design is being accessed when the checkpoint is being created, then the next available name will be used:

```
place_opt_checkpoint_MYDESIGN_080901_120402_2
```

If the user had used:

```
prompt> set_checkpoint_strategy -enable -prefix myrun
```

then the checkpoints created will be:

```
myrun_place_opt_checkpoint_MYDESIGN_080901_120402_1  
myrun_place_opt_checkpoint_MYDESIGN_080901_120402_2  
myrun_place_opt_checkpoint_MYDESIGN_080901_120402_3  
..., etc.
```

The user can also use the command `remove_checkpoint_designs` to remove the checkpoint designs created.

SEE ALSO

```
place_opt(2)  
clock_opt(2)  
route_opt(2)  
report_checkpoint_designs(2)  
remove_checkpoint_designs(2)
```

set_child_terminal

Sets attributes on the child terminals associated with the specified pin shapes.

SYNTAX

```
status set_child_terminal
objects
[-access_direction {directions}]
[-fix | -unfix]
[-layer layer]
[-boundary | -rectangles {rect_list}]
[-verbose]
```

Data Types

<i>objects</i>	collection
<i>directions</i>	list
<i>layer</i>	string
<i>polygon</i>	list
<i>rect_list</i>	list

ARGUMENTS

objects

Specifies the pin shapes in the top-level design that are associated with the child terminals on which to set the attributes.

-access_direction {*directions*}

Sets the **access_direction** attribute to the specified values.
You can specify one or more of the following values:

- **Left** indicates the object can be accessed from left.
- **Right** indicates the object can be accessed from right.
- **Up** indicates the object can be accessed from above.
- **Down** indicates the object can be accessed from below.

-fix

Sets the **is_fixed** attribute to true.
The **-fix** and **-unfix** options are mutually exclusive; you can specify only one.

-unfix

Sets the **is_fixed** attribute to false.
The **-fix** and **-unfix** options are mutually exclusive; you can specify only one.

-layer *layer*

Sets the **layer** attribute to the specified value. You can specify only a single layer, either as a collection of one object or as a layer name.

-boundary *{polygon*

Sets the **points** attribute to the specified polygon. The valid format of a polygon is {{x1 y1} {x2 y2} ... {xN yN} {x1 y1}}, where each {x y} pair specifies one point of the input polygon. The coordinate unit is specified in the technology file.

If you specify this option, the shape of the child terminals are changed to a rectilinear polygon.

The **-boundary** and **-rectangles** options are mutually exclusive; you can specify only one.

-rectangles *{rect_list}*

Sets the **bbox** when you specify one rectangle or the **points** attribute when you specify more than one rectangle. The argument format can be {llx1 lly1 urx1 ury1 ...} or {{llx1 lly1 urx1 ury1} ...}. Each set of {llx lly urx ury} defines a rectangular area. The coordinate unit is specified in the technology file.

If you specify this option, the shape of the child terminals are changed to one rectangle or to a list of rectangular shapes.

The **-boundary** and **-rectangles** options are mutually exclusive; you can specify only one.

-verbose

Enables the printing of additional debugging messages.

DESCRIPTION

This command sets the specified attributes on the child terminals associated with the specified pin shapes.

This command implicitly opens the specified child design in write mode as a side effect.

You need to run the **save_mw_cel -hierarchy** command before you close the top-level design.

The command returns a status value indicating success or failure.

EXAMPLES

The following example marks the clk terminal in the child design as fixed by specifying its corresponding pin shape, I_ORCA_TOP/I_BLENDER_4/clk, and then checks the attribute on the child terminal.

```
prompt> set_child_terminal -fix I_ORCA_TOP/I_BLENDER_4/clk
1
prompt> current_mw_cel BLENDER_2
{BLENDER_2}
prompt> get_attribute -class terminal clk is_fixed
true
```

The following example selects a pin shape, changes its bounding box, layer, and access direction, and then checks the changed attributes on the pin shape.


```

prompt> set ps1 [get_pin_shape I_ORCA_TOP/I_BLENDER_1/result[30]]
{I_ORCA_TOP/I_BLENDER_1/result[30]}
prompt> set layer [get_layer METAL2]
{METAL2}
prompt> set_child_terminal $ps1 \
-boundary {{1190 877} {1190.2 877} {1190.2 877.2} {1190 877.2} {1190 877}} \
-layer $layer -access_direction {Up} -verbose
Information: Succeeded in setting access_direction on result[30]. (MWUI-068)
Information: Succeeded in setting layer on result[30]. (MWUI-068)
Information: Succeeded in setting points on result[30]. (MWUI-068)
1
prompt> get_attribute $ps1 bbox
{1190.000 877.000} {1190.200 877.200}
prompt> get_attribute $ps1 layer
METAL2
prompt> get_attribute $ps1 access_direction
Up

```

The following example shows that you cannot set the attributes on a terminal of a design that is not writable, but you can change the **layer** and **points** attributes of the associated pin shape.

```

prompt> set two_ps \
[get_pin_shapes {U26/I "I_ORCA_TOP/I_BLENDER_2/VSS 72"}]
{U26/I I_ORCA_TOP/I_BLENDER_2/VSS 72}
prompt> set_child_terminal $two_ps -layer METAL2 \
-rectangles {1200 850 1201 851 1201 850 1203 852} -verbose
Warning: U26/I is skipped because its design is not writable. (MWUI-073)
Information: Succeeded in setting layer on VSS 72. (MWUI-068)
Information: Succeeded in setting points on VSS 72. (MWUI-068)
1
prompt> get_attribute \
[get_pin_shapes {"I_ORCA_TOP/I_BLENDER_2/VSS 72"}] points
{1200.000 850.000} {1201.000 850.000} {1201.000 851.000} {1200.000 851.000}
{1200.000 850.000} {1201.000 850.000} {1203.000 850.000} {1203.000 852.000}
{1201.000852.000} {1201.000 850.000}

```

SEE ALSO

```

get_attribute(2)
get_mw_cels(2)
get_pin_shapes(2)
save_mw_cel(2)

```

set_chiplevel_pad_physical_constraints

Set chiplevel physical constraints on PAD cells.

SYNTAX

```
status set_chiplevel_pad_physical_constraints  
[-dist_left_edge_to_pad float]  
[-dist_top_edge_to_pad float]  
[-dist_right_edge_to_pad float]  
[-dist_bottom_edge_to_pad float]
```

ARGUMENTS

[-dist_left_edge_to_pad float]
On the top and bottom die edges, specify minimum distance in microns pads must be placed away from the left die edge. Specified value must be positive floating point type. Default value is 0.

[-dist_top_edge_to_pad float]
On the left and right die edges, specify minimum distance in microns pads must be placed away from the top die edge. Specified value must be positive floating point type. Default value is 0.

[-dist_right_edge_to_pad float]
On the top and bottom die edges, specify minimum distance in microns pads must be placed away from the right die edge. Specified value must be positive floating point type. Default value is 0.

[-dist_bottom_edge_to_pad float]
On the left and right die edges, specify minimum distance in microns pads must be placed away from the bottom die edge. Specified value must be positive floating point type. Default value is 0.

DESCRIPTION

This command allows users to specify minimum distance (in microns) the pads must back-off from the implied die corner. When the design is saved, these constraints are also saved to Milkyway.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows how to specify the minimum distance for pads.

```
prompt> set_chiplevel_pad_physical_constraints -dist_left_edge_to_pad 50 -  
dist_top_edge_to_pad 50 -dist_right_edge_to_pad 50 -dist_bottom_edge_to_pad 50
```

SEE ALSO

```
remove_io_constraints(2)  
set_pad_physical_constraints(2)  
set_pin_physical_constraints(2)  
write_io_constraints(2)
```

set_clock_gating_check

Puts setup and hold checks on clock gating cells.

SYNTAX

```
int set_clock_gating_check  
[-setup setup_margin]  
[-hold hold_margin]  
[-rise]  
[-fall]  
[-high | -low]  
[object_list]
```

Data Types

<i>setup_margin</i>	float
<i>hold_margin</i>	float
<i>object_list</i>	list

ARGUMENTS

-setup *setup_margin*

Specifies the setup margin for the clock gating signal. For AND and NAND clock gating elements, the setup time is checked relative to the 0->1 transition of the clock input. For OR and NOR gating elements, the setup time is checked against the 1->0 transition of the clock input. For more complicated gating elements, the setup check will be against the clock transition where the clock input goes from a controlling value to a noncontrolling value for the clock gating element.

-hold *hold_margin*

Specifies the hold margin for the clock gating signal. For AND and NAND clock gating elements, the hold time is checked relative to the 1->0 transition of the clock input. For OR and NOR gating elements, the hold time is checked against the 0->1 transition of the clock input. For more complicated gating elements, the hold check will be against the clock transition where the clock input goes from a noncontrolling value to a controlling value for the clock gating element.

-rise

Specifies if rising delays are constrained for clock-gating checks. If you do not specify the **-rise** or **-fall** option, both rising and falling delays are constrained.

-fall

Specifies if falling delays are constrained for clock-gating checks. If you do not specify the **-rise** or **-fall** option, both rising and falling delays are constrained.

-high

Indicates that the check is to be performed on the high level of the clock. By default, timing analysis determines whether to use the high or low level

of the clock using information from the cell's logic. That is, for AND and NAND gates, the check is performed on the high level; for OR and NOR gates, on the low level. For some complex cells (for example, MUX, OR-AND) it cannot be determined which level to use, and thus no check is performed unless either **-high** or **-low** is specified. This option can be used only when the object list contains cells or pins.

-low

Indicates that the check is to be performed on the low level of the clock. By default, timing analysis determines whether to use the high or low level of the clock using information from the cell's logic. That is, for AND and NAND gates, the check is performed on the high level; for OR and NOR gates, on the low level. For some complex cells (for example, MUX, OR-AND) it cannot be determined which level to use, and thus no check is performed unless either **-high** or **-low** is specified. This option can be used only when the object list contains cells or pins.

object_list

Specifies a list of cells, pins, or clock objects where the clock gating setup or hold margins are to be checked. By default, if *object_list* is not specified, the clock gating check is applied to the *current_design*.

It is possible to enable clock gating checks only for selected clock gating cells by specifying the names of the cells in the object list. Clock gating checks for specific cells override any clock gating checks that may have been specified for the design.

Setting clock gating checks on pins can enable clock gating checks only for specific pins.

Setting clock gating checks on clock objects enables checks for all clock gating cells driven by the clock.

It is possible for the user to specify which interval of the clock should the clock gating checks be made by using the **-high** and **-low** option.

DESCRIPTION

The **set_clock_gating_check** command provides the ability to check setup or hold margins for control inputs of clock gating cells. The setup check ensures that the clock gating input stabilizes for a given amount of time before the clock input of the gating cell makes a transition to a noncontrolling value. The hold check ensures that the clock gating signal stabilizes for a given period of time after the clock input has gone back to a controlling value. Together, the two checks ensure that the clock gating signal stabilizes for the entire period of time when the gated clock input has a noncontrolling value. This is so that the gating signal can not "clip" a clock edge or generate spurious clock pulses by changing when the clock input is noncontrolling.

The clock gating checks created by **set_clock_gating_check** act as constraints for the **compile** command, which will try to adjust the delays of the logic driving the clock gating inputs in order to avoid setup or hold violations.

Be aware that the presence of clock gating checks on a clock gating cell disables certain logic transformations that otherwise might be applied to the cell. The **compile** command can adjust only the size of the cell by replacing it with other cells with the same logic function but with different drive capacity. No other logic transformations involving the clock gating cell is permitted.

The **set_clock_gating_check** command can only create setup and hold checks against a clock signal. The command cannot introduce setup or hold checks between two nonclock signals.

To undo **set_clock_gating_check**, use **remove_clock_gating_check**.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example creates setup checks with a 0.75 margin and hold checks with a 0.5 margin for all clock gating inputs in current design.

```
prompt> set_clock_gating_check -setup 0.75 -hold 0.5
```

The following example creates setup checks with a 0.58 rise margin for all cells driven by the clock CLK1.

```
prompt> set_clock_gating_check -setup 0.75 -hold 0.5 \  
      [get_clocks CLK1]
```

The following example creates only a hold check with margin 0.64 on all gating inputs of cell CLK_GATE1.

```
prompt> set_clock_gating_check -hold 0.64 CLK_GATE1
```

The following example creates a setup check with margin 0.44 and a hold check with margin 0.64 on all gating inputs of cell CLK_GATE1, and specifies that the clock is non-controlling when it is low.

```
prompt> set_clock_gating_check -hold 0.64 -low CLK_GATE1
```

SEE ALSO

```
create_clock(2)  
current_design(2)  
remove_clock_gating_check(2)  
report_clock(2)
```

set_clock_gating_registers

Forces the enabling or disabling of clock gating for specified registers in the current design, overriding all conditions necessary for automatic RTL clock gating by the **compile_ultra -gate_clock** command.

SYNTAX

```
status set_clock_gating_registers  
[-include_instances register_list]  
[-exclude_instances register_list]  
[-undo register_list]
```

Data Types

register_list list

ARGUMENTS

- include_instances *register_list***
Specifies a list of registers in the current design to be included in clock gating. A register cannot be on more than one of the **-include_instances**, **-exclude_instances**, and **-undo** lists.
- exclude_instances *register_list***
Specifies a list of registers in the current design to be excluded from clock gating. A register cannot be on more than one of the **-include_instances**, **-exclude_instances**, and **-undo** lists.
- undo *register_list***
Specifies a list of registers in the current design for which the previous specification to the **set_clock_gating_registers** command is to be undone. A register cannot be on more than one of the **-include_instances**, **-exclude_instances**, and **-undo** lists.

DESCRIPTION

This command specifies registers for which clock gating is to be either enabled or disabled, overriding all conditions necessary for automatic clock gating by the **compile_ultra -gate_clock** command. Registers on the **-include** and **-exclude** lists either receive or do not receive clock gating, regardless of any previous specifications.

Use the **-undo** option to remove the effect of an earlier invocation of the **set_clock_gating_registers** command.

You must run the **set_clock_gating_registers** command before running the **compile_ultra -gate_clock** command; the specifications persist during all subsequent executions of **compile_ultra -gate_clock**. The **set_clock_gating_registers** command directs **compile_ultra -gate_clock** command to implement synchronous load enable flip-flops with gated clocks instead of data feedback through multiplexers if certain conditions (for example, minimum register bank size) are satisfied. This command

overrides these conditions for the specified registers; thus, registers specified by the **-include** option are clock-gated, and registers specified by the **-exclude** option are not clock-gated, regardless of any previously-specified conditions necessary for automatic clock gating.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example excludes the register bank named A_reg in the current design, so that these registers are not clock-gated:

```
prompt> set_clock_gating_registers -exclude_instances A_reg[*]
```

The following example includes the register bank named D_OUT_reg in the MID subdesign, so that this register is always clock-gated:

```
prompt> set_clock_gating_registers -include_instances MID/D_OUT_reg[*]
```

The following example includes and excludes clock gating for registers in the ADDER subdesign:

```
prompt> set_clock_gating_registers \  
      -include_instances ADDER/out1_reg[*] \  
      -exclude_instances ADDER/out2_reg[*]
```

The following example removes the directive to include and exclude clock gating to and from the registers in the ADDER subdesign:

```
prompt> set_clock_gating_registers \  
      -undo {ADDER/out1_reg[*] ADDER/out2_reg[*]}
```

SEE ALSO

`report_clock_gating(2)`

set_clock_groups

Specifies clock groups that are mutually exclusive or asynchronous with each other in a design so that the paths between these clocks are not considered during the timing analysis.

SYNTAX

```
Boolean set_clock_groups  
-physically_exclusive  
  | -logically_exclusive  
  | -asynchronous  
[-allow_paths]  
[-name name]  
-group clock_list
```

ARGUMENTS

-physically_exclusive

Specifies that the clock groups are physically exclusive with each other. Physically exclusive clocks cannot co-exist in the design physically. An example of this is multiple clocks that are defined on the same source pin. The **-physically_exclusive**, **-logically_exclusive** and **-asynchronous** options are mutually exclusive; you must choose only one.

-logically_exclusive

The two types of exclusive clocks are physically exclusive ones and logically exclusive ones. An example of logically-exclusive clocks is multiple clocks, which are selected by a MUX but may have coupling with each other in the design. However, it is not recommended that you set these MUXed clocks to be exclusive if some physical paths exist among them somewhere in the design. The **-physically_exclusive**, **-logically_exclusive** and **-asynchronous** options are mutually exclusive; you must choose only one.

-asynchronous

Specifies that the clock groups are asynchronous to each other. Two clocks are asynchronous with respect to each other if they have no phase relationship at all. Signal integrity analysis uses an infinite arrival window on the aggressor unless all the arrival windows on the victim net and the aggressor net are defined by synchronous clocks. The **-physically_exclusive**, **-logically_exclusive** and **-asynchronous** options are mutually exclusive; you must choose only one.

-allow_paths

Enable the timing analysis between specified clock groups. If this option is not specified, the timing analysis among the defined clock groups are disabled. This option can be used with asynchronous clock groups only.

-name *name*

Specifies a name for the clock grouping to be created. Each command should specify a unique name, which identifies the exclusive or asynchronous relationship among specified clock groups, and this name is used later on to easily remove the clock grouping defined here. By default, the command

creates a unique name.

-group *clock_list*

Specifies a list of clocks.

You can use the **-group** option more than once in a single command execution. Each **-group** iteration specifies a group of clocks, which are exclusive or asynchronous with the clocks in all other groups. If only one group is specified, it means that the clocks in that group are exclusive or asynchronous with all other clocks in the design. A default other group is created for this single group. Whenever a new clock is created, it is automatically included in this group.

Substitute the list you want for *clock_list*.

DESCRIPTION

Specifies clock groups that are mutually exclusive or asynchronous with each other in a design. The timing paths between these clocks are not considered during timing analysis if **-allow_paths** is not specified. One clock cannot be defined in multiple times during one **set_clock_groups** command execution, but can be included in multiple groups by executing multiple **set_clock_groups** commands.

The two types of exclusive clocks are not treated differently for simple timing analysis. However, signal integrity analysis treats logically exclusive clocks as synchronous for timing windows. The physically exclusive clocks are not considered for timing window analysis with each other.

The paths between these exclusive or asynchronous clocks are not explored during timing analysis unless **-allow_paths** is specified. This is similar to declaring false paths between these clocks. Thus, you do not have to manually set a false path if you have already specified two clocks as exclusive or asynchronous. If a false path is already set between two clocks when you specify that they are exclusive or asynchronous, the false path is overwritten by the **set_clock_groups** command. Other exceptions are unaffected.

When the clocks are asynchronous to each other, the crosstalk analysis ignores the timing relationship between them during the timing window overlap analysis. This scenario is also referred as infinite window overlap. The results can be optimistic without infinite window overlap for the synchronous clocks. Hence it is important that the **set_clock_groups -asynchronous** should be used when the clocks are not synchronous to each other.

A generated clock and its master clock are not in the same group by default when some exclusive or asynchronous clock groups defined. You should put them explicitly in the same group if needed.

If multiple clock groups relationship is defined for the same pair of clocks, physically exclusive has the highest precedence followed by asynchronous and logically exclusive in that order.

To undo the **set_clock_groups** command, use the **remove_clock_groups** command. To report the clock groups defined in a design, use the **report_clock** command with the **-groups** option.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example defines two asynchronous clock domains.

```
prompt> set_clock_groups -asynchronous -name g1 -group CLK1 -group CLK2
```

The following example defines a clock named *CLK1* as asynchronous with all other clocks in a design.

```
prompt> set_clock_groups -asynchronous -group CLK1
```

The following example shows how to simultaneously analyze multiple clocks per register without manually specifying false paths. Assume two pairs of mutually-exclusive clocks that are multiplexed:

CLK1 and *CLK2*
CLK3 and *CLK4*.

If each pair of clocks is selected by a different signal, you must execute two **set_clock_groups** commands to simultaneously analyze all four clocks.

```
prompt> set_clock_groups -logically_exclusive -group CLK1 -group CLK2  
prompt> set_clock_groups -logically_exclusive -group CLK3 -group CLK4
```

If each pair of clocks is selected by the same signal, you must execute only one **set_clock_groups** command to simultaneously analyze all four clocks.

```
prompt> set_clock_groups -logically_exclusive \  
        -group {CLK1 CLK3} -group {CLK2 CLK4}
```

To define clocks *CLK1* and *CLK2* as physically exclusive

```
prompt> set_clock_groups -physically_exclusive \  
        -group {CLK1} -group {CLK2}
```

SEE ALSO

```
remove_clock_groups(2)  
report_clock(2)  
set_false_path(2)  
create_clock(2)  
create_generated_clock(2)
```

set_clock_latency

Specifies clock network latency.

SYNTAX

```
string set_clock_latency  
[-rise]  
[-fall]  
[-min]  
[-max]  
[-source]  
[-early]  
[-late]  
[-clock clock_list]  
delay  
object_list
```

Data Types

<i>clock_list</i>	list
<i>delay</i>	float
<i>object_list</i>	list

ARGUMENTS

- rise**
Indicates that **delay** is to apply only to rise clock network latency. By default, **delay** is applied to both rise and fall clock network latency.
- fall**
Indicates that **delay** is to apply only to fall clock network latency. By default, **delay** is applied to both rise and fall clock network latency.
- min**
Indicates that **delay** is to apply only to minimum clock network latency. By default, **delay** is applied to both minimum and maximum clock network latency.
- max**
Indicates that **delay** is to apply only to maximum clock network latency. By default, **delay** is applied to both minimum and maximum clock network latency.
- source**
Indicates that **delay** is to apply to clock source latency. By default, **delay** is applied to clock network latency.
- early**
Indicates that **delay** is to apply only to early clock source latency. By default, if **-source** is specified, **delay** is applied to both late and early clock source latency.
- late**
Indicates that **delay** is to apply only to late clock source latency. By

default, if **-source** is specified, **delay** is applied to both late and early clock source latency.

-clock *clock_list*

Indicates that **delay** is applied with respect to the specified clocks. By default, **delay** is applied to all specified objects.

delay

Specifies the clock latency value.

object_list

Specifies a list of names of clocks, ports, and pins for which the clock latency is to be set.

DESCRIPTION

Two types of clock latency can be specified for a design. The clock network latency is the time it takes a clock signal to propagate from the clock definition point to a register clock pin. The rise and fall latencies are the latencies for rising and falling transitions at the register clock pin, respectively. Inversion of the clock waveform, if present in the clock network, is taken into consideration when computing clock network latencies at register clock pins. Clock source latency (also called insertion delay) is the time it takes for a clock signal to propagate from its actual ideal waveform origin point to the clock definition point in the design. It can be used to model off-chip clock latency when a clock generation circuit is not part of the current design. For generated clocks, clock source latency can be used to model the delay from master-clock to generated clock definition point. The **-early** and **-late** options can be used to specify early and late clock source latencies respectively.

If multiple clocks are allowed per object, the clock latency can be specified with respect to each clock with **-clock** option. Different values can be applied, such that the different clock network can have the different clock latencies.

Design Compiler assumes ideal clocking, which means clocks have a specified network latency (from the **set_clock_latency** command) or zero network latency by default. Propagated clock network latency (from the **set_propagated_clock** command) is normally used for post-layout, after final clock tree generation. Ideal clock network latency provides an estimate of the clock tree for pre-layout.

Clock source latency can be specified for ideal or propagated clocks. The total clock latency at a register clock pin is the sum of clock source latency and clock network latency.

If **set_clock_latency** is applied to pins or ports, it affects all register clock pins in the transitive fanout of the pins or ports. Clock source latencies are only allowed on clocks and clock source pins.

To undo **set_clock_latency**, use **remove_clock_latency**.

To see clock network latency and clock source latency information, use **report_clock-skew**.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example specifies a rise latency of 1.2 and a fall latency of 0.9 for clock "CLK1".

```
prompt> set_clock_latency 1.2 -rise [get_clocks CLK1]
prompt> set_clock_latency 0.9 -fall [get_clocks CLK1]
```

The next example specifies an early rise and an early fall source latency of 0.8, and a late rise and a late fall source latency of 0.9 for "CLK1".

```
prompt> set_clock_latency 0.8 -source -early [get_clocks CLK1]
prompt> set_clock_latency 0.9 -source -late  [get_clocks CLK1]
```

SEE ALSO

```
create_clock(2)
current_design(2)
remove_clock_latency(2)
report_clock(2)
set_clock_transition(2)
set_clock_uncertainty(2)
set_input_delay(2)
set_propagated_clock(2)
```

set_clock_sense

Specifies the clock sense (with respect to the clock source) propagating forward from the specified pins.

SYNTAX

```
string set_clock_sense
[-stop_propagation]
[-positive]
[-negative]
[-
pulse {rise_triggered_high_pulse | rise_triggered_low_pulse | fall_triggered_high_p
ulse | fall_triggered_low_pulse}]
[-clocks clocks]
pins
```

Data Types

```
clocks      list
pins        list
```

ARGUMENTS

-stop_propagation

Stops propagation of the specified clocks at the specified pins.
The **-stop_propagation** option cannot be specified with **-positive**, **-negative**, or **-pulse**.

-positive

Propagates only the positive unate paths forward from the specified pins.
The **-positive** option cannot be specified with **-negative**, **-pulse**, or **-stop_propagation**.

-negative

Propagates only the negative unate paths forward from the specified pins.
The **-negative** option cannot be specified with **-positive**, **-pulse**, or **-stop_propagation**.

-pulse {rise_triggered_high_pulse | rise_triggered_low_pulse | fall_triggered_high_pulse | fall_triggered_low_pulse}" Specifies the type of pulse clock that is generated from the specified pins.
The **-pulse** option cannot be specified with **-positive**, **-negative**, or **-stop_propagation**.

-clocks *clocks*

Specifies the clocks to which the specified clock sense applies.
If you do not specify this option, the clock sense setting applies to any clock that passes through the specified pins.

pins

Specifies a list of pins on which to set the clock sense.

DESCRIPTION

This command provides the capability to define the clock sense at nonunate points in the clock network (the command is ignored if it is used on a unate point in the clock network). The specified clock sense propagates forward from the specified pins.

If the **-clocks** option is supplied, only the specified clocks are considered. Otherwise, all clocks passing through the specified pins are considered.

Warning messages are issued if the specified sense can not be respected on the specified pins. Hierarchical pins are not supported.

To undo **set_clock_sense**, use the **remove_clock_sense** command.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example specifies positive unateness for a pin named XOR/Z with respect to clock CLK1.

```
prompt> set_clock_sense -positive -clocks [get_clocks CLK1] XOR/Z
```

The following example specifies a pulse type for a pin named MUX/Z for all clocks.

```
prompt> set_clock_sense -pulse rise_triggered_high_pulse MUX/Z
```

SEE ALSO

`remove_clock_sense(2)`

set_clock_transition

Sets clock transition attributes on clock objects.

SYNTAX

```
int set_clock_transition  
transition  
[-rise | -fall]  
[-min]  
[-max]  
clock_list
```

Data Types

<i>transition</i>	float
<i>clock_list</i>	list

ARGUMENTS

transition

Allows for the value of the desired transition for clock nets directly fanning out to a sequential device clocked by the clock you specified. Express *transition* in the same unit as the technology library used during optimization.

-rise | -fall

Specifies whether the value is applicable for rising or falling transition. If neither is specified, both rising and falling clock transitions are set. If you specify **-rise** or **-fall**, the other transition value is assumed to be 0.0, if an explicit value does not already exist.

-min

Specifies that the value is applicable for minimum delay analysis. If no value is specified for minimum delay analysis, the maximum value is used.

-max

Specifies that the value is applicable for maximum delay analysis. If no value is specified for minimum delay analysis, the maximum value is used.

clock_list

Provides a list of names of clocks in the current design. The transition times on all register clock pins in the transitive fanout of specified clock are affected. Only ideal clocks can be specified in the list. The values are ignored for propagated skews on register clock pins.

DESCRIPTION

This command overrides the calculated transition times on clock pins of registers and associated nets. The **set_clock_transition** command places **clock_rise_transition** and **clock_fall_transition** attributes on the *clock_list*.

This command is especially useful for prelayout when clock trees are incomplete, and because calculated transition times at register clock pins can be pessimistic. The transition value, specified with this command, overrides the transition times of all nets directly feeding a sequential element clocked by the specified clock.

Use this command only with ideal clocks. For propagated clocks, the calculated transition times are used. Set propagated clocks only after the final clock tree is constructed.

For ideal clocks, the calculated transition values are used if no explicit transition times are specified using **set_clock_transition**.

To undo **set_clock_transition**, use the **remove_clock_transition**, **remove_attribute**, or **reset_design** command.

To list all clock transition values that have been set, use **report_clock -skew**.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example sets both rise and fall transition times to 0.75 on clock pins of all sequential elements clocked by *CLK*:

```
prompt> set_clock_transition 0.75 CLK
```

The following example sets only the fall transition time to 0.64 on clock pins of all sequential elements clocked by *CLK*:

```
prompt> set_clock_transition 0.64 -fall CLK
```

The following example sets both rise and fall transition times to 0.0 on clock pins of all sequential elements clocked by a certain clock:

```
prompt> set_clock_transition 0.0 all_clocks()
```

SEE ALSO

```
create_clock(2)  
current_design(2)  
remove_attribute(2)  
remove_clock_transition(2)  
report_clock(2)  
reset_design(2)
```

set_clock_tree_exceptions

Creates definitions for the following exceptions: float pins, stop pins, nonstop pins, exclude pins, dont_touch_subtree pins, dont_buffer nets, dont_size cells, and size_only cells.

SYNTAX

```
status set_clock_tree_exceptions
[-float_pin_max_delay_rise max_delay_rise_value]
[-float_pin_min_delay_rise min_delay_rise_value]
[-float_pin_max_delay_fall max_delay_fall_value]
[-float_pin_min_delay_fall min_delay_fall_value]
[-float_pins float_pin_collection]
[-stop_pins stop_pin_collection]
[-non_stop_pins non_stop_pin_collection]
[-exclude_pins exclude_pin_collection]
[-dont_touch_subtrees dont_touch_pin_collection]
[-dont_buffer_nets collection_or_list_of_nets]
[-dont_size_cells collection_or_list_of_cells]
[-size_only_cells collection_or_list_of_cells]
[-float_pin_logic_level logic_level_number]
[-max_float_pin_scale_factor max_scale_factor]
[-min_float_pin_scale_factor min_scale_factor]
[-preserve_hierarchy hierarchy_preservation_pin_or_cell]
[-clocks object_list]
```

Data Types

<i>max_delay_rise_value</i>	float
<i>min_delay_rise_value</i>	float
<i>max_delay_fall_value</i>	float
<i>min_delay_fall_value</i>	float
<i>float_pin_collection</i>	string
<i>stop_pin_collection</i>	string
<i>non_stop_pin_collection</i>	string
<i>exclude_pin_collection</i>	string
<i>dont_touch_pin_collection</i>	string
<i>collection_or_list_of_nets</i>	string
<i>collection_or_list_of_cells</i>	string
<i>logic_level_number</i>	integer
<i>max_scale_factor</i>	float
<i>min_scale_factor</i>	float
<i>hierarchy_preservation_pin_or_cell</i>	string

ARGUMENTS

-float_pin_max_delay_rise *max_delay_rise_value*
Specifies the maximum rising-edge-induced delay from the float pin to its end pin (the longest path). This option should be used in combination with the **-float_pins** option. The time unit is the same as the library unit.

- float_pin_min_delay_rise** *min_delay_rise_value*
 Specifies the minimum rising-edge-induced delay from the float pin to its end pin (the shortest path). This option should be used in combination with the **-float_pins** option. The time unit is the same as the library unit.
- float_pin_max_delay_fall** *max_delay_fall_value*
 Specifies the maximum falling-edge-induced delay from the float pin to its end pin (the longest path). This option should be used in combination with the **-float_pins** option. The time unit is the same as the library unit.
- float_pin_min_delay_fall** *min_delay_fall_value*
 Specifies the minimum falling-edge-induced delay from the float pin to its end pin (the shortest path). This option should be used in combination with the **-float_pins** option. The time unit is the same as the library unit.
- float_pins** *float_pin_collection*
 Specifies a collection of input pins to be set as float pins. A float pin is a synchronous pin whose delay is user-specified. A float exception can overwrite stop exception on a pin. If you use this option, you should also specify at least one of the **-float_pin_max_delay_rise**, **-float_pin_min_delay_rise**, **-float_pin_max_delay_fall**, and **-float_pin_min_delay_fall** options.
- stop_pins** *stop_pin_collection*
 Specifies a collection of input pins to be set as stop pins. A stop pin is a synchronous pin whose phase delay is 0.
- non_stop_pins** *non_stop_pin_collection*
 Specifies a collection of input pins to be set as non-stop pins. These are pins that clock tree synthesis treats as sinks, with some exceptions like implicit exclude pins. Currently, the tool does not support the specifying of a non-stop pin exception for hierarchical pins and boundary ports. Specifying a non-stop exception on a pin that has an already-existing stop, float exception, or exclude exception, overwrites the previous exception.
- exclude_pins** *exclude_pin_collection*
 Specifies a collection of input pins that the **set_clock_tree_exceptions** command is to set as exclude pins. An exclude exception can overwrite a stop or float exception on a pin.
- dont_touch_subtrees** *dont_touch_pin_collection*
 Specifies a collection of input pins that the **set_clock_tree_exceptions** command is to set as dont_touch_subtree pins.
- dont_buffer_nets** *collection_or_list_of_nets*
 Specifies clock nets that are not to be buffered during clock tree synthesis. You can use this option to prevent clock tree synthesis from inserting cells on specified nets. It can be used in conjunction with a configuration file. The **optimize_clock_tree** command also honors this attribute and does not insert delay cells on the specified nets. The primary use of this option is to preserve the subtree structure for later runs of synthesis.
- dont_size_cells** *collection_or_list_of_cells*
 Specifies clock tree cells that can be moved but cannot be sized during clock tree synthesis and optimization.

`-size_only_cells collection_or_list_of_cells`
 Specifies the clock tree cells that can only be sized but not moved during clock tree synthesis and optimization.

`-float_pin_logic_level logic_level_number`
 Specifies the logic level to be set for those float pins defined by the **-float_pins** option. This option should be used in combination with the **-float_pins** option. The default value is 0 if you do not use the **-float_pin_logic_level** option when you define a float pin.

`-max_float_pin_scale_factor max_scale_factor`
 Specifies a scale factor that should be applied to float pin delays. This scale factor applies for the maximum operating condition of the currently active multicorner-multimode scenario. Setting a value of 1.0 implies no scaling. See further information in the DESCRIPTION section.

`-min_float_pin_scale_factor min_scale_factor`
 Specifies a scale factor that should be applied to float pin delays. This scale factor applies for the minimum operating condition of the currently active multicorner-multimode scenario. Setting a value of 1.0 implies no scaling. See further information in the DESCRIPTION section.

`-preserve_hierarchy hierarchy_preservation_pin_or_cell`
 Specifies a collection of cells, pins, or hierarchy instances for which boundary needs to be preserved.

DESCRIPTION

This command sets the following types of exceptions: float pins, stop pins, nonstop pins, exclude pins, dont_touch_subtree pins, dont_buffer nets, dont_size cells, and size_only cells.

Float Pins

A float pin is a user-defined timing model that describes a delay as seen from a pin. If you expect that a float pin is to be rising edge, you should specify `max_rise` and `min_rise` delay. If you expect it to be falling edge, you should specify `max_fall` and `min_fall` delay. If you expect it to be both rising and falling edge, you should specify all 4 delays: `max_rise`, `min_rise`, `max_fall`, and `min_fall`. Float pins are used in the following situations:

- To define the data setup requirements for hard macros. For example, if a clock signal terminates on a RAM pin that has significant setup requirements, you might want to synthesize the clock tree so that the arrival time of the clock signal at the RAM occurs before the arrival time of the clock signal at the sequential clock sinks. The float pin alerts the clock tree synthesis engine that there is additional internal delay within the RAM.
- To provide a coarse model of the internal clock network delays inside a hierarchical block that is timing-arc-based timing modeled. (For example, one that is using STAMP, ETM, QTM, or a Library Compiler format.) If the internal clock network is already implemented, use the timing arcs between the clock pins and the dedicated internal clock pins to model the internal clock delays. This technique

allows you to properly reflect the sensitivity of the internal clock delays to the input clock transition delay.

- To capture the budgeted clock network delays of a portion of the current clock network that will be implemented separately. In this case, define the float pin as the input pin of the subtree root cell. If the subtree is already implemented, use the **-dont_touch_subtrees** option instead.

Stop Pins

A stop pin is an explicitly specified end pin of a clock tree. Unlike default clock sinks, a stop pin can be the input pin of a nonsequential cell. Clock tree synthesis treats a stop pin as a clock sink.

Clock tree synthesis minimizes skew up to the specified stop pin. The tool ignores any insertion delay in the transitive fanout between the stop pin and a default clock sink.

Clock tree synthesis determines the active edge of a stop pin from the timing arcs of the corresponding library cell. Usually, an input pin of a combinational cell is sensitive to both the rising edges and the falling edges. To override stop pin edge sensitivity and set a rising edge stop pin, use the following format:

```
set_clock_tree_exceptions -float_pins <pin_name> -float_pin_max_delay_rise 0 -float_pin_min_delay_rise 0. To set a falling edge stop pin, use:  
set_clock_tree_exceptions -float_pins <pin_name> -float_pin_max_delay_fall 0 -float_pin_min_delay_fall 0.
```

Exclude Pins

An exclude pin is a clock sink whose timing is not important. Excluding a pin from the clock tree removes individual default clock sinks or branches of a clock tree. Clock tree synthesis does not calculate skew and insertion delay for the excluded clock sinks or branches. Similarly, the **report_clock_tree** command does not consider exclude pins in the skew calculations.

dont_touch_subtree Pins

A dont_touch_subtree pin does not change the set of a sink that requires balancing, but it forces the **compile_clock_tree** command to maintain an existing part of a clock tree "as is." A dont_touch_subtree pin is used when two clock networks share part of some clock logic behind a multiplexer.

During clock tree synthesis, no modifications (such as buffering, clock net routing, attribute setting, or back-annotation) occur on any of the clock tree nets in the transitive fanout of a dont_touch_subtree pin. You must ensure that the transitive fanout of the dont_touch_subtree pin is properly synthesized. If some nets in the transitive fanout have not been properly buffered and back-annotated, the post-layout correlation of clock tree delays are likely to be poor. The other types of clock tree exceptions are supported in the transitive fanout of the dont_touch_subtree pin.

The **report_timing** command does not take into account clock-tree exceptions. For

instance, the **report_timing** command reports that clock network path delays are going through combinational clock gates bearing an exclude pin exception.

Scaling float pin delays in multiple corners

For multicorner clock optimization and reporting, you should set a "float pin scale factor" for each scenario of interest. Setting the scale factor simultaneously balances clocks across different process and temperature corners. The scale factor is applied to all float pin delays in the design. Set it by using the -**max_float_pin_scale_factor** and -**min_float_pin_scale_factor** options. Setting a value of 1.0 implies no scaling.

When multicorner-multimode scenarios are in use and scale factors are not set, the **report_clock_tree** command issues a warning, and the tool performs no scaling. This might result in misleading reports. If you do not provide scale factors, multicorner clock optimization (performed by the **optimize_clock_tree** command) automatically fills them in. After multicorner optimization is complete, you can safely report clock delays in all corners used for optimization. Note that multicorner clock optimization sets scale factors only for scenarios that are active when the tool runs the **optimize_clock_tree** command.

Preserve hierarchy for pin, cell, or hierarchy instances

Specifies a list of pin, cell, or hierarchy instances for which boundary will be preserved. There is neither new port punch nor port renaming for the specified instances.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The examples in this section are on based on the following scenario.

- A clock tree named CLK1 has been defined to begin at a pin of one of the I/O pads, with fanout to three child hierarchical blocks named Reg_block, Mult_block, and Viterbi_block. The precompiled clock tree of the Viterbi_block port cannot be altered.
- Timing analysis shows that the delay of the precompiled clock tree from the Viterbi_block/CLK port to the first end pin is 0.33nS and the delay to the last end point is 0.52nS.

The following example defines the Viterbi_block/CLK clock input port as a float pin.

```
prompt> set_clock_tree_exceptions\  
-float_pin_max_delay_rise 0.52\  
-float_pin_min_delay_rise 0.33\  
-float_pins [get_pins Viterbi_block/CLK]
```

The following example defines a RAM pin named Mult_block/RAM0/EN as a float pin. The internal setup requirement for the RAM is 0.12nS.

```
prompt> set_clock_tree_exceptions\  
-float_pin_max_delay_fall 0.12\  
-float_pin_min_delay_fall 0.12\  
-float_pins [get_pins Mult_block/RAM0/EN]
```

The following example explicitly defines sink pins. The original clock tree has in place intermediate combinational logic pins named U1_AND02/IN1 and U1_AND02/OUT before some of the implicit sequential sink pins.

```
prompt> set_clock_tree_exceptions\  
-stop_pins [get_pins U1_AND02/IN1]
```

The following example removes two sink pins from skew examination during subsequent synthesis of the clock tree.

```
prompt> set_clock_tree_exceptions\  
-exclude_pins [get_pins UReg_1/CLK UReg_2/CLK]
```

The following example prevents the cell CLKBUF1_G1B1I1 from being resized.

```
prompt> set_clock_tree_exceptions\  
-dont_size_cells {CLKBUF1_G1B1I1}
```

Alternatively, you can use a collection of cell instances as the argument to the **-dont_size_cells** option:

```
prompt> set_clock_tree_exceptions\  
-dont_size_cells [get_cell CLKBUF1_G1B1I1]
```

The following example specifies the cell CLKBUF1_G1B1I1 cell for cell sizing only.

```
prompt> set_clock_tree_exceptions\  
-size_only_cells {CLKBUF1_G1B1I1}
```

Alternatively, you can specify a collection of cell instances for cell sizing only.

```
prompt> set_clock_tree_exceptions\  
-size_only_cells [get_cell CLKBUF1_G1B1I1]
```

In the following example, the clock tree has intermediate combinational logic pins named U1_AND02/IN1 and U1_AND02/OUT. The U1_AND02/IN1 pin traces through several registers that do not need skew examination. This example removes from examination during clock tree synthesis the branch that begins at U1_AND02/IN1 and ends at sequential pins.


```
prompt> set_clock_tree_exceptions\  
-exclude_pins [get_pins U1_AND02/IN1]
```

The following example sets scale factors for float pins in different multicorner-multimode scenarios.

```
prompt> current_scenario scenario1  
prompt> set_clock_tree_exceptions\  
-max_float_pin_scale_factor 1.0\  
-min_float_pin_scale_factor 0.6  
prompt> current_scenario scenario2  
prompt> set_clock_tree_exceptions \  
-max_float_pin_scale_factor 0.9\  
-min_float_pin_scale_factor 0.5
```

The following example defines hierarchy preservation exception on hierarchy cell reg_g2.

```
prompt> set_clock_tree_exceptionsfp  
-preserve_hierarchy [get_cells reg_g2]fp
```

SEE ALSO

```
compile_clock_tree(2)  
remove_clock_tree_exceptions(2)  
set_clock_tree_options(2)  
set_clock_tree_references(2)
```

set_clock_tree_optimization_options

Sets options used by clock tree optimization.

SYNTAX

```
status set_clock_tree_optimization_options
[-clock_trees clock_name_or_collection]
[-enable_multicorner_optimization {none | all | corner_spec}]
[-corner_target_skew corner_values]
[-gate_sizing true | false]
[-gate_relocation true | false]
[-area_recovery true | false]
[-preserve_levels true | false]
[-relax_insertion_delay true | false]
[-balance_rc true | false]
```

Data Types

<i>clock_name_or_collection</i>	collection
<i>corner_values</i>	string

ARGUMENTS

- clock_trees *clock_name_or_collection***
Specifies a set of clocks to be processed. Only clock names defined through `create_clock` or `create_generated_clock` are accepted. If there is a list of clock names, the options then apply to those clocks only. By default, the options apply for all the clocks.
- enable_multicorner_optimization {none | all | corner_spec}**
Specifies a type of multicorner optimization to the command. This option applies to all clocks in the design, and cannot be specified together with `"-clock_trees clock_name_or_collection"`. The valid values are one of **none**, **all**, or *corner_spec*, a corner specification (described below). The default value is none.
When **none** is specified, optimization considers only the max corner in the current MCMM senario.
When **all** is given, optimization both the min and max corners of all currently active MCMM scenarios.
When *corner_spec* is used, optimization will look only at the specified corners. The specified corner may be from inactive MCMM corners, if you wish.
- corner_target_skew *corner_values***
Sets target skews for selected process corners. This setting overrides the setting of the `-target_skew` option of the `set_clock_tree_options` command. For corners that are not specified, the value from the `-target_skew` option of the `set_clock_tree_options` command will be used. If that value is not specified, the zero will be used. See below for details on the format of the corner values.
You can use `-corner_target_skew` to set a target value for RC-skew by using the special corner name "rc". If you do not set this value, the tool will select an appropriate value for you -- based on your technology and other

skew targets.

-gate_sizing true | false

Enables or disables gate sizing. The valid value is **true** or **false**, default is **true**. The gates in question are the user instantiated gates and buffers only and not introduced cells during `compile_clock_tree` or `optimize_clock_tree`; `Compile_clock_tree` or `optimize_clock_tree` introduced cells can always be sized, with the exception of boundary cells. User gates, for example, are those that are introduced during prior to `compile_clock_tree`, or user may have put them for some purpose. By default, `gate_sizing` is **true**.

-gate_relocation true | false

Enables or disables gate relocation; The valid value is **true** or **false**, default is **true**. The gates in question are the user instantiated gates and buffers only and not introduced cells during `compile_clock_tree` or `optimize_clock_tree`; `Compile_clock_tree` or `optimize_clock_tree` introduced cells can always be relocated. User gates, for example, are those that are introduced prior to `compile_clock_tree`, or user may have put them for some purpose. By default, `gate_relocation` is **true**.

-area_recovery true | false

Controls optimization via area recovery; The valid value is **true** or **false**, default is **true**. Area recovery will improve clock tree area/power, without degrading insertion delay or skew.

-preserve_levels true | false

Preserves levels. The valid value is **true** or **false**. Default is **false**. If set to **true**, then the number of levels will be preserved. This option is combining delay insertion and delay removal options. The `target_early_delay`, if specified with `set_clock_tree_options`, will override `preserve_levels` only at the root of the clock. By default, `preserve_levels` is **false**.

-relax_insertion_delay true | false

Adds insertion delay of the clock tree. The valid value is **true** or **false**. Default is **false**. If set to **true**, then the insertion delay of the clock tree is increased approximately 10% over the minimum possible value. The added insertion delay allows area/power to be improved -- sometimes significantly.

-balance_rc true | false

Equalizes the RC delay between different clock paths. The valid value is **true** or **false**. Default is **false**. If set to **true**, then optimization is done to help equalize the RC delay between wildly different clock paths. Buffers at the root of clock paths that cover short distances will be relocated so that the RC delay is closer to that of long paths. In general, this optimization adds relatively little wire, and it improves overall skew in many process and temperature corners.

DESCRIPTION

This command is the UI setting options for clock tree optimization in the tool. Before command `optimize_clock_tree` or `clock_opt`, user can use this command to set options to specific clocks if `-clock_trees` is given, or set options to the design which will be applied to all clocks that do not have the per clock option.

Enabling multi-corner optimization and defining appropriate MCMM scenarios, will cause `optimize_clock_tree` to perform multi-corner optimization. The user can also use this command to enable or disable some of the other processing steps of `optimize_clock_tree`.

This command only sets options for `optimize_clock_tree`. In `clock_opt`, `compile_clock_tree` and `optimize_clock_tree` are called. This `set_clock_tree_optimization_options` only affect `optimize_clock_tree` part of `clock_opt`.

Options `-gate_sizing`, `-gate_relocation` are both under `set_clock_tree_options` and `set_clock_tree_optimization_options`. If they are under `set_clock_tree_optimization_options`, they control `optimize_clock_tree`; if under `set_clock_tree_options`, they control optimization during `compile_clock_tree`. Under `set_clock_tree_options`, there are `-buffer_sizing`/`-buffer_relocation`/`-delay_insertion`. They also only control `compile_clock_tree`.

Specifying corners for optimization

The `-enable_multicorner_optimization` option of `set_clock_tree_optimization_options` turns on multicorner clock tree optimization if you use the "all" value or use a corner specification. The corner specification selects process/temperature/voltage corners from defined MCMM scenarios that should be to optimize clocks.

Each MCMM has two corners, referred to as "max" and "min". In each corner, you may define a operating condition (PVT) specification and a parasitic specification. The operating condition is set up using the `set_operating_conditions` command. The parasitic information comes from the `set_tlu_plus_files` command.

Additionally, the tool timing engine requires you to draw an association between the cell library used in the max corner and the cell library used in the min corner. Traditionally, this was done with the `set_min_library` command. Instead, we will suggest that users leverage the newer and more robust `set_scaling_lib_group` command. It is possible to define scaling groups which contain only one library. And once the scaling groups are defined, you can use them without restriction in any scenario.

It is important to note that `optimize_clock_tree` can optimize in multiple corners. However, it does not address multiple modes. The clock tree mode behavior is based on a single "cts scenario". By default, the cts scenario is the "current" scenario as set by the `current_scenario` command. You can override this default with the `set_cts_scenario` command. Other scenarios will only be used for there PVT and wire information.

If you choose to optimize your clock using only selected MCMM corners, you need to provide a corner specification to the `-enable_multicorner_optimization` option of `set_clock_tree_optimization_options`. The format of a corner specification is a space-separated string of corners. Each corner is a scenario name, followed by a colon (:), followed by "max" or "min". So if you wanted to select the max corner of `scenario1` and the min corner of `scenario2`, you would use

```
"scenario1:max scenario2:min"
```

Additionally, you can use the special scenario name called "default". The default scenario is always defined to be the same as the "cts scenario" (see above). Using

"default" is also useful if you have not defined any scenarios. In this case, "default" refers to the corners defined for non-MCMM optimization.

```
"default:max default:min"
```

You should note that the max corner of the cts scenario is always used as the first corner in multicorner clock optimization. It is not possible to turn off optimization of this corner. This first corner is treated specially, and should represent the slowest gates and slowest wires of any corner. This will give you the best optimization results.

Specifying corner values for optimization

The `-corner_target_skew` option of `set_clock_tree_optimization_options` requires a "corner_values" argument. This argument is similar to the corner specification in the previous section, except that each corner is followed by an equal sign (=) and a floating point value. So if you want to set the target skew of corner scenario1:max to 0.050, and corner scenario1:min to 0.030, then you would specify:

```
"scenario1:max=0.05 scenario2:min=0.030"
```

You can use `-corner_target_skew` to set a target value for RC-skew by using the special corner name "rc":

```
"rc=0.05"
```

Note there are no spaces between the corner, the equal sign and the floating point value. So the following is incorrect:

```
"scenario1:max = 0.05"      # incorrect
```

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example disables `gate_sizing` to clock `PCI_CLK`.

```
prompt> set_clock_tree_optimization_options\  
  \-clock_trees [get_clock PCI_CLK]\  
  -gate_sizing false  
1
```

This example sets the multicorner optimization type to "all" for all clocks.

```
prompt> set_clock_tree_optimization_options\  
  -enable_multicorner_optimization all
```

This example sets the multicorner optimization type to work on two selected corners.

```
prompt> set_clock_tree_optimization_options\  
-enable_multicorner_optimization "scenario1:max scenario2:min"
```

This example sets corner-specific target skews, and also sets a target for RC-skew.

```
prompt> set_clock_tree_optimization_options\  
-corner_target_skew "scenario1:max=0.100 scenario2:min=0.050 rc=0.050"
```

SEE ALSO

```
clock_opt(2)  
compile_clock_tree(2)  
create_scenario(2)  
current_scenario(2)  
define_scaling_lib_group(2)  
optimize_clock_tree(2)  
report_clock_tree_optimization_options(2)  
reset_clock_tree_optimization_options(2)  
set_clock_tree_options(2)  
set_cts_scenario(2)  
set_scaling_lib_group(2)
```

set_clock_tree_options

Specifies CTS constraints and options for clocks in the design.

SYNTAX

```
status_value set_clock_tree_options
[-clock_trees clock_name_or_collection]
[-layer_list list_of_layer_names_ (default=_all_routing_layers)]
[-target_early_delay minimum_insertion_delay_ (default=_0)]
[-target_skew desired_skew_ (default=_0)]
[-max_capacitance max_capacitance_value_ (default=_0.6pf)]
[-max_transition max_transition_value_ (default=_0.5ns)]
[-max_fanout max_fanout_value_ (default=_2000)]
[-max_buffer_levels number_of_levels_value_ (default=_50)]
[-max_rc_delay_constraint max_rc_delay_value_ (no_default)]
[-max_rc_scale_factor scale_factor_of_internal_derived_rc_delay_ (no_default)]
[-
routing_rule name_of_the_non_default_routing_rule_ (default=_default_routing_rule)]
[-use_default_routing_for_sinks n_ (default=_0)]
[-buffer_relocation true | false (default = true)]
[-buffer_sizing true | false (default = true)]
[-gate_relocation true | false (default = true)]
[-gate_sizing true | false (default = false)]
[-logic_level_balance true | false (default = false)]
[-ocv_clustering true | false (default = false)]
[-config_file_read filename_ (no_default)]
[-config_file_write filename_ (no_default)]
[-insert_boundary_cell true | false (default = false)]
[-operating_condition string_ (default=_max)]
[-delay_insertion true | false]
```

ARGUMENTS

- clock_trees** *clock_name_or_collection*
Sets the specified options for only those clock trees listed in *clock_name_or_collection*. Each element of *clock_name_or_collection* is the name of the source (a top-level design input port, a hierarchical port, or a library cell instance pin) of each clock tree. If there is a list of clock names given, the options apply to those clocks only. If *clock_trees* is not specified, the options apply for clocks defined in the design.
- layer_list** *list_of_layer_names_* (default=_all_routing_layers)
Specifies the list of layers that can be used during routing of clock nets. If the list has more than 2 elements, the min and max will be derived by the program. This option may either be set on specific clock trees, by using the *clock_trees* option, or on all clocks if *clock_trees* is not specified.
- target_early_delay** *minimum_insertion_delay_* (default=_0)
Specifies the minimum insertion delay constraint as *minimum_insertion_delay* for this clock tree. If specified, the clock tree synthesis engine first attempts to build an optimized clock tree and if the insertion delay (of the longest path) of this clock tree is smaller than the specified value, adds a

chain of cells from the reference list as needed to meet this delay. By default, target early delay is 0. This option may either be set on specific clock trees, by using the `clock_trees` option, or on all clocks if `clock_trees` is not specified.

`-target_skew desired_skew (default=_0)`

Specifies the desired value for maximum skew as `desired_skew` for this clock tree. Once this skew target is met, the optimization will concentrate more on other QOR goals (such as insertion delay, and area). The default value is 0. This option may either be set on specific clock trees, by using the `clock_trees` option, or on all clocks if `clock_trees` is not specified. - `max_capacitance max_capacitance_value` Sets maximum capacitance design rule checking (DRC) constraint for the cells in the clock tree. This value takes precedence over the `max_capacitance` constraint set on the design as well as over the one coming from the library. This constraint is used when reporting `max_capacitance` violations during CTS and is also used to control `max_capacitance` DRC fixing beyond exceptions. . By default, max capacitance is 0.6 (pf). This option may either be set on specific clock trees, by using the `clock_trees` option, or on all clocks if `clock_trees` is not specified.

`-max_capacitance max_capacitance_value (default=_f)`

Sets maximum capacitance design rule checking (DRC) constraint for the cells in the clock tree. This value takes precedence over the `max_capacitance` constraint set on the design as well as over the one coming from the library. This constraint is used when reporting `max_capacitance` violations during CTS and is also used to control `max_capacitance` DRC fixing beyond exceptions. . By default, max capacitance is 0.6 (pf). This option may either be set on specific clock trees, by using the `clock_trees` option, or on all clocks if `clock_trees` is not specified.

`-max_transition max_transition_value (default=_s)`

Sets maximum transition design rule checking (DRC) constraints for the buffers and inverters used while compiling given clock tree. This value takes precedence over the `max_transition` constraint set on the design as well as over the one coming from the library, but can be overridden on all instances of specific types of buffers and inverters in the clock tree by specifying a different value for this constraint on the **set_clock_tree_references** command line. The clock tree root cell and occasional gating cells present on the clock network are not impacted by the **set_clock_tree_references** DRC constraints unless they are instances of buffers or inverters with overridden DRC constraints. By default, max transition is 0.5 (ns).

`-max_fanout max_fanout_value (default=_2000)`

Sets maximum fanout design rule checking (DRC) constraint for the cells in the clock tree. Note that the `max_fanout` constraint set on the design (using `set_max_fanout`) as well as over the one coming from the library are by default ignored during CTS. By default, max fanout is 2000. This option may either be set on specific clock trees, by using the `clock_trees` option, or on all clocks if `clock_trees` is not specified.

`-max_buffer_levels number_of_levels_value (default=_50)`

Specifies the maximum number of levels that can be used for buffering a clock net. Note that this not the maximum number of buffer levels in the entire tree; rather it is the maximum number of buffer levels that are inserted at any level in the tree. If the receiving pins have different phase delay, the

value will automatically be relaxed (increased) so that the number of buffer levels between the driver of the net and the receiving pin with the largest phase delay satisfies the user-specified value. By default, max buffer levels is 50. This option may either be set on specific clock trees, by using the `clock_trees` option, or on all clocks if `clock_trees` is not specified.

`-max_rc_delay_constraint max_rc_delay_value (no_default)`

Specifies the maximum RC delay constraint from a driver pin to each receiver pin while compiling the given clock tree. This constraint is for the delay from a driver pin to each receiver pin. This option is mutually exclusive with `-max_rc_scale_factor` option (see below). By default, this max RC delay constraint is disabled. This option may either be set on specific clock trees, by using the `clock_trees` option, or on all clocks if `clock_trees` is not specified.

`-max_rc_scale_factor scale_factor_of_internal_derived_rc_delay (no_default)`

Enables the maximum RC delay constraint by specifying a scale factor while compiling the given clock tree. The true RC delay constraint is determined from multiplying this scale factor by an internal derived RC delay. This constraint is for the delay from a driver pin to each receiver pin. This option is mutually exclusive with `-max_rc_delay_constraint` option (see above). By default, this scale factor is unset and the max RC delay constraint is disabled. This option may either be set on specific clock trees, by using the `clock_trees` option, or on all clocks if `clock_trees` is not specified.

`-routing_rule name_of_the_non_default_routing_rule (default=_default_routing_rule)`

Specifies the nondefault routing rule to be used for clock tree nets. The default is to use the default routing rule for all clock nets. This option may either be set on specific clock trees, by using the `clock_trees` option, or on all clocks if `clock_trees` is not specified.

`-use_default_routing_for_sinks n (default=_0)`

Forces the default routing rule to be used on the leaf nets that drive the clock tree sinks, and nets at the bottom $n-1$ levels of the clock tree. A net is considered as leaf net if it drives at least one flip-flop or latch. Float pins, exclude pins and `dont_touch_subtree` pins are treated as non-sink pins of the clock tree for this purpose. For example, a net that drives only one float pin will not be treated as leaf net for this purpose. The default is 0. This option is only active if the `-routing_rule` option is specified.

`-buffer_relocation true | false (default = true)`

Enables and disables buffer relocation during the clock tree optimization that is performed during initial CTS. Note that this option does not control buffer relocation performed during `optimize_clock_tree`. By default, buffer relocation is true. This option may either be set on specific clock trees, by using the `clock_trees` option, or on all clocks if `clock_trees` is not specified.

`-buffer_sizing true | false (default = true)`

Enables and disables buffer sizing during the clock tree optimization that is performed during initial CTS. Note that this option does not control buffer relocation performed during `optimize_clock_tree`. By default, buffer sizing is true. This option may either be set on specific clock trees, by using the `clock_trees` option, or on all clocks if `clock_trees` is not specified.

`-gate_relocation true | false (default = true)`
 Enables and disables gate relocation during the clock tree optimization that is performed during initial CTS. Note that this option does not control buffer relocation performed during `optimize_clock_tree`. By default, gate relocation is true. This option may either be set on specific clock trees, by using the `clock_trees` option, or on all clocks if `clock_trees` is not specified.

`-gate_sizing true | false (default = false)`
 Enables and disables gate sizing during the clock tree optimization that is performed during initial CTS. Note that this option does not control buffer relocation performed during `optimize_clock_tree`. By default, gate sizing is false. This option may either be set on specific clock trees, by using the `clock_trees` option, or on all clocks if `clock_trees` is not specified.

`-logic_level_balance true | false (default = false)`
 Enables building the clock tree with balanced logic levels. By default, logic level balance is false. This is a global option and will affect all clock trees.

`-ocv_clustering true | false (default = false)`
 Enables OCV-aware register clustering for capturing timing-critical endpoints, to improve clock path sharing between the relatively more timing-critical register pairs. Any previous setups, that can affect the timing measured between register pairs (such as `set_clock_latency` and `set_clock_uncertainty`), may affect clustering criteria, and accordingly affect resulting clock trees. By default, ocv clustering is false. This is a global option and will affect all clock trees.

`-config_file_read filename_(no_default)`
 Directs CTS to build the clock tree according to the configuration specified in the specified configuration file. This is a global option and will affect all clock trees

`-config_file_write filename_(no_default)`
 Directs CTS to write out the specified clock tree configuration file after CTS. This is a global option and will affect all clock trees

`-insert_boundary_cell true | false (default = false)`
 Enables the insertion of boundary cells near clock input ports for block-level designs. By default, insert boundary cell is false. This is a global option and will affect all clock trees.

`-operating_condition string_(default=_max)`
 Sets the operating condition at which CTS will synthesize the clock trees. By default, the operating condition is set to max. This is a global option and will affect all clock trees.

DESCRIPTION

The general flow for synthesizing a clock tree is as follows:

1. Trace the initial clock tree structure (based on the initial netlist and SDC commands). Based on the results, proceed to step 2 below, if needed.
2. Define clock tree exceptions (**`set_clock_tree_exceptions`**).
3. Specify the buffers and/or inverters to use in the synthesis and optimization of

the clock tree (**set_clock_tree_references**).

4. Synthesize the clock tree and update the design netlist with the legal placement results (**compile_clock_tree**).

A clock tree structure is defined from a pin of an instance called the source pin. It is assumed that the clock tree is to be constructed from this source pin until leaf pins of sequential elements are encountered. Exceptions are used to modify this assumption. After clock tree exceptions have been added or modified, the clock tree can be synthesized

At minimum, specify the source pin of the clock tree to create an initial clock tree structure. (If source pin is not specified, the command defines the settings for all clock trees in the design.) Once the definition of the clock tree has been created, other clock tree commands use the same source pin to do one or more of the following to the clock tree:

- Add constraints to it
- Add exceptions to it
- Report on it
- Synthesize it

The **-clock_trees** option defines the pin that is the root, or starting point, of the clock tree. It must be a pin of a leaf cell, a hierarchical port or an input port of the top level design.

The **set_clock_tree_references** command specifies the library buffers and/or inverters to use for clock tree synthesis.

Tracing Rules

The following section assumes familiarity with the concepts of timing arcs, timing arc unateness, combinational arcs, and sequential arcs. The following set of rules is applied to determine which clock tree nets in the transitive fanout of the driving pin are buffered, and which load pins of those nets are considered as balance points during clock tree synthesis. The general rule is to gather a set of clock tree nets by iteratively tracing through cells in the clock network. Tracing starts at the net driven by the root pin and continues through the owning cells of its load pins when they meet the following conditions:

- The output pin has at least one fanout.
- The output pin is not on a pad net, which is a net between an I/O pad cell and a top-level port, such as a bonding net.
- There is at least one combinational timing arc between the input pin on the clock path and the output pin or there is a sequential clock-to-output data arc and the related output is the source of a generated clock.
- There is no three-state enable or disable timing arc between the input pin on the clock path and the output pin.

If more than one output on a cell meets these conditions, clock tree synthesis

selects a random pin and generates a warning message indicating which pin was chosen. Use the **set_disable_timing** command to disable as many unwanted timing arcs in the cell as needed to force a specific output pin to be selected. When both sequential arcs (to generated clock source pins) and combinational arcs can be traced, additional warning messages are issued.

When clock tree synthesis traces through a cell to an output pin, the driven net becomes a clock tree net and the rules are iteratively applied to all its load pins to determine the other clock tree nets among the transitive fanout of this net.

Clock tree synthesis does not trace through a clock positive-edge arc, a clock negative-edge arc, or a three-state enable arc. All other combinational cell timing arcs are traced through. This includes timing arcs of multiplexer cells or combinational arcs from data pins of latch cells (transparent mode). Sequential clock gating cells are also traced through if there is a combinational arc connected to the clock path. By default, all nonclock pins of sequential elements, except three-state enable or disable pins, become exclude pins unless they have an explicit stop pin exception. For example, the following types of pins are implicitly excluded:

- Pins of combinational cells without fanout
- Pins of combinational cells with disabled timing arcs
- Top-level output ports
- Input pins of pad cells
- Data pins of flip flops or memories

The **-exceptions** option of **report_clock_tree** reports implicitly excluded pins as *inferred*.

Applying the **set_clock_tree_exceptions** command with appropriate options to some pins in the transitive fanout of the clock root modifies this default behavior.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets a target_skew only for for the clock_tree of "my_clock"

```
prompt> set_clock_tree_options \  
        -clock_trees my_clock -target_skew 0.050
```

The following example adds an exception to exclude a fanout pin Reg0/CP from examination for skew optimization and analysis. This will create a new clock tree structure that will take into account this new exception.:

```
prompt> set_clock_tree_exceptions -exclude_pins [get_pins Reg0/CP]
```

The following example adds an exception to define pin Ram0/we as a float pin.

```
prompt> set_clock_tree_exceptions -float_pins [get_pins RAM0/we]
```

The following example reports the current definition of the CLK1 clock tree:

```
prompt> report_clock_tree -clock_trees CLK1
```

SEE ALSO

```
compile_clock_tree(2)  
report_clock_tree(2)  
set_clock_tree_optimization_options(2)  
set_clock_tree_exceptions(2)  
set_clock_tree_references(2)  
set_disable_timing(2)
```

set_clock_tree_references

Specifies the buffers, inverters, and clock gates to be used in clock tree synthesis.

SYNTAX

```
status set_clock_tree_references  
[-clock_trees clock_name_or_collection]  
-references references  
[-sizing_only]  
[-delay_insertion_only]  
[-boundary_cell_only]
```

ARGUMENTS

-clock_trees *clock_name_or_collection*
Specifies a list of clock domains to apply the references. By default, the references apply to all clocks if you do not specify this option.

-references *references*
Specifies the list of buffers, inverters, and clock gates to build clock trees.

-sizing_only
Limits the usage of specified references for cell sizing only.

-delay_insertion_only
Limits the usage of specified references for delay insertion only.

-boundary_cell_only
Limits the usage of specified references for boundary cell insertion only.

DESCRIPTION

The command specifies buffers, inverters, and clock gates to be used for clock tree synthesis. Specifying several references allows the clock tree synthesis engine more flexibility to build clock trees and provides improved clock tree results.

If you do not specify the **-clock_trees** option, reference lists apply to all clocks in the design. You can use the **-clock_trees** option to apply reference lists on a per-clock basis. Reference lists specified with the **-clock_trees** option override the reference lists without the option.

You can specify clock gates as reference cells during optimization for cell sizing. To size a gate, equivalent gates from reference lists are used. If no equivalent gates are found, all gates from the target library are available for cell sizing.

If you issue the **-set_clock_tree_references** command several times for the same clock tree, the new references you specify are added to existing references for that clock tree. You can view the references of a given clock tree by using the **report_clock_trees** command. To delete references, use the

reset_clock_tree_references command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses library buffers for all clocks during clock tree synthesis.

```
prompt> set_clock_tree_references -references buffer4x
```

The following example uses library buffers for clock CLK1 in the design.

```
prompt> set_clock_tree_references -clock_trees CLK1 -references buffer4x
```

The following example uses library buffers for clock CLK1 in the design and limits its usage for delay insertion only

```
prompt> set_clock_tree_references -clock_trees CLK1 \  
-references buffer4x -delay_insertion_only
```

The following example uses library gate AND2X2 and ANDX4 for clock CLK1 in the design. To size any AND2 gates in the CLK1 path, either AND2X2 or AND2X4 gate can be used.

```
prompt> set_clock_tree_references -clock_trees CLK1 \  
-references "AND2X2 AND2X4" -sizing_only
```

SEE ALSO

```
set_clock_tree_options(2)  
set_clock_tree_exceptions(2)  
compile_clock_tree(2)  
report_clock_tree(2)
```

set_clock_uncertainty

Specifies the uncertainty (skew) of specified clock networks.

SYNTAX

```
string set_clock_uncertainty
[object_list
 | -from from_clock
 | -rise_from rise_from_clock
 | -fall_from fall_from_clock
-to to_clock
 | -rise_to rise_to_clock
 | -fall_to fall_to_clock]
[-rise]
[-fall]
[-setup]
[-hold]
uncertainty
```

Data Types

<i>object_list</i>	list
<i>from_clock</i>	list
<i>rise_from_clock</i>	list
<i>fall_from_clock</i>	list
<i>to_clock</i>	list
<i>rise_to_clock</i>	list
<i>fall_to_clock</i>	list
<i>uncertainty</i>	float

ARGUMENTS

object_list
Specifies a list of clocks, ports, or pins for simple uncertainty; the uncertainty is applied either to capturing latches clocked by one of the clocks in *object_list*, or capturing latches whose clock pins are in the fanout of a port or pin specified in *object_list*. You must specify either the pair of **-from** and **-to**, or *object_list*; you cannot specify both.

-from from_clock
This option specified the source clocks for interclock uncertainty. You must specify either the pair of **-from/-rise_from/-fall_from** and **-to/-rise_to/-fall_to**, or *object_list*; you cannot specify both.

-rise_from rise_from_clock
Same as the **-from** option, but indicates that *uncertainty* applies only to rising edge of the source clock. You can use only one of the **-from**, **-rise_from**, or **-fall_from** options.

-fall_from fall_from_clock
Same as the **-from** option, but indicates that *uncertainty* applies only to falling edge of the source clock. You can use only one of the **-from**, **-**

rise_from, or **-fall_from** options.

-to *to_clock*

This option specifies the destination clocks for interclock uncertainty. You must specify either the pair of **-from/-rise_from/-fall_from** and **-to/-rise_to/-fall_to**, or *object_list*; you cannot specify both.

-rise_to *rise_to_clock*

Same as the **-to** option, but indicates that *uncertainty* applies only to rising edge of the destination clock. You can use only one of the **-to**, **-rise_to**, or **-fall_to** options.

-fall_to *fall_to_clock*

Same as the **-to** option, but indicates that *uncertainty* applies only to falling edge of the destination clock. You can use only one of the **-to**, **-rise_to**, or **-fall_to** options.

-rise

Indicates that *uncertainty* applies to only the rising edge of the destination clock. By default, the uncertainty applies to both rising and falling edges. This option is valid only for interclock uncertainty, and is now obsolete. Unless you need this option for backward-compatibility, use **-rise_to** instead.

-fall

Indicates that *uncertainty* applies to only the falling edge of the destination clock. By default, the uncertainty applies to both rising and falling edges. This option is valid only for interclock uncertainty, and is now obsolete. Unless you need this option for backward-compatibility, use **-fall_to** instead.

-setup

Indicates that *uncertainty* applies only to setup checks. By default, *uncertainty* applies to both setup and hold checks.

-hold

Indicates that *uncertainty* applies only to hold checks. By default, *uncertainty* applies to both setup and hold checks.

uncertainty

A floating point number that specifies the uncertainty value. Typically, clock uncertainty should be positive. Negative uncertainty values are supported for constraining designs with complex clock relationships. Setting the uncertainty value to a negative number could lead to optimistic timing analysis and should be used with extreme care.

DESCRIPTION

Specifies the clock uncertainty (skew characteristics) of specified clock networks. This command can specify either interclock uncertainty or simple uncertainty. For interclock uncertainty, use the **-from/-rise_from/-fall_from** and **-to/-rise_to/-fall_to** options to specify the source clock and the destination clock; all paths between these receive the uncertainty value. For simple uncertainty, use *object_list*; the uncertainty value is either to capturing latches clocked by one of the clocks in *object list*, or capturing latches whose clock pins are in the fanout

of a port or pin specified in *object_list*.

Set the uncertainty to the worst skew expected to the endpoint or between the clock domains. You can increase the value to account for additional margin for setup and hold.

When you specify interclock uncertainty, ensure that you specify it for all possible interactions of clock domains. For example, if you specify paths from CLKA to CLKB and CLKB to CLKA you must specify the uncertainty for both even if the values are the same. For an example, see the EXAMPLES section.

Interlock uncertainty is more specific than simple uncertainty. If a command that specifies interlock uncertainty conflicts with a command that specifies simple uncertainty, the command that specifies interlock uncertainty takes precedence. For an example, see the EXAMPLES section.

If there is no applicable interlock uncertainty for a path, the value for simple uncertainty is used. For an example, see the EXAMPLES section.

To remove the uncertainties set by **set_clock_uncertainty**, use the **remove_clock_uncertainty** command.

To view clock uncertainty information, use the **report_clock -skew** command.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example specifies that all paths leading to registers or ports clocked by CLK have setup uncertainty of 0.65 and hold uncertainty of 0.45.

```
prompt> set_clock_uncertainty -setup 0.65 [get_clocks CLK]  
prompt> set_clock_uncertainty -hold 0.45 [get_clocks CLK]
```

The following example specifies interclock uncertainties between PHI1 and PHI2 clock domains.

```
prompt> set_clock_uncertainty 0.4 -from PHI1 -to PHI1  
prompt> set_clock_uncertainty 0.4 -from PHI2 -to PHI2  
prompt> set_clock_uncertainty 1.1 -from PHI1 -to PHI2  
prompt> set_clock_uncertainty 1.1 -from PHI2 -to PHI1
```

The following example specifies interclock uncertainties between PHI1 and PHI2 clock domains with specific edges.

```
prompt> set_clock_uncertainty 0.4 -rise_from PHI1 -to PHI2
```

```
prompt> set_clock_uncertainty 0.4 -fall_from PHI2 -rise_to PHI2
prompt> set_clock_uncertainty 1.1 -from PHI1 -fall_to PHI2
```

The following example shows conflicting **set_clock_uncertainty** commands, one for simple uncertainty and one for interclock uncertainty. The interclock uncertainty value of 2 takes precedence.

```
prompt> set_clock_uncertainty 5 [get_clocks CLKA]
prompt> set_clock_uncertainty 2 \
        -from [get_clocks CLKB] -to [get_clocks CLKA]
```

The following example specifies the uncertainty from CLKA to CLKB and from CLKB to CLKA. Notice that both must be specified even though the value is the same for both.

```
prompt> set_clock_uncertainty 2 \
        -from [get_clocks CLKA] -to [get_clocks CLKB]
prompt> set_clock_uncertainty 2 \
        -from [get_clocks CLKB] -to [get_clocks CLKA]
```

The following example illustrates a situation in which simple uncertainty is used when there is no applicable interclock uncertainty for a path. The first command specifies a simple uncertainty of 5 for CLKA paths, and the second command specifies an interclock uncertainty of 2 for paths from CLKB to CLKA. If there are paths between CLKA and other clocks (for example, CLKC, CLKD, ...) for which interclock uncertainty has not been specifically defined, the simple uncertainty (in this case, 5) is used.

```
prompt> set_clock_uncertainty 5 [get_clocks CLKA]
prompt> set_clock_uncertainty 2 \
        -from [get_clocks CLKB] -to [get_clocks CLKA]
```

SEE ALSO

```
current_design(2)
remove_clock_uncertainty(2)
report_clock(2)
set_clock_latency(2)
set_clock_transition(2)
```

set_combinational_type

Sets attributes on cell instances to specify which combinational cells from the target library are to be used by **compile**.

SYNTAX

```
int set_combinational_type  
-replacement_gate replacement_gate  
[cell_list]
```

Data Types

<i>replacement_gate</i>	string
<i>cell_list</i>	list

ARGUMENTS

-replacement_gate *replacement_gate*
Specifies a combinational gate from the target_library to use by **compile** as the exact replacement gate for the cells in the cell list. Sets **combinational_type_exact** to the *replacement_gate* on all cells in *cell_list*. This argument is required.

cell_list
Specifies a list of cells in which to use the replacement combinational gate.

DESCRIPTION

The **set_combinational_type** command specifies replacement combinational gate information for **compile** to use by setting attributes on the cell instances. Specifying **-replacement_gate** sets the **combinational_type_exact** attribute to *replacement_gate*, indicating to use this attribute as the replacement gate for those cells.

In the mapping process for combinational gates, **compile** attempts to convert combinational gates tagged by **set_combinational_type** to the specified replacement combinational gate. In the absence of attributes on a combinational cell instance to control the mapping, **compile** chooses the one best for timing, power, or area.

After mapping the cell instances to the replacement gate, **compile** sets the **dont_touch** attribute on these cell instances to disable further optimization on these cells.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example indicates that the replacement combinational gate for cell U1 is cell AN2P. This sets the **combinational_type_exact** attribute to 'AN2P' on the cell 'U1'. The presence of this attribute indicates that cell 'U1' attempts an exact mapping to gate 'AN2P'.

```
prompt> set_combinational_type -replacement_gate AN2P U1
```

SEE ALSO

```
current_design(2)  
get_attribute(2)  
remove_attribute(2)  
reset_design(2)  
target_library(3)
```

set_congestion_options

Sets options for congestion optimization.

SYNTAX

```
int set_congestion_options  
[-max_util value]  
[-layer name]  
[-availability value]  
[-coordinate {X1 Y1 X2 Y2}]
```

Data Types

<i>value</i>	float
<i>name</i>	string

ARGUMENTS

-max_util *value*
Specifies the maximum utilization factor.

-layer *name*
Specifies the layer name whose availability is reduced.

-availability *value*
Specifies the availability of the routing resource for the layer

-coordinate {*X1 Y1 X2 Y2*}
Specifies the lower left and upper right coordinates for which the congestion options will apply. The numbers are in microns.

DESCRIPTION

The **set_congestion_options** command sets congestion options for the current design. Congestion occurs because the number of wires going through a region exceeds the capacity of that region. If the ratio of usage-to-capacity is larger than 1, the region is congested.

The **-max_util** option specifies how densely the tool can pack cells in uncongested regions to remove congestion in congested regions. You should set this variable based on how much placeable area is needed. The default maximum utilization is 0.95.

The **-layer** and **-availability** options specify how much of the routing resource for the given layer is available to be used. For example, in a design whose M5 and M6 will be 70% occupied by future P/G routing, you can set the availability of M5 and M6 to be 0.30. This means even if currently there is no route in M5 and M6, we will consider them to be 70% full. If you don't want to use M5 and M6 for signal routing at all, you can use **set_ignored_layer** to mark them as ignored. However, it is still desired to specify the availability in this case, because we still need to know the existence of these future nets to perform a good estimation on the coupling effects of them.

Congestion options can be design-wide or regional. If the **-coordinate** option is not specified, the values are design wide. If **-coordinate** option is specified, the values will only be applied to the bounding box specified by with the option. Regional congestion options will be assigned an ID, which can be used to report or remove the regional options.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example runs the **set_congestion_options** command.

```
prompt> set_congestion_options -layer METAL5 -availability 0.5
prompt> set_congestion_options -layer METAL5 -availability 0.7 \
        -coordinate {0 0 10 10}
prompt> set_congestion_options -layer METAL5 -max_util 0.9
```

SEE ALSO

```
current_design(2)
remove_congestion_options(2)
report_congestion_options(2)
report_congestion(2)
set_ignored_layers(2)
```

set_connection_class

Sets the connection class value on ports.

SYNTAX

```
status set_connection_class  
connection_class_value  
object_list
```

Data Types

<i>connection_class_value</i>	string
<i>object_list</i>	list

ARGUMENTS

connection_class_value
Specifies the desired *connection_class* value of ports contained in *object_list*. The *connection_class_value* is a single string value. This string can contain any number of space-separated connection classes (see the example below).

object_list
Specifies ports whose connection classes are set.

DESCRIPTION

Connection classes are placed on ports of designs in a technology library by using Library Compiler. The **set_connection_class** command places the *connection_class* attribute on ports of designs being optimized (such as the current design). This is used to indicate a connection class requirement for the network that is connected to the specified port.

The "connection class" label is an attribute that is used to describe connection requirements for a given technology. Only those loads and drivers with the same connection class label can be legally connected. The labels "universal" and "default" are reserved labels. The "universal" label indicates that a pin or port can legally connect with any other load or driver. The "default" label is used for those library pins that do not otherwise have a connection class assigned to them through any library attributes. For designs being optimized, the "universal" label is assigned to those ports that do not otherwise have a connection class assigned to them. To change the default connection class label for those ports, use the variable **default_port_connection_class**.

The connection requirements specified using connection classes are treated as Design Rule Constraints by the Design Compiler. Optimization attempts to build designs that meet connection class requirements even at the expense of other constraints on the design (such as area or power).

To view connection class values on ports, use the **report_port** or **get_attribute** command. To see the default connection class value for a library, use the **report_lib**

command. To check your design for connection class violations, use the **check_design** command. To reset a connection class value, use the **remove_attribute** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets a connection class "internal" to the port named "xyz" in the current design:

```
prompt> set_connection_class internal xyz
```

The following example sets the connection classes "internal" and "external" on the port "out" in the design "test":

```
prompt> set_connection_class "internal external" test/out
```

In the following example, the connection class on a port is removed:

```
prompt> remove_attribute [get_ports xyz] connection_class
```

SEE ALSO

```
all_outputs(2)
remove_attribute(2)
report_lib(2)
report_port(2)
reset_design(2)
current_design(3)
default_port_connection_class(3)
target_library(3)
```

set_context_margin

Specifies the margin by which to tighten or relax constraints.

SYNTAX

```
string set_context_margin  
[-percent]  
[-relax]  
[-min]  
[-max]  
value  
[object_list]
```

Data Types

<i>value</i>	float
<i>object_list</i>	list

ARGUMENTS

-percent	Indicates that the specified value is a percentage of the delay.
-relax	Relaxes the constraint.
-min	Specifies the margin for minimum constraints.
-max	Specifies the margin for maximum constraints.
value	Determines the margin in absolute value or percentage value.
object_list	Specifies a list of cells or pins.

DESCRIPTION

The **set_context_margin** command specifies a margin to add to or to subtract from input and output delay values when the values are generated by the **allocate_fp_budgets** command. The margin can be specified as an absolute value or as a percentage of the constraint.

The constraints are created using the **characterize** or **allocate_fp_budgets** command.

By default, the input and output delays are adjusted so that they are more constraining. This means that the specified margin is added to the maximum delay values and subtracted from the minimum delay values. If you specify the **-relax** option, the margin is subtracted from the maximum delay values and added to the

minimum delay values.

The margin applies to the current design when no object list is specified. When determining the margin for a pin, the value set for the pin is used. If you do not specify a value for the pin, the value set for the parent cell is used. If neither value is set, the value set for the current design is used. If no margin is specified, the default value is 0.0

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command adds a margin of 0.5 to all maximum values of input and output delay constraints and subtracts the margin from all minimum values of input and output delay constraints. The margin applies to all objects in the current design.

```
prompt> set_context_margin 0.5
```

The following command relaxes (or reduces) the maximum value input delays on *I2/IN* by 10 percent:

```
prompt> set_context_margin -relax -max -percent 10.0 I2/IN
```

SEE ALSO

`characterize(2)`
`allocate_fp_budgets(2)`
`write_script(2)`

set_cost_priority

Sets the `cost_priority` attribute to a specified value on the current design.

SYNTAX

```
int set_cost_priority  
[-default]  
[-delay]  
cost_list  
[-design_rules]  
[-min_delay]
```

Data Types

cost_list list

ARGUMENTS

-default

Removes the **cost_priority** attribute so that the **compile** command uses its default priority.

-delay

Specifies that **max_delay** has higher priority than the max design rules.

cost_list

Specifies a list of costs in decreasing order of priority, selected from the following: **max_delay**, **min_delay**, **max_transition**, **max_fanout**, **max_capacitance**, **cell_degradation**, and **max_design_rules**.

-design_rules

Specifies that **max_design_rules** cost has higher priority than the max delay cost.

-min_delay

Specifies that **min_delay** has higher priority than the **max_delay**, but lower priority than max design rules.

DESCRIPTION

The **set_cost_priority** command sets the **cost_priority** attribute on the current design. This attribute changes the precedence that **compile** uses when different constraints conflict with each other. For example, by default the design rule **max_transition** has priority over **max_delay**, meaning that if both constraints cannot be met, **compile** fixes the transition violation at the expense of delay.

If a list of costs is given, any costs that are not in the list are assumed to follow afterwards in their default relative priority.

If **set_cost_priority** is specified more than once on a design, the most recent setting is used and the earlier values are removed.

To undo **set_cost_priority**, use the **-default** option, the **remove_attribute** command, or the **reset_design** command.

Use the **report_constraints** command to obtain a summary of the constraints of the design.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets the priority so that design rule violations are not fixed if they increase max delay cost:

```
prompt> set_cost_priority -delay
```

The following example sets the priority so that **max_fanout** and **max_capacitance** have higher priority than delay, but **max_transition** and **cell_degradation** have lower priority:

```
prompt> set_cost_priority {max_fanout max_capacitance max_delay}
```

SEE ALSO

`current_design(2)`
`remove_attribute(2)`
`report_constraint(2)`
`reset_design(2)`

set_critical_range

Sets the **critical_range** attribute to a specified value on a list of designs.

SYNTAX

```
int set_critical_range  
range_value designs
```

Data Types

<i>range_value</i>	float
<i>designs</i>	list

ARGUMENTS

<i>range_value</i>	Specifies the value to which the critical_range attribute is to be set.
<i>designs</i>	Indicates the list of designs to which the critical_range attribute applies.

DESCRIPTION

Sets the **critical_range** attribute to *range_value* on the designs in **designs**.

The **compile** command uses the **critical_range** attribute of the top-level design as the default critical range for path groups that do not have a critical range set. If you do not set the **critical_range** attribute, such path groups get a critical range of 0.0. You can assign critical range values to individual path groups with the **group_path -critical_range** command.

Critical range specifies a margin of delay for path groups in optimization. This must be a positive float number or 0.0. A critical range of 0.0 means that only the most critical paths (the ones with the worst violation) are optimized. If you specify a nonzero critical range, near-critical paths within that amount of the worst path will also be optimized if possible.

To undo **set_critical_range**, use **remove_attribute** or **reset_design**.

To show the critical range values for each path group, use **report_path_group**. To show the critical range cost of the design, use **report_constraint**.

For some backward compatibility, if **compile** is run on a design with no **critical_range** attribute and the obsolete variable **compile_default_critical_range** is set, the attribute is created and set to the same value on the design. The attribute always has precedence over the variable setting. If **compile** is run when the attribute exists and has a different value than the variable, a warning is printed that the variable is being ignored. Consequently, if the value for critical range optimization has to be changed between two **compile** commands on the same design, the **set_critical_range** command must be used since changes to the **compile_default_critical_range** variable after the first compile will be ignored.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets the default critical range value for design "top" to be 10.0.

```
prompt> set_critical_range 10.0 top
```

SEE ALSO

```
current_design(2)  
group_path(2)  
remove_attribute(2)  
report_constraint(2)  
report_path_group(2)  
reset_design(2)
```

set_cts_batch_mode

Enables clock tree synthesis batch mode.

SYNTAX

```
status set_cts_batch_mode
```

ARGUMENTS

None.

DESCRIPTION

To reduce runtime in script that call CTS commands many times, in some cases, one clock at a time, this command turns on the batch mode. It skips CTS pre and post process whenever it can. For example, if 3 `compile_clock_tree` commands are called, the first call will run the CTS pre-process, but skipped the post process. The second and the third calls will skip both pre and post process. The following command, like `reset_cts_batch_mode`, `save_mw_cel` or other trigger commands will make up the CTS post process.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

```
set_cts_batch_mode
```

```
compile_clock_tree -clock_trees clk1
```

```
compile_clock_tree -clock_trees clk2
```

```
report_cts_batch_mode  
optimize_clock_tree  
balance_inter_clkok_delay  
reset_cts_batch_mode
```

SEE ALSO

```
reset_cts_batch_mode(2)  
report_cts_batch_mode(2)
```


set_cts_scenario

Sets the specified scenario as the clock tree synthesis scenario.

SYNTAX

```
string set_cts_scenario  
[scenario_name]
```

Data Types

scenario_name string

ARGUMENTS

scenario_name
 Specifies the name of the scenario to be marked as the clock tree synthesis scenario.

DESCRIPTION

This command sets the specified scenario as the clock tree synthesis scenario and returns the name of the clock tree synthesis scenario. The scenario must exist before you run this command. In the multicorner-multimode (MCMM) flow, the **compile_clock_tree**, **optimize_clock_tree**, and **balance_inter_clock_delay** commands work on the clock tree synthesis scenario (if defined), or on the current scenario if the clock tree synthesis scenario is not defined.

Multicorner-Multimode Support

This command uses information from active scenarios only.

EXAMPLES

The following example uses **set_cts_scenario** to set the clock tree synthesis scenario:

```
prompt> create_scenario MODE1  
prompt> set_cts_scenario MODE1  
MODE1  
prompt> get_cts_scenario  
MODE1  
prompt> remove_cts_scenario  
1  
prompt>get_cts_scenario  
  
prompt>
```

SEE ALSO

`balance_inter_clock_delay(2)`
`clock_opt(2)`
`compile_clock_tree(2)`
`get_cts_scenario(2)`
`optimize_clock_tree(2)`
`remove_cts_scenario(2)`

set_current_command_mode

SYNTAX

string **set_current_command_mode**
-mode *command_mode* | -command *command*

Data Types

<i>command_mode</i>	string
<i>command</i>	string

ARGUMENTS

-mode *command_mode*
Specifies the name of the new command mode to be made current. **-mode** and **-command** are mutually exclusive; you must specify one, but not both.

-command *command*
Specifies the name of the command whose associated command mode is to be made current. **-mode** and **-command** are mutually exclusive; you must specify one, but not both.

DESCRIPTION

The **set_current_command_mode** sets the current command mode. If there is a current mode in effect, the current mode is first cancelled, causing the associated clean up to be executed. The initialization for the new command mode is then executed as it is made current.

If the name of the given command mode is empty, the current command mode is cancelled and no new command mode is made current.

A current command mode stays in effect until it is displaced by a new command mode or until it is cleared by an empty command mode.

If the command completes successfully, the new current command mode name is returned. On failure, the command returns the previous command mode name which will remain current and prints an error message unless error messages are suppressed.

EXAMPLES

The following example sets the current command mode to a mode called "mode_1"

```
prompt> set_current_command_mode -mode mode_1
```

The following example clears the current command mode without setting a new mode.

```
prompt> set_current_command_mode -mode ""
```

The following example sets the current command mode to the mode associated with the command "modal_command"

```
prompt> set_current_command_mode -command modal_command
```

set_data_check

Sets data-to-data checks using the specified values of setup and hold time.

SYNTAX

```
string set_data_check  
-from from_object  
    | -rise_from from_object  
    | -fall_from from_object  
-to to_object  
    | -rise_to to_object  
    | -fall_to to_object  
[-setup | -hold]  
[-clock clock_object]  
[check_value]
```

Data Types

<i>from_object</i>	object
<i>to_object</i>	object
<i>clock_object</i>	object
<i>check_value</i>	float

ARGUMENTS

-from *from_object*
Specifies a pin or port in the current design as the related pin of the data-to-data check to be set. Both rising and falling delays are checked. You must specify one of **-from**, **-rise_from**, or **-fall_from**.

-rise_from *from_object*
Similar to the **-from** option, but applies only to rising delays at the related pin. You must specify one of **-from**, **-rise_from**, or **-fall_from**.

-fall_from *from_object*
Similar to the **-from** option, but applies only to falling delays at the related pin. You must specify one of **-from**, **-rise_from**, or **-fall_from**.

-to *to_object*
Specifies a pin or port in the current design as the constrained pin of the data-to-data check to be set. Both rising and falling delays are constrained. You must specify one of **-to**, **-rise_to**, or **-fall_to**.

-rise_to *to_object*
Similar to the **-to** option, but applies only to rising delays at the constrained pin. You must specify one of **-to**, **-rise_to**, or **-fall_to**.

-fall_to *to_object*
Similar to the **-to** option, but applies only to falling delays at the constrained pin. You must specify one of **-to**, **-rise_to**, or **-fall_to**.

-setup
Indicates that the data check value is for setup analysis only. If neither **-setup** nor **-hold** is specified, the value applies to both setup and hold.

-hold
Indicates that the data check value is for hold analysis only. If neither **-setup** nor **-hold** is specified, the value applies to both setup and hold.

-clock *clock_object*
Specifies the name of a single clock that launches the signal for related pin of the data check.

check_value
Specifies the value of the setup and/or hold time for the check.

DESCRIPTION

The **set_data_check** command specifies a data-to-data check to be performed between the from object and to object using the specified setup and/or hold value.

The pulse relation between clocks of related pin and constrained pin is considered to be zero cycles. The normal sequential check uses one cycle. Data check on clock pin is not supported.

The data check is treated as non-sequential; that is, the path goes through the related and constrained pins and is not broken at these pins. To report the constraint related to the data check, use **report_timing -to data_check_constrained_pin**.

To remove information set by **set_data_check**, use the **remove_data_check** command.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example creates a data-to-data check from and1/B to and1/A with respect to the rising edge of signal at and1/B, using a setup and hold time of 0.4.

```
prompt> set_data_check -rise_from and1/B -to and1/A 0.4
```

The following example creates a data-to-data check from and1/B to and1/A with respect to the rising edge of signal at and1/B, coming from the clock domain of CK1, constraining only the falling edge of signal at and1/A, using a setup time of 0.5 and no hold check.

```
prompt> set_data_check -rise_from and1/B -fall_to and1/A -setup \  
-clock [get_clock CK1] 0.5
```

SEE ALSO

`remove_data_check(2)`
`report_timing(2)`
`update_timing(2)`
`timing_enable_multiple_clocks_per_reg(3)`

set_default_drive

Sets the default driving strength for specified objects, to be used by Top-Down Environmental Propagation (TDEP).

SYNTAX

```
status set_default_drive
[-min]
[-max]
[-rise]
[-fall]
[-none]
[resistance]
[cell_or_pin_list]
```

Data Types

<i>resistance</i>	float
<i>cell_or_pin_list</i>	list

ARGUMENTS

-min	Indicates that <i>resistance</i> is to apply only to minimum resistance.
-max	Indicates that <i>resistance</i> is to apply only to maximum resistance. (This is the default if no -min or -max is specified and also if both options are given.)
-rise	Indicates that <i>resistance</i> is to apply only to rise resistance.
-fall	Indicates that <i>resistance</i> is to apply only to fall resistance.
-none	Indicates that previous information set by set_default_drive is to be removed.
<i>resistance</i>	Specifies the resistance value; allowed values are any nonnegative floating point number.
<i>cell_or_pin_list</i>	Specifies a list of names of cells or pins for which the default driving strength is to be set.

DESCRIPTION

This command specifies a default drive value, to be used later by Top-Down

Environmental Propagation (TDEP).

Note that you can specify both max and min values by issuing two separate commands, one with **-max** and one with the **-min** option. However, if you specify both or none of these two options only the max value will be set and the min value will be removed. To always remove both values use the **-none** option.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets default drives and then removes them by using the **-none** option.

```
prompt> set_default_drive -rise 5 U_L1
Performing set_default_drive on cell 'U_L1'.
1
prompt> set_default_drive -rise -min 7 U_L1
Performing set_default_drive on cell 'U_L1'.
1
prompt> set_default_drive -fall 6 U_L1
Performing set_default_drive on cell 'U_L1'.
1
prompt> set_default_drive -fall -min 8 U_L1
Performing set_default_drive on cell 'U_L1'.
1
prompt> set_default_drive -none
1
```

SEE ALSO

```
derive_constraints(2)
set_default_driving_cell(2)
set_default_fanout_load(2)
set_default_load(2)
```

set_default_driving_cell

Sets the default driving cell for specified objects, to be used by Top-Down Environmental Propagation (TDEP).

SYNTAX

```
int set_default_driving_cell  
[-lib_cell lib_cell_name]  
[-library lib]  
[-rise] [-fall]  
[-pin pin_name]  
[-from_pin from_pin_name]  
[-dont_scale] [-no_design_rule]  
[-multiply_by factor] [-none]  
cell_or_pin_list
```

Data Types

<i>lib_cell_name</i>	string
<i>lib</i>	string
<i>pin_name</i>	string
<i>from_pin_name</i>	string
<i>factor</i>	float
<i>cell_or_pin_list</i>	list

ARGUMENTS

-lib_cell *lib_cell_name*

Specifies the name of a library cell to be used to drive the ports; both the **driving_cell_rise** and the **driving_cell_fall** string attributes are set to to *lib_cell_name* on the ports. If the cell has more than one output pin, you must also use the **-pin** option. To specify different cells for the rising and falling cases, execute the command twice, once with **-rise** and once with **-fall**, using the appropriate *lib_cell_name* for each.

-library *lib*

You must use this option with the **-lib_cell** option. Specifies the library in which to find *library_cell_name*. This is either a library name or a collection. By default, the libraries in **link_library** are searched for the cell. To specify different libraries for the rising and falling cases, execute the command twice, once with **-rise** and once with **-fall**, using the appropriate *lib* for each.

-rise

Indicates that the **lib_cell_name**, **lib**, **pin_name**, and **from_pin_name** correspond to the rising case. You can use this option with **-fall** to specify both the rising and the falling case.

-fall

Indicates that the **lib_cell_name**, **lib**, **pin_name**, and **from_pin_name** correspond to the falling case. You can use this option with **-rise** to specify both the rising and the falling case.

-pin *pin_name*

You must use this option with the **-lib_cell** option. Specifies the output pin on the driving cell that is to drive the ports; needed if the driving cell has more than one output pin, or if the **-from_pin** option is used. The default is to use the first timing arc found on the library cell. To specify different pins for the rising and falling cases, execute the command twice, once with **-rise** and once with **-fall**, using the appropriate *pin_name* for each.

-from_pin *from_pin_name*

You must use this option with both **-pin** and **-lib_cell**. Specifies the name of the input pin on the driving cell that is to be used when finding a timing arc; needed if the driving cell has more than one input pin on the driving cell and the arcs from those pins have different drive characteristics. The default is to use the first timing arc found on the library cell.

-dont_scale

You must use this option with the **-lib_cell** option. Indicates that the timing analyzer is not to scale the drive capability of the ports according to the current operating conditions. By default, the port drive capability is scaled for operating conditions exactly as the driving cell itself would have been scaled.

-no_design_rule

Indicates that the design rules associated with the driving cell are not to be applied to the driven port. Timing-related attributes are still applied. By default, design rule attributes (*max_fanout*, *max_capacitance*, *max_transition*, *min_fanout*, *min_capacitance*, *min_transition*) are derived from the driving cell and its library and applied to the port.

-multiply_by *factor*

You must use this option with the **-lib_cell** option. Specifies a factor by which to multiply the delay characteristics of the ports; the default is 1.0. Both the load delay and the transition times of the port are affected.

-none

Delete previous *set_default_driving_cell* info.

cell_or_pin_list

Specifies a list of names of cells or pins to which the information applies.

DESCRIPTION

This command sets a default driving cell for specified cells or pins; this driving cell will later be used by Top-Down Environmental Propagation (TDEP).

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets the default driving cell and then removes it using the **-none** option.

```
prompt>set_default_driving_cell -lib_cell ND2 -library lsi_10k \  
-pin Z -from_pin A -rise -dont_scale -mult 2.3;  
1  
prompt>set_default_driving_cell -lib_cell IVP -library lsi_10k \  
-pin Y -from_pin B -fall;  
1  
prompt>set_default_driving_cell -none;  
1
```

SEE ALSO

```
derive_constraints(2)  
set_default_drive(2)  
set_default_fanout_load(2)  
set_default_load(2)
```

set_default_fanout_load

Sets the default fanout load to be used by Top-Down Environmental Propagation (TDEP).

SYNTAX

```
status set_default_fanout_load  
[-none]  
[fanout_load_value]  
[cell_or_pin_list]
```

Data Types

<i>fanout_load_value</i>	float
<i>cell_or_pin_list</i>	list

ARGUMENTS

-none
Indicates that previous information set by **set_default_fanout_load** is to be removed.

fanout_load_value
Specifies the fanout load value; allowed values are any nonnegative floating point number.

cell_or_pin_list
Specifies a list of names of cells or pins for which the default fanout load is to be set.

DESCRIPTION

This command specifies a default fanout load value, to be used later by Top-Down Environmental Propagation (TDEP).

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets a default load and then removes it by using the **-none** option.

```
prompt> set_default_fanout_load 2 U_L1/U10  
Performing set_default_fanout_load on cell 'U_L1/U10'.  
1  
prompt> set_default_fanout_load -none
```

SEE ALSO

```
derive_constraints(2)  
set_default_drive(2)  
set_default_load(2)  
set_default_driving_cell(2)
```

set_default_input_delay

Sets the value of the input delay as a percentage of the clock period to be assigned during environment propagation.

SYNTAX

```
int set_default_input_delay  
[-none]  
percent_delay  
[cell_or_pin_list]
```

Data Types

<i>percent_delay</i>	float
<i>cell_or_pin_list</i>	collection

ARGUMENTS

-none
Resets any existing default delays.

percent_delay
Specifies the delay as a percentage of the clock period. Substitute the value you want for *percent_delay*.

cell_or_pin_list
Specifies the list of cells or pins on which the input delay is to be set. Substitute the list you want for *cell_or_pin_list*.

DESCRIPTION

Sets the value of the input delay as a percentage of the clock period to be assigned during environment propagation. By default, a value of 30 is globally used. This value can be overridden on a global basis, on a cell by cell basis, or on a pin-by-pin basis. If no arguments are specified, it sets the global default to the new value. The environment propagation first looks for any values set on the pin. If none exists, it looks at the cell of the pin. If none exists, the global default is used.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

SEE ALSO

set_default_output_delay(2)

set_default_load

Sets the default load to be used by Top-Down Environmental Propagation (TDEP).

SYNTAX

```
status set_default_load
[-min]
[-max]
[-pin_load]
[-wire_load]
[-none]
[value]
[cell_or_pin_list]
```

Data Types

<i>value</i>	float
<i>cell_or_pin_list</i>	list

ARGUMENTS

-min	Indicates that <i>value</i> is to apply only to minimum capacitance.
-max	Indicates that <i>value</i> is to apply only to maximum capacitance. (This is the default if no -min or -max is specified and also if both options are given.)
-pin_load	Indicates that <i>value</i> is to apply only to pin capacitance.
-wire_load	Indicates that <i>value</i> is to apply only to wire capacitance.
-none	Indicates that previous information set by set_default_load is to be removed.
<i>value</i>	Specifies the capacitance value; allowed values are any nonnegative floating point number.
<i>cell_or_pin_list</i>	Specifies a list of names of cells or pins for which the default load is to be set.

DESCRIPTION

This command specifies a default load value, to be used later by Top-Down Environmental Propagation (TDEP).

Note that you can specify both max and min values by issuing two separate commands,

one with **-max** and one with the **-min** option. However, if you specify both or none of these two options only the max value will be set and the min value will be removed. To always remove both values use the **-none** option.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets a default load and then removes it by using the **-none** option.

```
i
prompt>set_default_load -wire -min 4
1
prompt>set_default_load -none
1
```

SEE ALSO

```
derive_constraints(2)
set_default_drive(2)
set_default_driving_cell(2)
set_default_fanout_load(2)
```

set_default_output_delay

Sets the output delay as a percentage of the clock period to be assigned during environment propagation.

SYNTAX

```
int set_default_output_delay  
[-none]  
percent_delay  
[cell_or_pin_list]
```

Data Types

<i>percent_delay</i>	float
<i>cell_or_pin_list</i>	collection

ARGUMENTS

-none
Reset any existing default delays.

percent_delay
The delay as the percentage of the clock period.

cell_or_pin_list
The list of cells or pins on which the output delay is to be set.

DESCRIPTION

This command sets the value of the output delay as a percentage of the clock period to be used during environment propagation. By default, a value of 30 is used globally. This value can be over-ridden on a global basis, on a cell by cell basis, or on a pin-by-pin basis. If no arguments are specified, it will set the global default to the new value. The environment propagation will first look for any values set on the pin. If none exists, it will look at the cell of the pin. If none exists, the global default will be used.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

SEE ALSO

set_default_input_delay(2)

set_delay_calculation

Defines the delay model used to compute a timing arc delay value for a cell or net.

SYNTAX

```
int set_delay_calculation  
[-arnoldi]  
[-elmore]  
[-clock_arnoldi]
```

ARGUMENTS

-arnoldi

Specifies that the delay calculation must use the accurate Arnoldi-based delay model. If you do not specify this option, the Elmore delay model will be used for net delays and total capacitance will be used for cell delay computation.

-elmore

This is the default behavior. Specifies that the Elmore delay model will be used for net delays and total capacitance will be used for cell delay computation. This option cannot be used with the **-arnoldi** option.

-clock_arnoldi

This specifies the delay calculation to use Arnoldi-based delay models on clock nets in a post-CTS or a post-Route design.

DESCRIPTION

The **set_delay_calculation** command specifies timing calculation models to be used for delay computation. Delays are computed via the Elmore delay model (option '-elmore') or the Arnoldi delay model (option '-arnoldi') based on parasitics information for nets of the current design. To provide parasitics information you can use fextract_rc or fread_parasitics. If parasitics information is not provided delays will be computed using the wire-load model.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command removes all annotated delays and specifies Arnoldi-based model to be used for delay calculation.

```
prompt> set_delay_calculation -arnoldi  
The following command computes and annotates Elmore delays.  
prompt> set_delay_calculation -elmore
```

SEE ALSO

`extract_rc(2)`
`read_parasitics(2)`
`report_delay_calculation(2)`
`report_timing(2)`

set_delay_estimation_options

Sets the parameters that influence preroute delay estimation.

SYNTAX

```
int set_delay_estimation_options
[-min_unit_horizontal_capacitance float]
[-min_unit_vertical_capacitance float]
[-min_unit_horizontal_resistance float]
[-min_unit_vertical_resistance float]
[-max_unit_horizontal_capacitance float]
[-max_unit_vertical_capacitance float]
[-max_unit_horizontal_resistance float]
[-max_unit_vertical_resistance float]
[-min_unit_horizontal_capacitance_scaling_factor float]
[-min_unit_vertical_capacitance_scaling_factor float]
[-min_unit_horizontal_resistance_scaling_factor float]
[-min_unit_vertical_resistance_scaling_factor float]
[-max_unit_horizontal_capacitance_scaling_factor float]
[-max_unit_vertical_capacitance_scaling_factor float]
[-max_unit_horizontal_resistance_scaling_factor float]
[-max_unit_vertical_resistance_scaling_factor float]
[-min_via_resistance float_resistance]
[-max_via_resistance float_resistance]
[-min_via_resistance_scaling_factor float]
[-max_via_resistance_scaling_factor float]
[-default]
```

ARGUMENTS

- min_unit_horizontal_capacitance *float*
Specifies the minimum horizontal capacitance per unit length for delay calculation.
- min_unit_vertical_capacitance *float*
Specifies the minimum vertical capacitance per unit length for delay calculation.
- min_unit_horizontal_resistance *float*
Specifies the minimum horizontal resistance per unit length for delay calculation.
- min_unit_vertical_resistance *float*
Specifies the minimum vertical resistance per unit length for delay calculation.
- max_unit_horizontal_capacitance *float*
Specifies the maximum horizontal capacitance per unit length for delay calculation.
- max_unit_vertical_capacitance *float*
Specifies the maximum vertical capacitance per unit length for delay calculation.

calculation.

`-max_unit_horizontal_resistance float`

Specifies the maximum horizontal resistance per unit length for delay calculation.

`-max_unit_vertical_resistance float`

Specifies the maximum vertical resistance per unit length for delay calculation.

`-min_unit_horizontal_capacitance_scaling_factor float`

Specifies the factor that will be multiplied by the minimum horizontal capacitance per unit length before use in delay calculation.

`-min_unit_vertical_capacitance_scaling_factor float`

Specifies the factor that will be multiplied by the minimum vertical capacitance per unit length before use in delay calculation.

`-min_unit_horizontal_resistance_scaling_factor float`

Specifies the factor that will be multiplied by the minimum horizontal resistance per unit length before use in delay calculation.

`-min_unit_vertical_resistance_scaling_factor float`

Specifies the factor that will be multiplied by the minimum vertical resistance per unit length before use in delay calculation.

`-max_unit_horizontal_capacitance_scaling_factor float`

Specifies the factor that will be multiplied by the maximum horizontal capacitance per unit length before use in delay calculation.

`-max_unit_vertical_capacitance_scaling_factor float`

Specifies the factor that will be multiplied by the maximum vertical capacitance per unit length before use in delay calculation.

`-max_unit_horizontal_resistance_scaling_factor float`

Specifies the factor that will be multiplied by the maximum horizontal resistance per unit length before use in delay calculation.

`-max_unit_vertical_resistance_scaling_factor float`

Specifies the factor that will be multiplied by the maximum vertical resistance per unit length before use in delay calculation.

`-min_via_resistance float_resistance`

Specifies the minimum via resistance for delay calculation.

`-max_via_resistance float_resistance`

Specifies the maximum via resistance for delay calculation.

`-min_via_resistance_scaling_factor float`

Specifies the factor that will be multiplied by the minimum via resistance before use in delay calculation.

`-max_via_resistance_scaling_factor float`

Specifies the factor that will be multiplied by the maximum via resistance before use in delay calculation.

`set_delay_estimation_options`

-default

Resets all per-unit resistance and capacitance scaling factors and all resistance and capacitance scaling factors to default values.

DESCRIPTION

The **set_delay_estimation_options** command specifies the parameters that influence preroute RC and delay estimation. Typical values for the scaling factors are between 0.5 and 5.0. The unit distance is 1 micron. Specifying **-default** restores all per-unit resistances and capacitances to library-derived values and all the resistance and capacitance scaling factors to 1.0.

The design must be read before applying the delay estimation options using this command.

The scaling factors are scenario aware. When you create a scenario or set a current scenario, you can at the same time create scaling factors using this command for the current scenario.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example runs the **set_delay_estimation_options** command.

```
prompt> set_delay_estimation_options \  
-max_unit_vertical_resistance 0.00004 \  
-min_unit_horizontal_capacitance_scaling_factor 2.0 \  
-min_via_resistance 0.00001 \  
-min_via_resistance_scaling_factor 1.5 \  
-max_via_resistance 0.00002 \  
-max_via_resistance_scaling_factor 1.5
```

SEE ALSO

report_delay_estimation_options(2)

set_design_license

Adds license information to the current design and can be used to require a license before a design can be read in.

SYNTAX

```
status set_design_license  
[-dont_show references]  
[-quiet]  
[-limited limited_keys]  
regular_keys
```

Data Types

<i>references</i>	list
<i>limited_keys</i>	list
<i>regular_keys</i>	list

ARGUMENTS

-dont_show *references*
Specifies a list of references that will not be displayed.

-quiet
Specifies that no informational messages will be displayed.

-limited *limited_keys*
Specifies a list of licenses that provide limited access to the design.

regular_keys
Specifies a list of licenses that provide regular access to the design.

DESCRIPTION

This command adds license information to a design. For more information on licensing designs, please refer to the section on licensing in the DesignWare manual.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

In this example, the full license "ADDERS" and the limited license "EVAL" is added to the current design. In addition, **-dont_show** indicates that the design will not become visible until all the existing references are changed.

```
prompt> set_design_license ADDERS\
```

```
set_design_license  
2322
```



```
-limited EVAL -dont_show ""
```

SEE ALSO

```
list_licenses(2)
```

set_design_top

Specifies the top-level design instance.

SYNTAX

```
status set_design_top  
instance_name
```

Data Types

```
instance_name      string
```

ARGUMENTS

```
instance_name  
    Specifies the top-level instance in the design.
```

DESCRIPTION

The **set_design_top** command specifies the top-level design instance. This information is used only by simulation and verification tools.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLE

The following example shows how to use the **set_design_top** command:

```
prompt> set_design_top ALU07
```

set_die_area

Sets the dimensions of the die area.

SYNTAX

```
int set_die_area  
-coordinate {X1 Y1 X2 Y2}
```

ARGUMENTS

```
-coordinate {X1 Y1 X2 Y2}  
    Specifies the lower left and upper right coordinates of the rectangular die  
    area. The coordinates are in microns relative to the chip origin.
```

DESCRIPTION

The **set_die_area** command defines the die area of the chip. It overwrites any previous die area setting. The die area should be larger than the core area of the chip.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows how to set a rectangular die area with the lower-left corner at coordinates (100,100) and upper-right corner at the coordinates (2000,2000):

```
prompt> set_die_area -coordinate {100 100 2000 2000}
```

SEE ALSO

```
get_die_area(2)
```

set_direct_power_rail_tie

Sets the **direct_power_rail_tie** attribute on library pins.

SYNTAX

```
int set_direct_power_rail_tie  
lib_pin_list [true | false]
```

ARGUMENTS

lib_pin_list

A list of library pins on which the **direct_power_rail_tie** attribute is to be set. If more than one library pin name is specified, they must be enclosed either in quotes or in braces ({}). For more information about object names, refer to the **find** command manual page.

true | false

Specifies the value with which to set the **direct_power_rail_tie** attribute. The default is *true*.

DESCRIPTION

Sets the **direct_power_rail_tie** attribute on library pins. If a match is found, a **direct_power_rail_tie** attribute is assigned to the library_pin(s).

If the value of **direct_power_rail_tie** attribute on a library pin is true, then all the pins in the design, for which this is the library pin, it won't connect it to a tieoff cell for connecting it to constant signal. Instead, it will keep them connecting to generic constant signals, so that during power routing these pins can be connected directly to power rails.

To remove **direct_power_rail_tie** attribute, use **remove_attribute**.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command specifies that the all the **GND** pins of cells with the library cells **leaf_cell** won't be connected to tieoff cells to connect it to a constant signal.

```
prompt> set_direct_power_rail_tie target_lib/leaf_cell/GND
```

SEE ALSO

```
report_direct_power_rail_tie(2)  
remove_attribute(2)
```

set_disable_clock_gating_check

Disables the clock gating check for specified objects in the current design.

SYNTAX

```
string set_disable_clock_gating_check  
object_list
```

Data Types

object_list list

ARGUMENTS

object_list
 Specifies a list of cells and pins for which the clock gating check is to be disabled.

DESCRIPTION

This command disables the clock gating check for specified cells and pins in the current design.

Any cell or pin in the current design or its subdesigns can be disabled. Cells at lower levels in the design hierarchy are specified as *instance1/instance2/cell_name*. Specify pins at lower levels in the design hierarchy as *instance1/instance2/cell_name/pin_name*.

For subdesigns in the hierarchy, specify instance names instead of design names.

When the clock gating checking through a cell is disabled, all gating checks in the cell are disabled. When the checking through a pin is disabled, any gating check is disabled if it uses the disabled pin as a gating clock pin or a gating enable pin.

The compile command adjusts only the size of the cell by replacing it with other cells having the same logic function but with different drive capacity. No other logic transformations involving the clock gating cell are permitted.

To restore gating checks disabled by the **set_disable_clock_gating_check** command, use the **remove_disable_clock_gating_check** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example disable gating checks on the specified cell or pin:

```
prompt> set_disable_clock_gating_check an2/Z
```

```
prompt> set_disable_clock_gating_check or1
```

SEE ALSO

```
remove_disable_clock_gating_check(2)
```

set_disable_timing

Disables timing arcs in the current design.

SYNTAX

```
int set_disable_timing  
object_list  
[-from from_pin_name -to to_pin_name]  
[-restore]
```

Data Types

<i>object_list</i>	list
<i>from_pin_name</i>	string
<i>to_pin_name</i>	string

ARGUMENTS

object_list
Specifies a list of cells, pins, ports, library pins, or library cells through which timing is to be disabled. The cells or the cells of pins are set to be size only to leave room for compile to apply sizing on them.

-from *from_pin_name*
Specifies that arcs only from this pin on the specified cell are to be disabled. The *object_list* must contain only cells if **-from** and **-to** are specified. When used, the -from and -to options must be specified together.

-to *to_pin_name*
Specifies that arcs only to this pin on the specified cell are to be disabled. The *object_list* must contain only cells if **-from** and **-to** are specified. When used, the -from and -to options must be specified together.

-restore
Indicates that the specified arcs are to be restored and not disabled.

DESCRIPTION

The **set_disable_timing** command disables timing through the specified cells, pins, or ports in the current design.

Any cell, pin, or port in the current design or its subdesigns can be disabled. Cells at lower levels in the design hierarchy are specified as *instance1/instance2/cell_name*. Pins at lower levels in the design hierarchy are specified as *instance1/instance2/cell_name/pin_name*. Ports at lower levels in the design hierarchy are specified as *instance1/instance2/cell_name/port_name*.

Library pins and library cells can also be disabled.

Note: For subdesigns in the hierarchy, you must specify instance names instead of design names.

When the timing through a cell is disabled, only those cell arcs that lead to the output pins of the cell are disabled. When the timing through a pin or port is disabled, only those cell arcs that lead to or from that pin or port are disabled.

When the **-from** and **-to** pins are specified, all arcs between these pins on the cell are disabled.

Use **report_lib -timing_arcs** to list the timing arcs for a library cell, and **report_design** to list the disabled arcs in the current design.

To enable the disabled cells, pins, or ports, use **set_disable_timing -restore** or **reset_design**.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

In the following example, all timing arcs are disabled that lead to output pins of cells "U2" and "U3" of the current design.

```
prompt> set_disable_timing {U2 U3}
```

In the following example, all timing arcs that lead to output pin "Z" of cell "U5" in the current design are disabled.

```
prompt> set_disable_timing U5/Z
```

In the following example, all timing arcs between pins 'A' and 'Z' on cell U1/U1 in the current design are disabled.

```
prompt> set_disable_timing U1/U1 -from A -to Z
```

In the following example, all timing arcs between pins 'D' and 'Q' on cell FD1 in the library 'my_lib' are disabled. This command disables arcs on a library cell; therefore, all instances of that library cell are affected in any design linked to the same library.

```
prompt> set_disable_timing my_lib/FD1 -from D -to Q
```

SEE ALSO

```
current_design(2)  
remove_attribute(2)  
report_lib(2)  
reset_design(2)
```


set_distributed_route

Initializes the network for distributing routing jobs.

SYNTAX

```
status set_distributed_route
[-lsf]
[-lsf_advanced bsub_options]
[-jp_disconnect]
[-jp_machines {mname1 mname2 mname3...}]
[-jp_limit {mname0 limit mname1 limit mname2 limit...}]
[-jp_bin {mname1 bin_path mname2 bin_path...}]
[-jp_lib {mname1 lib_path mname2 lib_path...}]
```

Data Types

bsub_options string

ARGUMENTS

-lsf

Selected the option to setup LSF pool for job distributions. Jobs will be run based on default LSF setup. Used JP setup if the option is not selected.

-lsf_advanced *bsub_options*

Users can specify advance options here to management their submitted jobs on LSF. This option is available only if -lsf option is selected.

-jp_disconnect

Disconnect the network. Default is to connect network to local host and specified machines when the network has not yet been connected, and to get the status of each machine when the network has already been set up. This option is not available for LSF run.

-jp_machines {*mname1 mname2 mname3...*}

Lists machine names where users want to connect to. Local host will always be used, which is on default, so there is no needed to specify host machine here. User can specify total up to 16 machines (including host machine). The list will be kept from previoud run, and will be overwritten partially by next run. This option is not available for LSF run.

-jp_limit {*mname0 limit mname1 limit mname2 limit...*}

Users can limit the number of CPUs used on each machine. Speified 0 if users do not want this machine to be used. Specified -1 if users want all of the available CPUs on this machine to be used. All available CPUs will be used if this option is off. The option is not available for LSF run.

The machine name must be appeared in -jp_machines option if users try to specify CPU limit by using this option. The limit setting will be ignored if the macine users specified here can not be matched with the name in -jp_machines list. However, if the machine had been connected due to previous run, even though users don't specified it in -jp_machines, all CPUs will be used on that machine.

mname and limit must be in pairs. Meaning that machine name is required if users want to limit the CPUs, and each machine name must be followed by the number of CPU limit.

All available CPUs will be used for certain machines that users don't specify the limit.

`-jp_bin {mname1 bin_path mname2 bin_path...}`

bin_path is where the executable files are located. From different machines, it is likely that the paths are completely different. The default is to set current binary on local host machine, so users don't need to set the bin_path for local host machine. The current binary will be applied for all if this option is off. The option is not available for LSF run.

The machine name must be appeared in -jp_machines option if users try to specify any binary path by using this option. The bin_path setting will be ignored if the machine users specified here can not be matched with the name in -jp_machines list. However, if the machine had been connected due to previous run, even though users don't specified it in -jp_machines, the existing binary will be used on that machine.

mname and bin_path must be in pairs. Meaning that machine name is required if users want to setup bin_path, and each machine name must be followed by a binary path.

The path of preceding item will be considered for certain machines that users don't specify the bin_path.

`-jp_lib {mname1 lib_path mname2 lib_path...}`

The library path is where the working library is located. From different machines, you have to provide this path so that the library can be reached. The default is to set current library on local host machine, so users don't need to set the lib_path for local host machine. The current library will be applied for all if this option is off. The option is not available for LSF run.

The machine name must be appeared in -jp_machines option if users try to specify any library path by using this option. The lib_path setting will be ignored if the machine users specified here can not be matched with the name in -jp_machines list. However, if the machine had been connected due to previous run, even though users don't specified it in -jp_machines, the existing library will be used on that machine.

mname and lib_path must be in pairs. Meaning that machine name is required if users want to setup lib_path, and each machine name must be followed by a library path.

The path of preceding item will be considered for certain machines that users don't specify the lib_path.

DESCRIPTION

Set up the network and connect local machine as a host to distribute jobs on available CPUs.

No other operation can be executed except for disconnect if network was connected by previous run, or if JP server is up, only if users shut it down before they can modify any option setting.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example is to setup local host machine for distributing jobs on all available CPUs. Current binary and library are used.

```
prompt> set_distributed_route
```

The following example is to setup host machine, genop0023, for running jobs on LSF. The memory limit is set to 3000, and jobs will be run only on specified queue, normal_test.

```
prompt> set_distributed_route -lsf -lsf_advanced {-m "genop0023" \  
-M 3000 -q "normal_test"}
```

The following example is to distribute jobs on all available CPUs of host machine and only one CPU of machine_0. Current binary and library are used for both machines.

```
prompt> set_distributed_route -jp_machines {machine_0} -jp_limit \  
{machine_0 1}
```

The following example is to distribute jobs on two CPUs of host machine (mach_host), one CPU on machine_1, and 3 CPUs on machine_2. Current binary and library are used for host machine. /user1/username/src/bin are used for machine_1 and machine_2. /user1/username/data/library/design are used for machine_1 and machine_2.

```
prompt> set_distributed_route -jp_machines {machine_1 machine_2} \  
-jp_limit {mach_host 2 machine_1 1 machine_2 3} -jp_bin {machine_1 \  
/user1/username/src/bin} -jp_lib {machine_1 /user1/username/data/library/  
design}
```

SEE ALSO

```
report_distributed_route(2)  
remove_distributed_route(2)  
close_distributed_route(2)
```

set_domain_supply_net

Set the primary power net and primary ground net of an already existing power_domain. This command is supported only in UPF mode.

SYNTAX

```
int set_domain_supply_net  
domain_name  
-primary_power_net supply_net_name  
-primary_ground_net supply_net_name
```

Data Types

domain_name	string
supply_net_name	string

ARGUMENTS

domain_name
Specify the name of the power domain.
If a power domain with the specified name does not exist, in the current scope, this command fails.
The domain_name option is a required option and must be specified.

-primary_power_net supply_net_name
Specify the power net of the power domain.
If a supply net with the specified name does not exist in the current scope, the command fails. If the supply net is not associated with specified power domain, the command fails.
This is a required option and must be specified.

-primary_ground_net supply_net_name
Specify the ground net of the power domain.
If a ground net with the specified name does not exist in the current scope, the command fails. If the ground net is not associated with specified power domain, the command fails.
This is a required option and must be specified.

DESCRIPTION

The *set_domain_supply_net* command enables you to set the primary power net and the primary ground net of a power domain. All extent of the power domain shares the same power/ground supply until explicitly excluded. Here "explicitly excluded" means it is explicitly connected with another supply_net (non primary power/ground supply_net) by command *connect_supply_net*. This command errors out if power domain already has a primary power and ground net.

The supply_net must be created in the same power_domain at first before it could be set as the primary power or ground supply_net. Use command *create_supply_net* to create a supply_net at a specified power_domain.

Command returns 1 on success, returns 0 otherwise.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates two supply_net on power_domain PD1, then set them as primary power or ground net.

```
prompt> create_power_domain PD1 -elements INST_1 PD1 prompt> create_supply_net A_VDD  
-domain PD1 A_VDD prompt> create_supply_net A_VSS -domain PD1 A_VSS prompt>  
set_domain_supply_net PD1 -primary_power_net A_VDD -primary_ground_net A_VSS 1
```

SEE ALSO

```
create_power_domain(2)  
create_supply_net(2)  
connect_supply_net(2)
```

set_dont_touch

Sets the **dont_touch** attribute on cells, nets, references, and designs in the current design, and on library cells, to prevent modification or replacement of these objects during optimization.

SYNTAX

```
status set_dont_touch  
object_list  
[true | false]
```

Data Types

object_list list

ARGUMENTS

object_list
Specifies list of objects (cells, nets, references, designs, and library cells) on which the **dont_touch** attribute is to be set, so that the objects will not be modified or replaced. If more than one object name is specified, each name must be enclosed in quotation marks (" ") or braces ({}).

true | false
Specifies the value with which to set the **dont_touch** attribute. The default is **true**.

DESCRIPTION

This command sets the **dont_touch** attribute on cells, nets, references, designs in the current design, and on library cells, to prevent modification or replacement of these objects during optimization. Setting the **dont_touch** attribute on a hierarchical cell implies "dont_touch" on all cells below it that do not have **dont_touch** set to **false**. Setting the **dont_touch** attribute on a library cell implies "dont_touch" on all instances of that cell. Setting the **dont_touch** attribute on a net implies "dont_touch" only on mapped combinational cells connected to that net. All mapped combinational cells connected to the net must be at the same level of hierarchy. Setting the **dont_touch** attribute on a reference implies "dont_touch" on all cells of that reference during subsequent optimizations of the design.

Setting the **dont_touch** attribute on a design has an effect only when the design is instantiated within another design as a level of hierarchy. In this case, "dont_touch" on the design implies that all cells under that level of hierarchy are "dont_touch." Setting **dont_touch** on the top-level design has no effect because it is not instantiated within any other design.

Note the following information:

- The **dont_touch** attribute applied by the tool on the cells and nets in the transitive fanout of **dont_touch_network** objects overrides the **false dont_touch**

attribute value that is set by the **set_dont_touch** command. You can see the **dont_touch** value on cells or nets by using the **report_cell** or **report_net** command.

When an object name is specified, the **set_dont_touch** command first looks within the current design for a cell that matches that name. If it finds a match, a **dont_touch** attribute is assigned to that cell. If no match is found, the command next looks within the current design for a net that matches the specified name. If a match is found, a **dont_touch** attribute is assigned to that net. If no match is found, the command next looks for a reference, then a design in the system, and finally for a library cell in the system. The tool assigns a **dont_touch** attribute to the first object for which it finds a match.

To remove the **dont_touch** attribute, use the **remove_attribute** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example specifies that the cells named *block1* and *analog1* are not to be modified during optimization:

```
prompt> set_dont_touch [get_cells {block1 analog1}]
```

The following example specifies that the net named *N1* is not to be modified during optimization:

```
prompt> set_dont_touch [get_nets N1] true
```

The following example specifies that the net named *N1* can be modified during optimization. Setting **set_dont_touch** to **false** removes the previous **dont_touch**.

```
prompt> set_dont_touch [get_nets N1] false
```

SEE ALSO

```
report_cell(2)  
report_net(2)  
report_reference(2)  
set_dont_touch_network(2)
```

set_dont_touch_network

Sets the **dont_touch_network** attribute on clocks, pins, or ports in the current design to prevent cells and nets in the transitive fanout of the **set_dont_touch_network** objects from being modified or replaced during optimization.

SYNTAX

```
status set_dont_touch_network  
object_list  
[-no_propagate]
```

Data Types

object_list list

ARGUMENTS

object_list
Specifies a list of clocks, pins, or ports in the current design on which to set the **dont_touch_network** attribute. If you include more than one object, they must be enclosed in either quotation marks or braces ({}).

-no_propagate
Indicates that the dont_touch network is not propagated through logic gates.

DESCRIPTION

This command sets the **dont_touch_network** attribute on clocks, pins, or ports in the current design. When a design is optimized, the **compile** command assigns **dont_touch** attributes to all cells and nets in the transitive fanout of **dont_touch_network** objects so that they are not modified or replaced during optimization. A **dont_touch** attribute assigned by the tool using the **set_dont_touch_network** command overrides the **FALSE** **dont_touch** attribute value from the **set_dont_touch** command. To see the **dont_touch** value on the cells or nets, use the **report_cell** or **report_net** command. The **dont_touch** assignment stops at registers. An element is recognized as a register only if it has setup or hold constraints. If you specify a clock, the transitive fanout of all the clock's sources is affected. The **report_net** command displays the nets that are implicitly **dont_touch** as a result of using the **set_dont_touch_network** command.

The **set_dont_touch_network** command is intended primarily for clock circuitry. Placing a **dont_touch_network** on a clock object prevents the **compile** command from modifying the clock buffer network.

Note that the **dont_touch_network** attribute does not prevent compile from mapping unmapped logic (for example, cells read in from an HDL description).

To remove the **dont_touch_network** attribute, use the **remove_attribute** command on specific object on which **dont_touch_network** has been set.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command adds a **dont_touch_network** attribute to a port named *clock_in*:

```
prompt> set_dont_touch_network clock_in
```

The following command removes the **dont_touch_network** attribute on port named *clock_in*:

```
prompt> remove_attribute [get_port clock_in] dont_touch_network
```

SEE ALSO

```
remove_attribute(2)  
report_cell(2)  
report_clock(2)  
report_design(2)  
report_net(2)  
report_port(2)  
set_dont_touch(2)
```

set_dont_touch_placement

Fixes the location of a cell or cluster.

SYNTAX

```
status set_dont_touch_placement  
object_list
```

ARGUMENTS

object_list
List of objects, each of which is a leaf cell or cluster.

DESCRIPTION

This command fixes the location of a leaf cell or cluster by applying fixed placement restriction. The placement of these cells will not be altered.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows how to prevent the placer from moving the instance INST_1.

```
prompt> set_dont_touch_placement INST_1
```

SEE ALSO

```
set_cell_location(2)  
create_placement(2)  
remove_dont_touch_placement(2)
```

set_dont_use

Sets the **dont_use** attribute on library cells to exclude them from the target library during optimization.

SYNTAX

```
int set_dont_use  
[-power] object_list
```

Data Types

object_list list

ARGUMENTS

-power

Specifies that the list of objects passed should not be considered for Power Compiler clock gate mapping in addition to excluding them from target library. With out this switch the objects are excluded from target library, but are still available to Power Compiler for clock gate mapping.

object_list

Specifies a list of objects (*lib_cells*, *modules*, or *implementations*) on which the **dont_use** attribute is to be set. Object names must contain a library prefix; for more information, refer to the EXAMPLES section of this manual page, and to the manual page for the **find** command. If more than one objects is included, they must be enclosed in quotes or braces ({}).

DESCRIPTION

Sets the **dont_use** attribute on specified library objects, so that they are not used. The specified objects must be in one of the target libraries or in a library already loaded into Design Analyzer. If this attribute exists on a library cell, module or implementation **compile** does not use it for a design.

When -power switch is used, an additional attribute *pwr_cg_don_use* is set in addition to *dont_use*. When this attribute is present on an object, Power Compiler will not consider it for mapping.

NOTE: This command affects only the copy of the library that is currently loaded into memory, and has no effect on the version that exists on disk. However, if the library is written out using **write_lib**, any **dont_use** attribute will be written out, causing library objects to be permanently disabled.

To remove **dont_use** attributes, use the **remove_attribute** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command disables the lib_cells called 'G1' and 'G2' from the 'tech_lib' library:

```
prompt> set_dont_use {tech_lib/G1 tech_lib/G2}
```

The following command disables the implementations called 'imp1' and 'imp2' from the module 'syn_lib/mod':

```
prompt> set_dont_use {syn_lib/mod/imp1 syn_lib/mod/imp2}
```

The following command removes the lib_cells called 'icg_1' icg_2' from the list of integrated clock gating lib_cells available for clock gate mapping.

```
prompt> set_dont_use -power {tech_lib/icg_1 tech_lib/icg_2}
```

SEE ALSO

remove_attribute(2)
report_lib(2)
target_library(3)

set_dp_int_round

Set the rounding positions on datapath output nets.

SYNTAX

```
status set_dp_int_round  
nets  
external_rounding_position  
[internal_rounding_position]
```

Data Types

<i>nets</i>	list
<i>external_rounding_position</i>	int
<i>internal_rounding_position</i>	int

ARGUMENTS

nets
List of nets that the rounding will be applied to.

external_rounding_position
The bit position for external rounding.

internal_rounding_position
The bit position for internal rounding.

DESCRIPTION

This command specifies the external and internal rounding positions on the supported nets that are driven by a DW multiplier-like component or a DW complex datapath block.

The supported nets are: Nets that are driven by cells that infer DesignWare operator MULT_UNSP and MULT_TC_OP; Nets that are driven by cells that instantiate the following DesignWare components: DW02_mult, DW_mult_uns, DW_mult_tc, DW_square, DW02_prod_sum, DW02_prod_sum1, DW_prod_sum_uns, DW_prod_sum_tc, DW02_mac, DW_mac_tc, DW_mac_uns.

Internal rounding allows to make a tradeoff between precision and area in arithmetic circuits. All bits lower than the internal rounding position are discarded from the internal addends. Area is saved because no logic is required to generate and add these bits. However, a rounding error is introduced, which degrades the precision of the result. An offset is automatically added to keep the error range as symmetric and the bias as small as possible.

The result of a datapath with internal rounding is usually truncated somewhere above the internal rounding position. This truncation can also be rounded by setting the external rounding position to this bit position. Rounding is achieved by adding a constant 1 at the next lower bit position.

Since internal rounding changes the functionality of a datapath block, a behavioral model that reflects the changed functionality is written out as Verilog file. This behavioral Verilog file can be used for simulation and verification of the new netlist. The file is created in the dwsvf directory.

A rounding error report is printed with the **report_resources** command.

To remove the rounding attribute, use the **remove_dp_int_round** command.

A few notes about using this command: 1. Currently, if you set_implementation on the cells that infer DesignWare operators, internal rounding will not happen. 2. We only support internal rounding on generated implementations. Therefore, if you set_implementation to the non-generated implementation for a instantiated DesignWare cell, the internal rounding will not happen.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets the external rounding position on net z1* to be 7, and the internal rounding position to be 5:

```
prompt> set_dp_int_round [get_nets z1*] 7 5
```

The report_resources command reports the rounding errors:

```
prompt> report_resources
```

DW rounding error report in design mult_inst_DW_mult_uns_1(cell mult_6):
Rounding Error Report for product[15:0]

Rounding	Relative to Bit	Relative Error			Absolute Error		
		Min	Max	Avg	Min	Max	Avg
Internal	5	-3.00	2.00	-0.50	-96	64	-16
Internal	7	-0.75	0.50	-0.12	-96	64	-16
External	7	-0.50	0.49	-0.00	-64	63	-0.5
Total	7	-1.25	0.99	-0.13	-160	127	-16.5

The behavioral Verilog file has extension .bvrl and is found in the dwsvf_* directory:

```
prompt> ls dwsvf_*
dwarchs-0.v.e  mult_inst_DW_mult_uns_1.round.bvrl
```

SEE ALSO

`remove_dp_int_round(2)`

set_drive

Sets the **rise_drive** or **fall_drive** attributes to specified resistance values on specified input and inout ports.

SYNTAX

```
int set_drive  
resistance  
[-rise] [-fall] [-min] [-max]  
port_list
```

Data Types

<i>resistance</i>	float
<i>port_list</i>	list

ARGUMENTS

resistance
Specifies a nonnegative resistance value to which the **rise_drive** or **fall_drive** attributes are to be set on the ports in *port_list*. The *resistance* is the output resistance of the cell that drives the port, such that a higher drive strength (resistance) means less drive capability and longer delays. Thus, a *resistance* of 0 is infinite drive, or no delay between the ports and all that is connected to them. The *resistance* must be in unit consistent with the technology library used during optimization.

-rise -fall
Indicates that the **rise_drive** or **fall_drive** attributes of *port_list* are to be set to *resistance*. If you don't specify either, both are set to *resistance*.

-min -max
Indicates that the drive strength is to be applied for minimum or maximum delay analysis. If no value is specified for minimum analysis, the maximum value is used.

port_list
Specifies a list of input or inout port names of the current design on which the drive attributes are to be set. If you specify more than one port, you must enclose the ports in either quotes or braces ({}).

DESCRIPTION

Sets the **rise_drive** or **fall_drive** attributes to *resistance* on specified input and inout ports in the current design.

Note: The **set_driving_cell** command is more convenient and accurate than **set_drive** for describing the drive capability of a port. The **set_drive** command removes any corresponding rise or fall driving cell attributes on the specified ports. Use **set_driving_cell** instead of **set_drive**, if possible.

During optimization, the drive of an input port is used to calculate the timing delay to gates driven by that port. The delay to these gates is computed as follows for the generic CMOS delay model:

$$\text{Time} = \text{arrival\_time} + [\text{drive} * \text{load}(\text{net})] + \text{connect\_delay} \text{ (if any)}$$

To view drive information on ports, use **report_port -drive**. To remove drive attributes from ports, use **remove_attribute**. The **reset_design** command removes all attributes from a design, including the drive attributes.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example sets the rise and fall drives of ports "A", "B", and "C" to 2.0.

```
prompt> set_drive 2.0 {A B C}
```

The following example sets the rise and fall drives of all input ports to 2 and sets the rise drive on port "B" to 1.

```
prompt> set_drive 2 all_inputs( )
prompt> set_drive -rise 1 B
```

The following example sets both rise and fall drives of port "C" to corresponding drives of pin "OUT" of "INVERTER" from "TECH_LIBRARY" using the **drive_of** command.

```
prompt> set_drive -rise drive_of(-rise TECH_LIBRARY/INVERTER/OUT) {C}

prompt> set_drive -fall drive_of(-fall TECH_LIBRARY/INVERTER/OUT) {C}
```

SEE ALSO

all_inputs(2)
current_design(2)
drive_of(2)
remove_attribute(2)
report_port(2)
reset_design(2)
set_driving_cell(2)
set_load(2)

set_driving_cell

Sets attributes on input or inout ports of the current design, specifying that a library cell or output pin of a library cell drives specified ports.

SYNTAX

```
int set_driving_cell
[-lib_cell lib_cell_name]
[-library lib]
[-rise]
[-fall]
[-min]
[-max]
[-pin pin_name]
[-from_pin from_pin_name]
[-dont_scale]
[-no_design_rule]
[-none]
[-input_transition_rise rtran]
[-input_transition_fall ftran]
[-multiply_by factor]
port_list
[-cell obsolete_-please_use_-lib_cell_instead]
```

Data Types

<i>lib_cell_name</i>	string
<i>lib</i>	string
<i>pin_name</i>	string
<i>from_pin_name</i>	string
<i>rtran</i>	float
<i>ftran</i>	float
<i>factor</i>	float
<i>port_list</i>	list

ARGUMENTS

-lib_cell *lib_cell_name*
Specifies the name of the library cell to use to drive ports. You can use this option with the **-pin** option when the cell has more than one output pin to set the **driving_cell_rise** and the **driving_cell_fall** string attributes to *lib_cell_name* on the ports. To specify different cells for the rising and falling cases, execute the command twice, once using the **-rise** option and once using the **-fall** option, specifying the appropriate *lib_cell_name* for each. When you use this option, you must also use the **-dont_scale** and **-multiply_by** options. By default, the command searches the libraries in **link_library** for the cell.

-library *lib*
Specifies the library name or a collection of libraries in which to find the name of the library cell to use to drive ports (*lib_cell_name*). If the lib cell can be found in multiple libraries, the library with matching operating

condition on the port will be used. When no library is specified, all the libraries will be searched for the lib cell with matching operating condition. If a library cell with a matching operating condition cannot be found, the first one found with a matching lib cell name will be used. You can use this option only with the **-lib_cell** option. It sets the **driving_cell_library_rise** and the **driving_cell_library_fall** string attributes to the library name on the ports. To specify different libraries for the rising and falling cases, execute the command twice, using the **-rise** option once and the **-fall** option once, specifying the appropriate *lib* for each.

-rise

Specifies that the *lib_cell_name*, *lib*, *pin_name*, and *from_pin_name* correspond to the rising case, and sets the **driving_cell_rise**, **driving_cell_library_rise**, **driving_cell_pin_rise**, and **driving_cell_from_pin_rise** attribute strings, respectively, on the objects. You can use this option with the **-fall** option to specify both the rising and falling cases.

-fall

Specifies that the *lib_cell_name*, *lib*, *pin_name*, and *from_pin_name* correspond to the falling case, and sets the **driving_cell_fall**, **driving_cell_library_fall**, **driving_cell_pin_fall**, and **driving_cell_from_pin_fall** attribute strings, respectively, on the objects. You can use this option with **-rise** to specify both the rising and falling cases.

-min

Sets driving cell information for analysis at the minimum operating condition only.

-max

Sets driving cell information for analysis at the maximum operating condition only.

-pin *pin_name*

Specifies the output pin on the driving cell that is to drive the ports. The **-pin** option is required when you use the **-lib_cell** option and the driving cell has more than one output pin or when using the **-from_pin** option. This option sets the **driving_cell_pin_rise** and the **driving_cell_pin_fall** string attributes to *pin_name* on the ports. To specify different pins for the rising and falling cases, execute the command twice, once using the **-rise** option and once using the **-fall** option, specifying the appropriate *pin_name* for each. The default is to use the first timing arc found on the library cell.

-from_pin *from_pin_name*

Specifies the input pin on the driving cell to use when finding a timing arc. This option is required when the driving cell has more than one input pin and the arcs from those pins have different drive characteristics, and when using the **-pin** and **-lib_cell** options. The **-from_pin** option sets the **driving_cell_from_pin_rise** and **driving_cell_from_pin_fall** string attributes to *from_pin_name* on the ports. The default is to use the first timing arc found on the library cell.

-dont_scale

Specifies that the timing analyzer is not to scale the drive capability of the ports according to the current operating conditions. You can use this option only with the **-lib_cell** option. The **-dont_scale** option sets the **driving_cell_dont_scale** Boolean attribute to **true** on the ports. By default, the port drive capability is scaled for operating conditions exactly as the driving cell would be scaled.

-no_design_rule

Indicates that the design rules associated with the driving cell are not to be applied to the driven port. This option sets the **driving_cell_no_drc** Boolean attribute to **true** on the ports. Timing-related attributes are still applied. The tool issues a warning if this option is not used. By default, design rule attributes (**max_fanout**, **max_capacitance**, **max_transition**, **min_fanout**, **min_capacitance**, **min_transition**) are derived from the driving cell and its library and applied to the port.

-none

Removes previous driving cell information.

-input_transition_rise *rtran*

Specifies the input rise transition time associated with the **-from_pin** option. Use the **-input_transition_rise** and **-input_transition_fall** options to obtain a more accurate transition time and delay time at the output pin by capturing the accurate transition time associated with the **-from_pin**. The default value is **0**.

-input_transition_fall *ftran*

Specifies the input fall transition time associated with the **-from_pin** option. The default value is **0**.

-multiply_by *factor*

Specifies a factor by which to multiply the delay characteristics of the ports. You can use this option only with the **-lib_cell** option. It affects both the load delay and the transition times of the port. It sets the **driving_cell_multiply_by** float attribute to the specified factor on the ports. The default is **1.0**.

port_list

Specifies a list of names of input or inout ports in the current design on which the driving cell attributes are to be placed. If more than one port is specified, they must be enclosed in either quotes ({}).

DESCRIPTION

This command sets attributes on the specified input or inout ports in the current design to associate an external driving cell with the ports. The drive capability of the port is the same as if the specified driving cell were connected in the same context to allow accurate modeling of the port drive capability for nonlinear delay models. For the CMOS2 delay model, the edge rate of pins driven by the port is the same as if the driving cell were substituted for the port. Unless you specify the **-dont_scale** option, the drive capability of the port is scaled according to the current operating conditions.

To view drive information on ports, use the **report_port** command with the **-drive** option.

You can use the **characterize** command to automatically set driving cell attributes on subdesign ports based on their context in the entire design.

You can use the **remove_driving_cell** or **reset_design** command to remove driving cell attributes on ports. The **remove_driving_cell** command removes any corresponding rise or fall drive resistance attributes (from the **set_drive** command) on the specified ports. It is considered best practice to use the **set_driving_cell** command instead of **set_drive**, because **set_driving_cell** is more accurate.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example associates the drive capability of the *AND2* library cell with the *IN1* port.

```
prompt> set_driving_cell -lib_cell AND2 {IN1}
```

The following example associates the drive capability of the *INV/Z* library pin from the *tech_lib* library with all input and inout ports.

```
prompt> set_driving_cell -lib_cell INV -pin Z \  
-library tech_lib [all_inputs]
```

The following example associates the drive capability of the *INV* library cell with the *IN1* port and specifies that the delay values should not be scaled by operating conditions factors.

```
prompt> set_driving_cell -lib_cell INV -dont_scale {IN1}
```

The following example associates the *Z* pin of *BUF1_TS* with the *IN1* port for rising delays and the *Q* pin of *DFF_TS* for falling delays.

```
prompt> set_driving_cell -rise -lib_cell BUF1_TS -pin Z  
prompt> set_driving_cell -fall -lib_cell DFF_TS -pin Q {IN1}
```

The following example sets the *AN2* driving cell to the *top1* input port with an input rise transition time of 2.3 on *A* specified by the **-from_pin** option.

```
prompt> set_driving_cell -lib_cell AN2 \  
-input_transition_rise 2.3 -from_pin A top1
```

SEE ALSO

`all_inputs(2)`
`characterize(2)`
`current_design(2)`
`remove_driving_cell(2)`
`report_port(2)`
`reset_design(2)`
`set_drive(2)`
`set_load(2)`
`set_max_capacitance(2)`
`set_max_fanout(2)`
`set_max_transition(2)`
`set_min_capacitance(2)`
`set_port_fanout_number(2)`

set_droute_options

Sets detailed router cell-persistent options.

SYNTAX

```
status set_droute_options  
[-name name]  
[-value value]  
[-default]
```

Data Types

```
name      string  
value     integer|real|string
```

ARGUMENTS

-name *name*
Specifies the name of the option to be set. Use command **report_droute_options** to check for valid names.

-value *value*
Specifies the value of the option to be set. Use command **report_droute_options -name** to check for valid range, type, default value, current value and a brief description of a named option.

-default
Set all droute options to their default values. To set a specific droute option to its default value, use "-name" and "-value" arguments. Use command **report_droute_options** to check for default value.

DESCRIPTION

The **set_droute_options** command sets options to tune detailed router operation. These settings are persistent in the Milkyway design. This command saves cell-persistent options in cell data base, and marks cell as being modified. To permanently save these options, you have to save cell upon exit. This command is optional, if you want defaults to apply. All options are optional too. However, if you want to set a specific option value, "-name" and "-value" have to be specified. To reset all options to their default values, use "-default" argument. To print all available detailed router options, their current value, default value, range, type and a brief description, use command **report_droute_options**

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

```
prompt> set_droute_options -name shiftVarWidthWire -value 2
```

```
prompt> set_droute_options -name via1MacroDensity -value 20.0  
prompt> set_droute_options -default
```

SEE ALSO

`report_droute_options(2)`

set_equal

Defines two input ports as logically equivalent.

SYNTAX

```
int set_equal  
port1  
port2
```

Data Types

```
port1      string  
port2      string
```

ARGUMENTS

```
port1 port1  
Two logically-equivalent input port in the current design.
```

DESCRIPTION

Defines two input ports in the current design as logically equivalent. This information is used to eliminate redundant ports, improving optimization quality.

Logical inconsistencies are checked for in relationships between ports. Specifying an inconsistent logical relationship between ports is not allowed.

Use the **reset_design** command to remove this property from a port.

EXAMPLES

The following example sets two input ports "A" and "B" logically equal:

```
prompt> set_equal A B
```

The following example sets up a contradictory relationship between two ports and creates an error:

```
prompt> set_equal A B  
prompt> set_opposite A B  
Warning: Can't set equal ports opposite in design 'example.ddc': 'A' 'B'
```

SEE ALSO

```
reset_design(2)  
set_opposite(2)  
current_design(3)
```

set_error_view_property

Adds or replaces values for error view properties.

SYNTAX

```
status set_error_view_property
[-error_view mw_error_view]
[-writer product_name]
[-version version_string]
[-ignore_type_name_property true | false]
[-run_set run_set_name]
[-areas areas]
[-excluded_areas areas]
[-command command_string]
```

Data Types

<i>mw_error_view</i>	list or collection of one item
<i>product_name</i>	string
<i>version_string</i>	string
<i>run_set_name</i>	string
<i>areas</i>	list of rectangles
<i>command_string</i>	string

ARGUMENTS

-error_view *mw_error_view*
Specifies the error view in which to add or replace property values.
If you do not specify this option, the properties are written to the top-level design. You can specify only one error view; specifying more than one error view causes an error.

-writer *product_name*
Sets the Writer property. This is the product name of the error view writer.

-version *version_string*
Sets the Version property. This is the product version of the error view writer.

-ignore_type_name_property true | false
Sets the IgnoreTypeNameProperty property. If this property is true, the TypeName property of error type is ignored. The TypeName property is used to store type names that are longer than can be normally stored in the database. If the value is false, the IgnoreTypeNameProperty property can be deleted.

-run_set *run_set_name*
Sets the Runset property. This is the name of the runset file input to the DRC check.

-areas *areas*
Sets the Areas property. These are the areas of the design that were included in the DRC check. To specify the areas, specify one or more rectangles using

set_error_view_property

the following format: `{{llx lly} {urx ury}}`.

`-excluded_areas areas`

Sets the ExcludedAreas property. These are the areas of the design that were excluded from the DRC check. To specify the areas, specify one or more rectangles using the following format: `{{llx lly} {urx ury}}`.

`-command command_string`

Sets the "Command" property. This is the command string that was used to invoke the DRC or LVS check that produced the error view.

DESCRIPTION

This command sets error view properties that help with the interpretation of the error data. Error view properties are also a way to record how the error objects in the error view were created.

SEE ALSO

`get_error_view_property(2)`

set_extraction_options

Sets the parameters that influence extraction.

SYNTAX

```
status set_extraction_options
[-max_cap_scale max_cap_scaling]
[-min_cap_scale min_cap_scaling]
[-max_res_scale max_res_scaling]
[-min_res_scale min_res_scaling]
[-max_ccap_scale max_ccap_scaling]
[-min_ccap_scale min_ccap_scaling]
[-max_net_ccap_thres max_net_ccap_threshold]
[-min_net_ccap_thres min_net_ccap_threshold]
[-max_net_ccap_ratio max_net_ccap_ratio]
[-min_net_ccap_ratio min_net_ccap_ratio]
[-max_net_ccap_avg_ratio max_net_ccap_avg_ratio]
[-min_net_ccap_avg_ratio min_net_ccap_avg_ratio]
[-max_process_scale max_process_scaling]
[-min_process_scale min_process_scaling]
[-no_obstruction]
[-no_break_segments]
[-max_segment_length max_segment_length]
[-real_metalfill_extraction none | floating | grounded | auto]
[-virtual_shield_extraction true | false]
[-fan_out_thres high_fan_out_threshold]
[-default]
```

Data Types

<i>max_cap_scaling</i>	float
<i>min_cap_scaling</i>	float
<i>max_res_scaling</i>	float
<i>min_res_scaling</i>	float
<i>max_ccap_scaling</i>	float
<i>min_ccap_scaling</i>	float
<i>max_net_ccap_threshold</i>	float
<i>min_net_ccap_threshold</i>	float
<i>max_net_ccap_ratio</i>	float
<i>min_net_ccap_ratio</i>	float
<i>max_net_ccap_avg_ratio</i>	float
<i>min_net_ccap_avg_ratio</i>	float
<i>max_process_scaling</i>	float
<i>min_process_scaling</i>	float
<i>max_segment_length</i>	integer
<i>high_fan_out_threshold</i>	integer

ARGUMENTS

-max_cap_scale *max_cap_scaling*
Specifies the factor to be multiplied by the maximum capacitance. The default value is 1.0.

`-min_cap_scale min_cap_scaling`
 Specifies the factor to be multiplied by the minimum capacitance. The default value is 1.0.

`-max_res_scale max_res_scaling`
 Specifies the factor to be multiplied by the maximum resistance. The default value is 1.0.

`-min_res_scale min_res_scaling`
 Specifies the factor to be multiplied by the minimum resistance. The default value is 1.0.

`-max_ccap_scale max_ccap_scaling`
 Specifies the factor to be multiplied by the maximum coupling capacitance. The default value is 1.0.

`-min_ccap_scale min_ccap_scaling`
 Specifies the factor to be multiplied by the minimum coupling capacitance. The default value is 1.0.

`-max_net_ccap_thres max_net_ccap_threshold`
 Specifies the net-to-net filtering threshold for the maximum coupling capacitance. The unit is your main library unit. The default value is 0.003 pF.

`-min_net_ccap_thres min_net_ccap_threshold`
 Specifies the net-to-net filtering threshold for the minimum coupling capacitance. The unit is your main library unit. The default value is 0.003 pF.

`-max_net_ccap_ratio max_net_ccap_ratio`
 Specifies the net-to-net filtering ratio for the maximum coupling capacitance. The default value is 0.03 of the total capacitance.

`-min_net_ccap_ratio min_net_ccap_ratio`
 Specifies the net-to-net filtering ratio for the minimum coupling capacitance. The default value is 0.03 of the total capacitance.

`-max_net_ccap_avg_ratio max_net_ccap_avg_ratio`
 Specifies the net-to-net filtering ratio based on average net-to-net ccap for the maximum coupling capacitance. This ratio can be bigger than 100% of average coupling capacitance. The default value is 0, which means it is off.

`-min_net_ccap_avg_ratio min_net_ccap_avg_ratio`
 Specifies the net-to-net filtering ratio based on average net-to-net ccap for the minimum coupling capacitance. This ratio can be bigger than 100% of average coupling capacitance. The default value is 0, which means it is off.

`-max_process_scale max_process_scaling`
 Specifies the maximum process scaling factor. The default value is 1.0.

`-min_process_scale min_process_scaling`
 Specifies the minimum process scaling factor. The default value is 1.0.

-no_obstruction
Ignores routing obstructions during extraction.

-no_break_segments
Specifies that long wire segments are not broken during extraction.

-max_segment_length *max_segment_length*
Specifies the maximum length of wire segment. This option has no effect if the **-no_break_segments** option is used. The default value is 50um.

-real_metalfill_extraction none | floating | grounded | auto
This option applies to post route extraction only. It specifies the metal fill extraction mode. The default value is **none**. Use this option in conjunction with the **set_tlu_plus_files** command. The valid values have different meaning depending on **set_tlu_plus_files**
If you specify emulation TLUPlus files in **set_tlu_plus_files** using **-max_emulation_tluplus** and/or **-min_emulation_tluplus**, and set **-real_metalfill_extraction** to **none**, the tool performs metal fill emulation extraction as follows:

- Does not read in metal fill polygons from the MilkyWay database.
- Uses emulation TLUPlus files in the extraction process.

If you set the value to **floating**, **grounded**, or **auto**, the tool performs real metal fill extraction as follows:

- Reads in metal fill polygons for processing.
- Uses normal *tluplus* files specified with **-max_tluplus** and/or **-min_tluplus** in the extraction.

If only **-max_tluplus** or **-min_tluplus** are specified in the **set_tlu_plus_file** (without **-max_emulation_tluplus** and **-min_emulation_tluplus**), and you set **-real_metalfill_extraction** to **none**, the tool performs normal extraction, without reading in the fill polygons.

There are different models when handling metal fill polygons in real metal fill extraction, based on the option value you selected:

none - The tool performs either normal extraction or emulation metal fill extraction, depending on the settings used in the **set_tlu_plus_files** command.

floating - The tool reads in metal fill polygons for processing use normal TLUPlus files specified using **-max_tluplus** and/or **-min_tluplus** in the extraction. The metal fill polygons are as treated as floating and the tool reduces the coupling capacitance to the fill polygon to the next signal net.

grounded - The metal fill polygons as treated as grounded and uses grounding coupling capacitance to fill polygons to VDD/VSS.

auto - The metal fill polygons are treated as is, based on the fill wire track property and net ID.

-virtual_shield_extraction true | false
This option applies to post route extraction only. When this option is set to **true** (default value), ICC performs virtual shield extraction by considering shielding rules. Otherwise, shileding rules are not considered during extraction. Shielding rules are defined in Nondefault Routing rules (NDRs) using the **define_routing_rule** command. Both the nets with NDR defined and their neighbors may see their RC affected due to shielding effects when this option is set to **true**.

Note that as of version Y-2006.06-SP2, this option replaces the **physopt_enable_virtual_shield** variable.

-fan_out_thres *high_fan_out_threshold*

Specifies the high fanout threshold for extraction so that nets with a higher fanout than the threshold are not extracted. The default value is 1000.

-default

Resets all options to their default values.

DESCRIPTION

The **set_extraction_options** command specifies the parameters that influence the extraction engine.

The following options specify the scaling factors for capacitance, resistance, and coupling capacitance:

```
[-max_cap_scale max_cap_scaling]
[-min_cap_scale min_cap_scaling]
[-max_res_scale max_res_scaling]
[-min_res_scale min_res_scaling]
[-max_ccap_scale max_ccap_scaling]
[-min_ccap_scale min_ccap_scaling]
```

This set of scaling factors is treated separately from the scaling factors specified with the **set_delay_estimation_options** command, which controls preroute resistance and capacitance estimation.

The following options control the coupling capacitance filtering:

```
[-max_net_ccap_thres max_net_ccap_threshold]
[-min_net_ccap_thres min_net_ccap_threshold]
[-max_net_ccap_ratio max_net_ccap_ratio]
[-min_net_ccap_ratio min_net_ccap_ratio]
[-max_net_ccap_avg_ratio max_net_ccap_avg_ratio]
[-min_net_ccap_avg_ratio min_net_ccap_avg_ratio]
```

The unit used as the threshold is your main library unit. If your main library unit is in picofarad, and you want to specify 0.2 fF as the threshold, set the value to 0.0002. If your main library unit is in femtofarad, set the value to 0.2. The ratio controls the relativeness of the filtering, and the default value is 0.03 of the total capacitance.

You can also specify process scaling factors with the **-max_process_scale** and **-min_process_scale** options. Process scaling factors can impact both **extract_rc** and **extract_rc -estimate** command activity if **extract_rc -estimate** uses extractor or TLUPlus-based RC.

Use this command with the **set_tlu_plus_files** command.

LIMITATIONS

The command is only available in DC-Topographical mode.

Multicorner-Multimode Support

This command uses information from the current scenario only. Only R, C and CC scaling are scenario specific, the rest are global.

EXAMPLES

The following example runs the **set_extraction_options** command with a 1.20 -**max_cap_scale** factor and a 0.7 -**min_cap_scale** factor:

```
prompt> set_extraction_options -max_cap_scale 1.20 -min_cap_scale 0.7
```

SEE ALSO

```
extract_rc(2)
read_parasitics(2)
report_extraction_options(2)
set_delay_estimation_options(2)
set_tlu_plus_files(2)
define_routing_rule(2)
```


set_false_path

Removes timing constraints from particular paths.

SYNTAX

```
int set_false_path
[-rise | -fall] [-setup | -hold]
[-from from_list
  | -rise_from rise_from_list
  | -fall_from fall_from_list]
[-through through_list]
[-rise_through rise_through_list]
[-fall_through fall_through_list]
[-to to_list
  | -rise_to rise_to_list
  | -fall_to fall_to_list]
[-reset_path]
```

Data Types

<i>from_list</i>	list
<i>rise_from_list</i>	list
<i>fall_from_list</i>	list
<i>through_list</i>	list
<i>rise_through_list</i>	list
<i>fall_through_list</i>	list
<i>to_list</i>	list
<i>rise_to_list</i>	list
<i>fall_to_list</i>	list

ARGUMENTS

-rise

Marks rising delays false, as measured on the path endpoint. If you do not specify either **-rise** or **-fall**, both rise and fall timing are marked false.

-fall

Marks falling delays false, as measured on the path endpoint. If you do not specify either **-rise** or **-fall**, both rise and fall timing are marked false.

-setup

Marks setup (maximum) paths false. **-setup** disables setup checking for specified paths. If you do not specify either **-setup** or **-hold**, both setup and hold timing are marked false.

-hold

Marks hold (minimum) paths false. **-hold** disables hold checking for specified paths. If you do not specify either **-setup** or **-hold**, both setup and hold timing are marked false.

-from *from_list*

Specifies start points (clocks, ports, pins, or cells) of disabled paths. If

you do not specify a *from_list*, all paths to end points in *to_list* are disabled. *from_list* can include clocks, pins, or ports. If you specify a clock, all path startpoints related to the specified clock are affected. If you specify an internal pin, the pin must be a path startpoint (the clock pin of a flip-flop, for example). If a cell is specified, one path startpoint on that cell is affected.

-rise_from *rise_from_list*

Same as the **-from** option, except that the path must rise from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by rising edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of **-from**, **-rise_from**, and **-fall_from**.

-fall_from *fall_from_list*

Same as the **-from** option, except that the path must fall from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of **-from**, **-rise_from**, and **-fall_from**.

-through *through_list*

A list of path throughpoints (port, pin, or leaf cell names) of the current design. The false path applies only to paths that pass through one of the points in the *through_list*. If more than one object is included, you must enclose the objects in quotes or in '{}' braces. If you specify the **-through** option multiple times, the false path setting applies to paths that pass through a member of each *through_list* in the order the lists were given. That is, the path must first pass through a member of the first *through_list*, then through a member of the second list, and so on for every through list specified. If you use the **-through** option in combination with the **-from** or **-to** options, the false path applies only if the **-from** or **-to** conditions are satisfied and the **-through** conditions are satisfied.

-rise_through *rise_through_list*

Same as the **-through** option, but applies only to paths with a rising transition at the specified objects. You can specify **-rise_through** more than once in a single command invocation as with the **-through** option.

-fall_through *fall_through_list*

Same as the **-through** option, but applies only to paths with a falling transition at the specified objects. You can specify **-fall_through** more than once in a single command invocation as with the **-through** option.

-to *to_list*

Specifies end points (clocks, ports, pins, or cells) of paths disabled. If you do not specify *to_list*, all paths from start points in *from_list* are disabled. The *to_list* can include clocks, pins, or ports. If you specify a clock, all path endpoints related to the specified clock are considered. If you specify an internal pin, the pin must be a path endpoint (for example, the data pin of a flip-flop). If you specify a cell, one path endpoint on that cell is affected.

`-rise_to` *rise_to_list*

Same as the **-to** option, but applies only to paths rising at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths captured by rising edge of the clock at clock source, taking into account any logical inversions along the clock path. You can use only one of **-to**, **-rise_to**, and **-fall_to**.

`-fall_to` *fall_to_list*

Same as the **-to** option, but applies only to paths falling at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of **-to**, **-rise_to**, and **-fall_to**.

`-reset_path`

Removes existing point-to-point exception information on the specified paths. Only information of the same rise/fall or setup/hold type is reset. This is equivalent to using the **reset_path** command with similar arguments before the **set_false_path** is issued.

DESCRIPTION

Removes timing constraints from specified paths that you know do not affect circuit operation. **set_false_path** can disable both maximum delay (setup) checking and minimum delay (hold) checking.

The **set_false_path** command disables timing from path startpoints through path throughpoints to path endpoints. Path startpoints are input ports or register clock pins. Path throughpoints can be cells, pins, or ports. Path endpoints are register data pins or output ports.

To disable the timing at a particular cell in the design, use **set_disable_timing**. This removes certain timing arcs on a cell from the timing graph, so that paths along those arcs are not traced. The **set_false_path** command still allows tracing of the paths, but removes any timing constraints on them.

set_false_path is a point-to-point timing exception command. This means it assists in overriding the default single-cycle timing relationship for one or more timing paths. Other point-to-point timing exception commands include **set_max_delay**, **set_min_delay**, and **set_multicycle_path**.

If a path satisfies multiple timing exceptions, the following rules assist in determining which exceptions take effect. Rules referring to **-from** apply equally to **-rise_from** and **-fall_from**, and similarly for the rise and fall options of **-through** and **-to**.

1. Two **group_path** commands might conflict with each other. But a **group_path** exception by itself does not conflict with another type of exception. So the remaining rules apply for two **group_path** exceptions, or two non-**group_path** exceptions.

2. If both exceptions are **set_false_paths**, there is no conflict.
3. If one exception is a **set_max_delay** and the other is **set_min_delay**, there is no conflict.
4. If one exception is a **set_multicycle_path -hold** and the other is **set_multicycle_path -setup**, there is no conflict.
5. If one exception is a **set_false_path** and the other is not, the **set_false_path** takes precedence.
6. If one exception is a **set_max_delay** and the other is not, the **set_max_delay** takes precedence.
7. If one exception is a **set_min_delay** and the other is not, the **set_min_delay** takes precedence.
8. If one exception has a **-from pin** or **-from cell** and the other does not, the former takes precedence.
9. If one exception has a **-to pin** or **-to cell** and the other does not, the former takes precedence.
10. If one exception has any **-through** points and the other does not, the former takes precedence.

11. If one exception has a **-from clock** and the other does not, the former takes precedence.

12. If one exception has a **-to clock** and the other does not, the former takes precedence.

13. The exception with the more restrictive constraint then takes precedence. For **set_max_delay** and **set_multicycle_path -setup**, this is the constraint with the lower value. For **set_min_delay** and **set_multicycle_path -hold**, it is the constraint with the higher value.

To undo the effect of **set_false_path**, use **reset_path** or **reset_design**.

Use **report_timing_requirements** to list the point-to-point exceptions on a design.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example removes timing constraints on paths from "ff12" to "ff34".

```
prompt> set_false_path -from {ff12} -to {ff34}
```

The following example removes timing constraints on paths through "u14/Z" to "ff29/Reset" which are rising at the endpoint.

```
prompt> set_false_path -rise -through {u14/Z} -to {ff29/Reset}
```

The following example disables hold checking (minimum delay timing) for endpoints clocked by "PHI1". The flip-flops and latches clocked by "PHI1" are checked for setup violations, but not for hold violations.

```
prompt> set_false_path -hold -to [get_clocks PHI1]
```

The following example removes timing constraints for all paths that first pass through either "u1/Z" or "u2/Z" then pass through "u5/Z" or "u6/Z".

```
prompt> set_false_path -through {u1/Z u2/Z} -through {u5/Z u6/Z}
```

The following example disables rising timing paths through "U14/Z" to "ff29/Reset".

```
prompt> set_false_path -rise_through {U14/Z} -to {ff29/Reset}
```

SEE ALSO

`current_design(2)`
`reset_design(2)`
`reset_path(2)`
`set_disable_timing(2)`
`set_max_delay(2)`
`set_min_delay(2)`
`set_multicycle_path(2)`

set_fanout_load

Sets the **fanout_load** attribute to a specified value on specified output ports of the current design.

SYNTAX

```
int set_fanout_load  
value  
port_list
```

Data Types

```
value          float  
port_list      list
```

ARGUMENTS

value

Specifies the value to which the **fanout_load** attribute is to be set on the ports in *port_list*. *value* must be expressed in units consistent with the **max_fanout** and **fanout_load** values in the technology library used during optimization.

port_list

Specifies the ports in the current design on which the **fanout_load** attribute is to be set. If more than one port is specified, they must be enclosed either in quotes or in braces ({}).

DESCRIPTION

Sets the **fanout_load** attribute to a specified value on specified output ports in the current design. **compile** attempts to ensure that the sum of this value, together with all **fanout_load** attributes for pins connected to the driver that drives this port, does not exceed the driving pin's **max_fanout** capability. The default **fanout_load** value for a port is 0.0.

NOTE: Bidirectional ports are not included in maximum fanout calculations, so using the **set_fanout_load** command with a bidirectional port has no effect on **compile**.

Use **remove_attribute** or **reset_design** to remove **fanout_load** values from ports. Use **report_port** to list fanout load values on ports.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following command sets a fanout load of 2 units on all output ports:

```
prompt> set_fanout_load 2 all_outputs()
```

SEE ALSO

```
all_outputs(2)  
remove_attribute(2)  
report_port(2)  
reset_design(2)  
set_max_fanout(2)
```


set_fast_mode

Turns on or turns off fast mode.

SYNTAX

```
status set_fast_mode  
[-no_congestion]  
[set_fast_mode]
```

ARGUMENTS

-no_congestion
Prevents the place_opt stage of the fast mode from performing congestion removal at the end.

set_fast_mode
Turns fast mode on or off. Valid values are **true** or **false**.

DESCRIPTION

This command takes a boolean value: when it is true, the fast mode is turned on; when it is false, the fast mode is turned off. If the boolean value is not given, by default it turns on the fast mode. Option specified by **set_fast_mode** take precedence over the mega commands. In other words, the command **set_fast_mode** changes the flow effort setting in the mega commands. But it does not affect any atomic commands. By default, the fast mode has congestion removal at the end of place_opt stage. To disable it, specify **-no_congestion** option.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example turns on the fast mode without congestion removal in place_opt stage.

```
prompt> set_fast_mode true -no_congestion  
Info: turn on fast mode
```

SEE ALSO

report_fast_mode(2)

set_feasibility_options

Sets feasibility options for place_opt and reporting commands.

SYNTAX

```
status set_feasibility_options
[-enable true | false]
[-zero_path_violations true | false]
[-zero_wire_load_violations true | false]
[-zero_path_margin zero_path_margin_num]
[-zero_wire_load_margin zero_wire_load_margin_num]
[-io_margin io_margin_num]
[-group_io true | false]
[-design_slack_threshold slack_threshold_num]
[-slack_threshold_path_groups path_groups_list]
[-path_group_slack_threshold path_group_slack_threshold_num]
[-verbose_level verbose_level_num]
[-feasibility_reporting boolean-string]
```

Data Types

zero_path_margin_num	integer
zero_wire_load_margin_num	integer
io_margin_num	integer
slack_threshold_num	float
path_groups_list	list
path_group_slack_threshold_num	float
verbose_level_num	integer

ARGUMENTS

-enable true | false

A value of true enables the feasibility mode. The default value is false. Feasibility mode is used to identify the timing feasibility of the design and enable constraint relaxation for unachievable endpoints for reporting commands and the place_opt command. The required time at unachievable endpoints are relaxed based on the unachievable timing portion. The following reporting commands honor options set by set_feasibility_options: **report_qor**, **report_qor_snapshot**, **report_timing**, **report_constraints** and **get_timing_paths**. These reporting commands will issue reports based on relaxed constraints at endpoints. The **place_opt** command applies feasibility mode constraint relaxation before starting optimization. Based on the feasibility mode reporting commands and the feasibility mode place_opt command, you can identify feasible and nonfeasible parts of the design. The remaining options to the **set_feasibility_options** command are valid only when the **-enable** option is set to a value of true. When feasibility mode is turned off by using **-enable false**, constraint relaxation is turned off.

-zero_path_violations true | false

A value of true value enables an adjustment to a required time of Zero Path violating endpoints during feasibility place_opt. The required time adjustment for this option is related to the value set by the -

zero_path_margin option. The default value of this option is true. Zero path violations occur when the required time at a path endpoint is equal or smaller than arrival time at the startpoint of the path.

-zero_wire_load_violations true | false

A value of true enable an adjustment to required time of Zero Wire Load violating unachievable endpoints during feasibility place_opt. The required time adjustment for this option is related to the value set by the **-zero_wire_load_margin** option. Setting this option to true does not have any effect if high-fanout net synthesis has not been run on the design. In that case, setting this option will produce a warning message. The default value is true. All Zero Wire Load violations may not be unachievable. If the Zero Wire Load violation is treated as unachievable, then the required time on endpoints is adjusted.

-zero_path_margin zero_path_margin_num

Adds a margin at Zero Path violating endpoints during the feasibility flow. This option is valid only if **-zero_path_violation** is set to true. The *zero_path_margin_num* is a percentage value added to the required time on top of the absolute value of existing Zero Path Violating slack. The default value is 0. The value of *zero_path_margin_num* should be ≥ 0 . The *zero_path_margin_num* is used to calculate the adjusted required time at endpoints as follows:

$$\text{adjusted required time} = \text{required time} + (1 + \text{zero\_path\_margin\_num}/100)\text{ZP}$$

-zero_wire_load_margin zero_wire_load_margin_num

Adds a margin at Zero Wire Load violating endpoints during the feasibility flow. This option is valid only if **-zero_wire_load_violation** is set to true. The *zero_wire_load_margin_num* is a percentage value added to the required time on top of the absolute value of the existing Zero Wire Load Violating slack. The default value is 0. The value of *zero_wire_load_margin_num* should be ≥ 0 . The *zero_wire_load_margin_num* is used to calculate the adjusted required time at endpoints as follows:

$$\text{adjusted required time} = \text{required time} + (1 + \text{zero\_wire\_load\_margin\_num}/100)\text{ZWL}$$

-io_margin io_margin_num

Enables a margin addition to Input and Output endpoints. The *io_margin_num* is a percentage value added on top of the existing required time value for relaxing the required time at Input and Output endpoints. The default is 0. The value of *io_margin_num* should be ≥ 0 . The *io_margin_num* is used to calculate the adjusted required time at endpoints as follows:

$$\text{adjusted required time} = (1 + \text{io\_margin\_num}/100) * \text{required time}$$

To ignore Input and Output endpoints during **place_opt** optimization you can set the *io_margin_num* value to a very high number. Limiting optimization on Input and Output endpoints will expose the good, constrained parts of the design for optimization, but in some cases, runtimes might be worse.

-group_io true | false

A value of true groups input and output paths into one path group. The default value is true. If you set this value to false, input and output paths are not grouped. If this option is set to true, and the input and output paths are already in multiple path groups, the tool will create new path groups exclusively for input and output paths.

`-design_slack_threshold slack_threshold_num`

Specifies the slack threshold to be applied to the entire design. The optimization engine will skip endpoints with slack equal to or smaller than the given slack threshold value. The default value is MIN_INT, which means that the default value will not change the optimization behavior. This option is layered, which means you can specify different **design_slack_threshold** values for different layers.

`-slack_threshold_path_groups path_groups_list`

Specifies that endpoints belonging to any path group in the given will be filtered with the slack threshold value given by the - **path_group_slack_threshold** option and this value will override the `design_slack_threshold` value, if given. If the **design_slack_threshold** is not given, then only path_groups that have `slack_threshold_path_groups` given will be affected. No filtering will be done on unspecified path groups. This option must be used in conjunction with **-path_group_slack_threshold** option, otherwise an error will be reported. This option is also layered, so you can specify different values for each path group in different scenarios. To specify different values for different path_groups, simply group those with the same slack threshold and specify those with different slack thresholds in separate calls of the command. If a path group is specified more than once, then the last one to be specified takes effect.

`-path_group_slack_threshold path_group_slack_threshold_num`

The argument is a floating point number and must be used in conjunction with the **-slack_threshold_path_groups** option, otherwise an error will be reported. This option specifies the slack value to be used to filter and adjust the endpoints belonging to path groups specified with the - **slack_threshold_path_groups** option.

`-verbose_level verbose_level_num`

This options allows for additional information to be printed every time the endpoint margins are adjusted according to the feasibility setting. The value of `verbose_level_num` should be ≥ 0 and the default value is 0. Higher settings will cause more information to be printed.

DESCRIPTION

Feasibility mode enables constraint relaxation for unachievable endpoints for reporting commands and the **place_opt** command. During feasibility mode unachievable endpoints are identified and required time on the unachievable endpoints are relaxed based on the unachievable timing portion. Endpoints with Zero path violation and Zero wire load violations are considered unachievable endpoints.

The following reporting commands honor options set by **set_feasibility_options**: **report_qor**, **report_qor_snapshot**, **report_timing**, **report_constraints** and **get_timing_paths**. These reporting commands will report QoR and timing based on relaxed constraint at endpoints. In the **report_timing** command, the adjusted timing and reason is also reported, if applicable. In the **report_qor** command, the report will be two columns, under original constraints and adjusted constraints, respectively. The **place_opt** command applies feasibility mode constraint relaxation before starting optimization. Based on the feasibility mode reporting commands and feasibility mode `place_opt`, you can identify feasible and non-feasible parts of the design. Feasibility mode also provides the functionality to group Input and Output

paths in separate path group and also ability to relax required time at Input and Output endpoints.

Feasibility mode also enables skipping optimization on endpoints which have slack equal to or smaller than the value specified by the slack threshold argument (either **-design_slack_threshold** or **-path_group_slack_threshold**). The slack threshold functionality should be used on optimized design as this approach is intended to skip endpoints identified to be difficult to optimize after initial optimization.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example sets zero_path margin to 10% and turns off zero_wire_load violation adjustment and enables feasibility mode:

```
prompt>set_feasibility_options -enable true -zero_path_margin 10 -  
zero_wire_load_violation false
```

The following example turns off zero_path and zero_wire_load violation but adds 20% margin to input and output end points (input and output paths are grouped together as -group_io option is ON by default) and enables feasibility mode:

```
prompt>set_feasibility_options -enable true -zero_path_violation false -  
zero_wire_load_violation false -io_margin 20
```

The following example turns on slack threshold filtering:

```
prompt>set_feasibility_options -enable true -design_slack_threshold -10 -  
slack_threshold_path_groups "clk1 clk2 path1" -path_group_slack_threshold -9  
prompt>set_feasibility_options -slack_threshold_path_groups "clk3 clk4 path1" -  
path_group_slack_threshold -8
```

As the options are additive and overriding, we have the following slack thresholds:

clk1	: -9
clk2	: -9
clk3	: -8
clk4	: -8
path1	: -8
other path groups	: -10

The following example turns off feasibility mode:

```
prompt>set_feasibility_options -enable false
```

SEE ALSO

`place_opt(2)`
`report_feasibility_options(2)`
`get_adjusted_endpoints(2)`
`report_adjusted_endpoints(2)`
`report_qor(2)`
`report_qor_snapshot(2)`
`report_timing(2)`
`get_timing_paths(2)`

set_fix_hold

Sets a **fix_hold** attribute on clocks in the current design.

SYNTAX

```
int set_fix_hold  
clock_list
```

Data Types

clock_list list

ARGUMENTS

clock_list
A list of clocks in the current design.

DESCRIPTION

Creates a **fix_hold** attribute on the named clock objects in the current design. **fix_hold** informs **compile** that hold time violations of the specified clocks should be fixed. To fix a hold violation requires slowing down data signals. **compile** will fix hold violations during the **Fixing Design Rules** step, but only if the maximum delay cost is not increased, or if the **set_cost_priority** command is used to prioritize hold violations ahead of maximum delay cost. **set_fix_hold** can violate setup and fix hold as long as **WNS** is not made worse. **compile** might fix hold and violate setup on non_critical paths.

The **report_clock -attributes** command identifies the clocks that have the **fix_hold** attribute present.

Use the **remove_attribute** command to undo the **set_fix_hold** command.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following command sets a **fix_hold** attribute on clock "clk1".

```
prompt> set_fix_hold clk1
```

The following command removes the **fix_hold** attribute from clock "clk1".

```
prompt> remove_attribute [get_clocks clk1] fix_hold
```

SEE ALSO

`create_clock(2)`
`current_design(2)`
`remove_attribute(2)`
`report_clock(2)`
`reset_design(2)`
`set_prefer(2)`

set_fix_hold_options

Specifies options for hold fixing during optimization.

SYNTAX

```
int set_fix_hold_options  
-default  
-prioritize_tns  
-prioritize_min  
[-preferred_buffer]
```

ARGUMENTS

-default

Specifies default option for hold fixing. The tool rejects any hold fix solution that degrades WNS or max transition. The user may expect some TNS degradation.

-prioritize_tns

Specifies the option to the tool to reject any hold fix solution that degrades TNS (total negative slack) as well as WNS and max transition .The user should be aware that protecting both WNS and TNS sometimes severely restricts hold fixing.

-prioritize_min

Specifies an option to the tool to prioritize min_delay ahead of max_delay. If the user specifies this option, the tool will try more to fix most of the hold violations. The user may observe timing degradation after the hold fixing is performed. The tool can sacrifice max-delay to fix min-delay. This option is recommended only if user wants to fix hold violations aggressively without considering the impact to timing.

DESCRIPTION

This command defines the options to control hold fixing inside place_opt, clock_opt and route_opt. This command enables the user to provide priority between min_delay and max_delay.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows how to set hold fixing options

```
prompt> set_fix_hold_options -default
```

```
prompt> set_fix_hold_options -prioritize_tns
```

SEE ALSO

`set_fix_hold(2)`

set_fix_multiple_port_nets

Sets the **fix_multiple_port_nets** attribute to a specified value on the current design or a list of designs.

SYNTAX

```
int set_fix_multiple_port_nets  
-default | -all  
[-feedthroughs]  
[-outputs]  
[-constants]  
[-buffer_constants]  
[design_list]
```

ARGUMENTS

-default

Removes the **fix_multiple_port_nets** attribute so that **compile** will use its default priority, that is no multiple port nets fixing.

-all

Inserts the same as **-feedthroughs -outputs -constants**. Note that logic constants are duplicated, not buffered. To buffer logic constants, use the **-buffer_constants** option with the **-all** option.

-feedthroughs

Inserts buffers to isolate input ports from output ports at all levels of the hierarchy.

-outputs

Inserts buffers so that no cell driver pin drives more than one output port at any level of the hierarchy.

-constants

Duplicates logic constants so that no constant drives more than one output port at any level of the hierarchy.

-buffer_constants

Buffers logic constants instead of duplicating them.

design_list

Specifies a list of designs. The default is the current design.

DESCRIPTION

Sets the **fix_multiple_port_nets** attribute on a list of designs. If you do not specify a design list, the current design is used.

This attribute controls whether **compile** inserts extra logic into the design to ensure that there are no feedthroughs, or that there are no two output ports connected to the same net at any level of hierarchy.

The default is not to add any extra logic into the design to fix such cases.

Certain three-state nets cannot be buffered, because this changes the logic functionality of the design.

If **set_fix_multiple_port_nets** is specified more than once on a design, the most recent setting is used and the earlier values are removed. To undo **set_fix_multiple_port_nets**, use the **-default** option, the **remove_attribute** command, or the **reset_design** command.

To view the current setting of the **fix_multiple_port_nets** attribute, use **report_compile_options**.

For some backward compatibility, if **compile** is run on a design with no **fix_multiple_port_nets** attribute and the obsolete variable **compile_fix_multiple_port_nets** is set to true, the attribute is set on the design to correspond with **set_fix_multiple_port_nets -all**. The attribute always has precedence over the variable setting. If **compile** is run when the attribute exists and has a different value than the variable, a warning is printed that the variable is being ignored. Consequently, if the choice of multiple port net fixing has to be changed between two **compile** commands on the same design, the **set_fix_multiple_port_nets** command must be used since changes to the **compile_fix_multiple_port_nets** variable after the first compile will be ignored.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets the **fix_multiple_port_nets** attribute so that feedthroughs are buffered and constants are duplicated.

```
prompt> set_fix_multiple_port_nets -feedthroughs -constants
```

The following example sets the **fix_multiple_port_nets** attribute so that multiple outputs and constants are buffered.

```
prompt> set_fix_multiple_port_nets -outputs -buffer_constants
```

SEE ALSO

current_design(2)
remove_attribute(2)
reset_design(2)

set_flip_chip_bump_attributes

Sets flip-chip bump cell attributes for flip-chip routing.

SYNTAX

```
status set_flip_chip_bump_attributes  
-bump_cells collection_of_bump_cells  
-target_edges {target_edges}
```

Data Types

<i>collection_of_bump_cells</i>	collection
<i>target_edges</i>	string

ARGUMENTS

```
-bump_cells collection_of_bump_cells  
    Specifies the names of the bump cells on which attributes are set.
```

```
-target_edges {target_edges}
```

You can specify the target edges of flip-chip bump cells. The flip-chip router connects wires to the bump target edges you specify. Normally, the flip-chip router uses 90 degree routing to connect the net to one of the four targets, north, east, south, or west. If turning on 45 degree routing, the flip-chip router will connect the net to one of the eight targets (each side of the bump pin). Setting the target edges value with this option will allow the flip-chip router to connect to specified targets.

The available target edges are E|S|W|N|NW|NE|SE|SW|Free, where E is east, S is south, W is west, N is north, NW is northwest, NE is northeast, SE is southeast, SW is southwest and free means that the router is free to route to any target edge. The values will be saved with the design. If the *target_edges* value has never been set to the bump cell, then the default value is Free.

The -target_edges is a soft rule. The flip-chip router may connect to a non-target edge if no target edges are accessible.

DESCRIPTION

This command specifies attributes on bump cells for the flip-chip router to honor. The values are saved with the design. It is recommended that you run the **report_flip_chip_bump_attributes** command first to check the attribute settings in your design.

EXAMPLES

```
The following example sets target edges of North, East, and South on all bump ce  
lls  
(cells with mask_layout_type of flip_chip_pad):  
prompt> set_flip_chip_bump_attributes -bump_cells * \  
-target_edges {N E NE}
```

Set bump target_edges attribute on 1845 bump cell(s)

1

The following example sets a target edge of South on specified bump cells:

```
prompt> set_flip_chip_bump_attributes -bump_cells {SNPS_BUMP_1 SNPS_BUMP_2} \  
       -target_edges {S}
```

Set bump target_edges attribute on 2 bump cell(s)

1

SEE ALSO

report_flip_chip_bump_attributes(2)

set_flip_chip_cell_site

Modifies flip_chip_cell_site properties to set different flip chip driver legal location constraints.

SYNTAX

```
status_value set_flip_chip_cell_site
-flip_chip_site {site_list}
[-personality {personality_list}]
[-
personality_type_constraints {{personality_type max_num} {personality_type max_num}
...}]
[-
reserved_for_cell {{cell_ref {row_index col_index}...} {cell_ref {row_index col_index}...} ...}]
```

Data Types

<i>site_list</i>	collection
<i>personality_list</i>	list
<i>personality_type</i>	string
<i>max_num</i>	integer
<i>cell_ref</i>	collection
<i>row_index</i>	positive integer
<i>col_index</i>	positive integer

ARGUMENTS

-flip_chip_site{site_list}
Specifies a collection of flip chip driver cell sites to be modified.

-personality{personality_list}
Specifies a list of personality types which are allowed to place in flip chip driver ring or array. Each personality type is a string.

-personality_type_constraints{{personality_type max_num} {personality_type max_num}...}
Specifies the maximum number of flip chip driver of the particular personality type that can be placed in a flip chip driver island. Each driver island can have up to five personality constraints. If constraint of a personality type is not specified, there will be no limit on the number of drivers of such personality type.

-reserved_for_cell {{cell_ref {row_index col_index}...} {cell_ref {row_index col_index}...} ...}}
Specifies that a certain location pointed by the row and column indices in an flip chip driver island is reserved for a certain flip chip driver cell. Only the specified flip chip driver can be placed in the location. By default, all drivers can be placed on all locations.

DESCRIPTION

This command modifies the writable attributes of a selected flip chip cell site, including the site's X and Y dimension, personality type constraint and slots reserved for a certain cell reference.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example changes a flip chip array's dimension to 12 columns and 4 rows. The "blue" type driver can take no more than 4 slots, the "red" type driver can take no more than 6 slots. The slot at row 3 and column 3 is reserved for "ESD" and the slot at column 4 and row 1 is reserved for "SHOULDER".

```
prompt> set_flip_chip_cell_site -flip_chip_site {FC_ARRAY_2}
      -personality_type_constraints {{"blue" 4} {"red" 6}} -reserved_for_cell
      {{ESD {2 3}} {SHOULDER {4 1}}}
```

SEE ALSO

```
set_flip_chip_driver_ring(2)
set_flip_chip_driver_array(2)
set_flip_chip_driver_island(2)
```


set_flip_chip_driver_array

Defines the legal locations for flip chip driver cells in a matrix configuration.

SYNTAX

```
status_value set_flip_chip_driver_array  
-personality_type {personality_type_list}  
-max_driver_size {width height}  
-start_point start_point  
-dimension {col row}  
-delta {pitch_x pitch_y}  
[-orientation N | W | S | E | FN | FS | FW | FE]
```

Data Types

<i>personality_type_list</i>	list
<i>width</i>	distance
<i>height</i>	distance
<i>start_point</i>	point
<i>col</i>	positive integer
<i>row</i>	positive integer
<i>pitch_x</i>	distance
<i>pitch_y</i>	distance

ARGUMENTS

-personality_type{*personality_type_list*}

Specifies a list of personality types of flip-chip driver cells to be placed in the array. Each personality type is a string name. If unspecified, no driver will be placed.

-max_driver_size{*width height*}

Specifies the maximum width and height in micron of driver cells that can be placed in the array location.

-start_point *start_point*

Specifies the lower-left coordinates of the lower left point of the flip chip driver array.

-dimension {*col row*}

Specifies the dimension of the array.

-delta{*pitch_x pitch_y*}

Specifies the pitch of the array in micron: x between columns and y between rows.

-orientationN | W | S | E | FN | FS | FW | FE

Specifies the orientation of drivers placed in the array. Default is the N orientation.

N	rotate 0
W	rotate 90
S	rotate 180

```
E    rotate 270
FN   rotate 0 and mirror X
FS   rotate 0 and mirror Y
FW   rotate 90 and mirror X
FE   rotate 90 and mirror Y
```

DESCRIPTION

The `set_flip_chip_driver_array` command defines the legal locations for flip chip driver cells in a array configuration. A flip chip driver is the IO driver that connects signal pin to pad pin (flip chip bump).

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example defines legal location for "blue" and "green" type driver cells in 2 columns and 5 row starting at coordinate (100, 100). The spacing between drivers is 80um on either dimension. The largest flip chip driver allowed is 20um by 20um.

```
prompt> set_flip_chip_driver_array -personality_type {"blue" "green"}
      -max_driver_size {20 20} -start_point {100 100} -dimension {2 5} -delta {80 80}
```

SEE ALSO

```
set_flip_chip_driver_ring(2)
set_flip_chip_driver_island(2)
```

set_flip_chip_driver_island

Defines the legal locations to place flip chip drivers in an island style.

SYNTAX

```
status_value set_flip_chip_driver_island
-start_point start_point
-repeat {num_columns num_rows}
-spacing {x_spacing y_spacing}
-island_size {width height}
-num_driver {num_x num_y}
[-filler cell_ref]
[-
personality_type_constraints {{personality_type max_num} {personality_type max_num}
...}]
[-default_orientation N | W | S | E | FN | FE | FS | FW]
[-orient_by_row]
[-compaction vertical | horizontal | none]
[-center_packing]
[-forced_orientation {{orientation col_index} {orientation col_index} ...}]
[-
reserved_for_cell {{cell_ref {col_index row_index}}...} {cell_ref {col_index row_index}
ex}...} ...}]
```

Data Types

<i>start_point</i>	point
<i>num_columns</i>	positive integer
<i>num_rows</i>	positive integer
<i>x_spacing</i>	distance
<i>y_spacing</i>	distance
<i>width</i>	distance
<i>height</i>	distance
<i>num_x</i>	positive integer
<i>num_y</i>	positive integer
<i>cell_ref</i>	collection
<i>personality_type</i>	string
<i>max_num</i>	positive integer
<i>orientation</i>	N W S E FN FE FS FW
<i>col_index</i>	positive integer
<i>row_index</i>	positive integer

ARGUMENTS

```
-start_point start_point
    Specifies the lower-left coordinates of the first (bottom-left) island.

-repeat {num_columns num_rows}
    Specifies the number of rows and columns of the flip chip island array.

-spacing {x_spacing y_spacing}
    Specifies the X and Y spacing between two adjacent flip chip driver islands.
```

The unit is in micron.

`-island_size {width height}`

Specifies the width and height of each flip chip driver island. The unit is in micron.

`-num_driver {num_x num_y}`

Specifies the number of flip chip drivers that can be placed in the X and Y direction of each island.

`-filler cell_ref`

Specifies a cell reference to be used as filler for empty space in the island when the flip chip driver compaction operation is performed. The `cell_ref` collection must contain one cell reference of the cell library.

`-personality_type_constraints {{personality_type max_num} {personality_type max_num} ...}`

Specifies the maximum number of flip chip driver of the particular personality type that can be placed in a flip chip driver island. Each island can have up to five personality constraints. If constraint of a personality type is not specified, there will be no limit on the number of drivers of such personality type.

`-default_orientation N | W | S | E | FN | FE | FS | FW`

Specifies the orientation of flip chip driver when placed. Default is N.

`-orient_by_row`

Specifies the forced orientation of the drivers are set by rows. If this option is used together with `-forced_orientation` option, the `col_index` will indicate `row_index` in practice. By default, the forced orientations of the drivers are set by rows.

`-compaction vertical | horizontal | none`

Specifies the flip chip island compaction direction. Compaction also shrinks the island size to make room for standard cells placement. If this option is not specified, a vertical compaction will be performed.

`-center_packing`

Forced the flip chip drivers to be packed towards the center of the island. The default packing orientation depends on the orientation of the drivers.

`-forced_orientation {{orientation col_index} {orientation col_index} ...}`

Forced a certain orientation on a flip chip driver placed on the specified column. The orientation is one of the values in {N | W | S | E | FN | FE | FS | FW}. If the `-orient_by_row` option is used, specify the row index position in this option instead of the column index position.

`[-reserved_for_cell {{cell_ref {col_index row_index} ...} {cell_ref {col_index row_index} ...} ...}]`

Specifies that a certain location pointed by the row and column indices in an island is reserved for a certain flip chip driver cell. Only the specified flip chip driver can be placed in the location. By default, all drivers can be placed on all locations.

DESCRIPTION

Defines flip chip cell sites that form the legal locations for flip chip drivers to be placed in an island fashion. An island can contain a two dimensional array of flip chip driver slots. Each slot can hold a flip chip driver. This command creates the flip chip cell sites and also defines constraints on flip chip driver placement.

allows you to define an array of islands while minimizing the number of filler cells inserted in each island..

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates an array of 20 by 20 (400) flip chip driver islands starting at coordinates (300, 600). The islands have spacing of 100um in either directions and the size of each island is 30um by 30um. Each island is capable of holding 9 rows by 2 columns of flip chip drivers. The first and last row of each island is reserved for cell "SHOULDER". The eighth row of the island is reserved for cell "ESD". The flip chip drivers on column 1 will be at the W orientation and that on column 2 will be at the FW orientation. Compaction will be performed in the vertical direction after driver placement.

```
prompt> set_flip_chip_driver_island -start_point {300 600} -repeat {20 20}
      -personality_type_constraints {"red" 10} {"green" 15}}
      -spacing {100 100} -island_size {30 30} -num_driver {2 9} -filler {"DUMMY"} -
forced_orientation {{W 1} {FW 2}} -
reserved_for_cell {"SHOULDER" {1 1} {2 1} {1 9} {2 9}} {"ESD" {1 8} {2 8}}}
```

1

SEE ALSO

```
set_flip_chip_cell_site(2)
set_flip_chip_driver_ring(2)
set_flip_chip_driver_array(2)
set_flip_chip_options(2)
```

set_flip_chip_driver_ring

Defines the legal locations for flip chip driver cells in a ring configuration.

SYNTAX

```
status_value set_flip_chip_driver_ring
-personality_type {personality_type_list}
-max_driver_size {width height}
-max_driver_num_per_ring driver_num
-min_driver_spacing driver_spacing
-ring_number ring_number
-ring_spacing ring_spacing
-outer_ring boundary_box
[-orientation N | W | S | E | FN | FS | FW | FE]
[-num_extra_spacing num_extra]
[-extra_spacing extra_spacing]
[-stagger_drivers]
```

Data Types

<i>personality_type_list</i>	list
<i>width</i>	distance
<i>height</i>	distance
<i>driver_num</i>	positive integer
<i>driver_spacing</i>	distance
<i>ring_number</i>	positive integer
<i>ring_spacing</i>	distance
<i>boundary_box</i>	rectangle
<i>num_extra</i>	positive integer
<i>extra_spacing</i>	distance

ARGUMENTS

```
-personality_type {personality_type_list}
    Specifies a list of personality types of flip-chip driver cells to be placed
    in the rings. Each personality type is a string name. If unspecified, no
    driver will be placed.

-max_driver_size {width height}
    Specifies the maximum width and height in micron of driver cells that can be
    placed in the rings.

-max_driver_num_per_ring driver_num
    Specifies the maximum number of drivers allowed in each ring.

-min_driver_spacing driver_spacing
    Specifies the minimum spacing between adjacent driver cells of the same ring
    in micron.

-ring_number ring_number
    Specifies the number of rings to be created.
```

`-ring_spacing ring_spacing`
 Specifies the spacing between adjacent rings in micron.

`-outer_ring boundary_box`
 Specifies the lower-left coordinate {llx lly} and up-right coordinate {urx ury} of the bounding box of the outermost ring in the format of {{llx lly} {urx ury}}.

`-orientation N | W | S | E | FN | FS | FW | FE`
 Specifies the orientation of drivers placed on the left edge of the ring. The top, right and bottom edge have an additional 90, 180 and 270 degrees of rotation respectively. Default is the N orientation.

N	rotate 0
W	rotate 90
S	rotate 180
E	rotate 270
FN	rotate 0 and mirror X
FS	rotate 0 and mirror Y
FW	rotate 90 and mirror X
FE	rotate 90 and mirror Y

`-num_extra_spacing num_extra`
 Specifies the size of driver cell group for which additional spacing is inserted. This option should be used together with the `-extra_spacing` option to specify, for example, an addition spacing of 10um for every 5 driver cells. By default, there will be no additional spacing.

`-extra_spacing extra_spacing`
 Specifies additional spacing for a group of driver cells in micron. This option must be used together with `-num_extra_spacing` option to specify, for example, an addition spacing of 10um for every 5 driver cells. By default, there will be no additional spacing.

`-stagger_drivers`
 This option creates a staggered flip chip driver placement in the even numbered ring. The tool places drivers on every other ring with an offset given by half of the sum of minimum driver spacing and driver width.

DESCRIPTION

This command defines the legal locations for flip chip driver cells in a ring configuration. A flip chip driver is a special cell that connects to a flip chip bump. This command is used for peripheral I/O style of flip chip driver placement where the drivers are placed on the four sides of a chip.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example defines ring style legal location constraint for flip chip drivers with the "blue" and "green" personality type if their sizes are smaller than 20um in width and 100um in height. Flip chip drivers are placed in two rings with the outer most ring at coordinate (100, 100) (900, 900) and a spacing of 15.0um between the two rings. Each ring can hold at most 150 drivers with a minimum spacing of 1.2um between two adjacent drivers of the same ring.

```
prompt> set_flip_chip_driver_ring\  
-personality_type {"blue" "green"}\  
-max_driver_size {20 100}\  
-max_driver_num_per_ring 150\  
-min_driver_spacing 1.2\  
-ring_number 2 -ring_spacing 15.0\  
-outer_ring {{100 100}{900 900}}
```

SEE ALSO

```
set_flip_chip_driver_array(2)  
set_flip_chip_driver_island(2)
```


set_flip_chip_grid

Creates equally spaced grid points for flip chip driver placement.

SYNTAX

```
status_value set_flip_chip_grid  
-grid_origin {llx lly}  
-x_step x  
-y_step y
```

Data Types

llx	distance
lly	distance
x	distance
y	distance

ARGUMENTS

```
-grid_origin {llx lly}  
    Specifies the lower-left coordinates in microns for the flip chip grid.  
  
-x_step x  
    Specifies the spacing of the grid in the X-direction in microns.  
  
-y_step y  
    Specifies the spacing of the grid in the Y-direction in microns.
```

DESCRIPTION

This command defines a secondary placement grid for flip chip drivers that is usually different from the placement grid used by standard cells. A flip chip driver must be placed on the secondary placement grid. In addition, all flip chip driver legal location sites must be on this secondary placement grid.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets the secondary flip chip placement grid starting at (0,0) with 10um spacing:

```
prompt> set_flip_chip_grid - grid_origin {0 0} -x_step 10 -y_step 10
```

SEE ALSO

```
set_flip_chip_driver_island(2)
```

```
set_flip_chip_driver_array(2)
set_flip_chip_driver_ring(2)
```

set_flip_chip_options

Sets general flip-chip driver placement options during virtual flat placement.

SYNTAX

```
status_value set_flip_chip_options  
[-disable_driver_placement]  
[-package_driven]  
[-one_softmacro_per_island]  
[-flip_chip_net_weight weight]  
[-softmacro_net_weight sm_weight]  
[-guardband_width {x y}]  
[-multiple width | height | both]
```

Data Types

<i>weight</i>	integer from 1 to 1000
<i>sm_weight</i>	integer from 1 to 1000
<i>x</i>	distance
<i>y</i>	distance

ARGUMENTS

- disable_driver_placement
Disables flip chip driver placement. Drivers will not be moved when standard cells are placed.
- package_driven
Enables the package driven placement flow. Use this option when flip chip driver to bump connections already exist and you want drivers to be placed as close as possible to their matching bumps. If this option is specified, the "-one_softmacro_per_island" cannot be used.
- one_softmacro_per_island
Instructs the tool to place flip chip drivers connected to the same soft macro on the same flip chip driver island. Other drivers not connected to the soft macros will not be placed on the island. If this option is specified, the "-package_driven" option cannot be used
- flip_chip_net_weight *weight*
Sets the net weights on the nets between the drivers and the bumps. Default net weight is 1 if not specified, giving equal consideration between the flip chip nets and a regular signal net. Weight value from 1 to 1000 can be given. The higher weight of a flip chip net indicates the higher cost of the same wire length than regular signal net, i.e the tool will try harder to place the drivers and bumps close.
- softmacro_net_weight *sm_weight*
Sets the net weights on nets connecting drivers and soft macros. Default net weight is one if not specified, giving equal consideration between the flip chip nets and a regular signal net. Weight value from 1 to 1000 can be given. The higher weight of a flip chip net indicates the higher cost of the same

wire length than regular signal net, i.e the tool will try harder to place the drivers and soft macros close.

`-guardband_width {x y}`

Sets the guardband of a flip chip island in the X and Y directions in micron. If not specified, the old value will be used. If never set before, 0 will be given as default value.

`-multiple width | height | both`

Allows drivers whose height or width or both is greater than a flip chip slot size to be placed inside the island. A driver will consume more than one slot in an island. Default is not allow multiple of either.

DESCRIPTION

Sets various options related to flip chip flow. The settings are Milkyway persistent and will be honored through out the design flow. A flip chip island is a cell site for flip chip drivers placement. An island is further partitioned into a two dimensional array of slots as defined by the `set_flip_chip_driver_island` command. Each slot can hold a unit sized flip chip driver. A flip chip driver is used to send or receive off-chip signal through a flip chip bump. A flip chip bump is a piece of top level metal that for forms electrical connection to the chip package. Standard cells may overlap with a flip chip bump but not allowed with a flip chip driver.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets the package driven flow and sets a 4um guardband on either directions of a flip chip island.

```
prompt> set_flip_chip_options -package_driven -guardband_width {4 4}
1
```

SEE ALSO

`set_flip_chip_driver_island(2)`

set_flip_chip_type

Assigns personality type to flip chip bumps or drivers according to the specified net or cell list.

SYNTAX

```
status_value set_flip_chip_type  
-personality_type type  
net_or_cell_list
```

Data Types

<i>type</i>	string
<i>net_or_cell_list</i>	collection

ARGUMENTS

-personality_type type
Specifies the personality type to be assigned to flip chip bumps or drivers.

net_or_cell_list
Specifies a collection of nets or cells where personality type will be assigned. The collection must contain nets, flip chip drivers or flip chip bumps. For specified flip chip drivers or bumps, the personality type will be assigned to the cells directly. And for specified net, the personality type will be assigned to the flip chip drivers and flip chip bumps it connects to.

DESCRIPTION

This command assigns personality types to flip chip bumps or drivers according to the specified net or cell list.

In a flip chip design, a bump is a piece of metal at the topmost layer of the chip that forms an electrical contact to the chip's packaging. A flip chip driver is an I/O circuitry that drives or receives signals from or to the chip thru the bump. A bump and its associated driver may be placed at different locations but need to be electrically connected.

A personality type is a string associated to a flip chip driver or a flip chip bump. Only drivers and bumps of identical personality type can be electrically connected to form a complete I/O circuit. Drivers and bumps without a personality type setting will have a default personality type, which is an empty string. The personality type settings are stored in the design and are Milkyway persistent. This command can be issued multiple times. A new personality type setting overrides an existing personality type setting.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets the instances "DRIVER_*" with the "red" personality type:

```
prompt> set_flip_chip_type -personality_type "red" {"DRIVER_*"}
```

SEE ALSO

`report_flip_chip_type(2)`

`assign_flip_chip_nets(2)`

set_fp_base_gate

Sets either a library leaf cell area or a user-specified cell area as the base unit area to use for gate equivalence calculations related to estimating the size of black boxes.

SYNTAX

```
status set_fp_base_gate
{-cell master_name | -area cell_area}
```

ARGUMENTS

```
-cell master_name
    Specifies the master name of the library leaf cell to use as the base unit
    area for gate equivalence calculations.

-area cell_area
    Specifies the cell area in square microns to use as the base unit area for
    gate equivalence calculations.
```

DESCRIPTION

Sets either a library leaf cell area or a user designated cell area as the base unit area to use for gate equivalence calculations related to estimating the size of black boxes.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

Set base gate area to cell master UNIT

```
prompt> set_fp_base_gate -cell UNIT
```

Set base gate area to 10 square microns

```
prompt> set_fp_base_gate -area 10
```

SEE ALSO

```
estimate_fp_black_boxes(2)
import_fp_black_boxes(2)
```

set_fp_black_boxes_estimated

Sets the specified objects as estimated so that they are flagged as not needing to be sized before floorplanning.

SYNTAX

```
status set_fp_black_boxes_estimated  
black_boxes
```

ARGUMENTS

```
black_boxes  
    black boxes to set estimated
```

DESCRIPTION

Specifies that the supplied black box objects have been estimated and so are valid for floorplanning.

Multicorner-Multimode Support

This command is not supported in a multi-scenario design flow.

EXAMPLES

Set alu1 black box state to estimated

```
> set_fp_black_boxes_estimated [get_cells alu1]
```

SEE ALSO

```
set_fp_black_boxes_unestimated(2)  
get_cells(2)
```


set_fp_black_boxes_unestimated

Sets the specified objects as unestimated so that they are flagged as needing to be sized correctly before floorplanning.

SYNTAX

```
status set_fp_black_boxes_unestimated  
black_boxes
```

ARGUMENTS

```
black_boxes  
    black boxes to set unestimated
```

DESCRIPTION

Sets the specified objects as unestimated so that they are flagged as needing to be sized correctly before floorplanning.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

Set alu1 black box state to unestimated

```
> set_fp_black_boxes_unestimated [get_cells alu1]
```

SEE ALSO

```
set_fp_black_boxes_estimated(2)  
get_cells(2)
```

set_fp_block_ring_constraints

Defines the power and ground rings that are automatically created around plan groups and macros when power network straps are synthesized by power network synthesis (PNS).

SYNTAX

```
status set_fp_block_ring_constraints
-add
  | -remove
  | -remove_all
  | -save_file file_name
  | -load_file file_name
[-block_type master | instance | plan_group | voltage_area]
-block blocks
[-all_blocks]
-nets nets
[-horizontal_layer layer]
[-vertical_layer layer]
[-horizontal_width distance]
[-vertical_width distance]
[-horizontal_offset distance]
[-vertical_offset distance]
[-spacing distance]
```

Data Types

<i>file_name</i>	string
<i>blocks</i>	collection or list
<i>nets</i>	collection or list

ARGUMENTS

-add
Add or update block ring constraints.
This option is mutually exclusive with the **-remove**, **-remove_all**, **-load_file**, and **-save_file** options; you must specify only one.

-remove
Remove the specified block ring constraints.
This option is mutually exclusive with the **-add**, **-remove_all**, **-load_file**, and **-save_file** options; you must specify only one.

-remove_all
Remove all block ring constraints.
This option is mutually exclusive with the **-add**, **-remove**, **-load_file**, and **-save_file** options; you must specify only one.

-save_file *file_name*
Saves the block ring constraints in the specified file.
This option is mutually exclusive with the **-add**, **-remove**, **-remove_all** and **-load_file** options; you can specify only one.

-load_file *file_name*
 Loads the block rings constraints from the specified file.
 This option is mutually exclusive with the **-add**, **-remove**, **-remove_all** and **-save_file** options; you can specify only one.

-block_type *master | instance | plan_group | voltage_area*
 Identifies the type of block specified in the **-block** option.
 If you specify **master**, rings are created around all instances that use the specified reference cells.
 If you specify **instance**, rings are created around the specified cell instances.
 If you specify **plan_group**, rings are created around the specified plan groups.
 If you specify **voltage_area**, rings are created around the specified voltage areas.
 The default type is 'master'.

-block *blocks*
 Specifies the blocks for which power and ground rings are created.
 There is no default value; you must specify this option.
 If you specify multiple blocks, power and ground rings are created for each block. For example, to create power and ground rings for blocks B1, B2, B3 and B4, enter **-block {B1 B2 B3 B4}**.

-all_blocks
 For a given block type, specifies all the blocks for power and ground ring creation.

-nets *nets*
 Specifies the power and ground nets used for the generated rings.
 There is no default value; you must specify this option.
 If you specify multiple nets, multiple rings are created around the specified blocks. The first net is the innermost block ring. For example, to create four rings, enter **-nets {VDD1 VSS1 VDD2 VSS2}**.

-horizontal_layer *layer*
 Specifies the horizontal layer for the block ring. You can specify only one horizontal layer per block ring constraint.
 The default is the highest layer with preferred horizontal direction.

-vertical_layer *layer*
 Specifies the vertical layer for the block ring. You can specify only one vertical layer per block ring constraint.
 The default is the highest layer with preferred vertical direction.

-horizontal_width *distance*
 Specifies the width (in microns) of the horizontal wires.
 The default is the minimum width defined in the technology file.

-vertical_width *distance*
 Specifies the width (in microns) of the vertical wires.
 The default is the minimum width defined in the technology file.

-horizontal_offset *distance*
 Specifies the horizontal offset (in microns) from the block boundary to the

block ring.

The default value is the minimum horizontal spacing defined in the technology file.

`-vertical_offset distance`

Specifies the vertical offset (in microns) from the block boundary to the block ring.

The default value is the minimum vertical spacing defined in the technology file.

`-spacing distance`

Specifies the spacing between block rings.

The default value is the minimum spacing defined in the technology file.

DESCRIPTION

This command defines the power and ground rings that are created around the specified blocks (macros or plan groups) when power network straps are synthesized by power network synthesis (PNS). Rectangular and rectilinear rings are supported.

Multicorner-Multimode Support

This command is not supported in a multi-scenario design flow.

EXAMPLES

The following example shows how to create power rings for the VDD net around all blocks with the R16X16M1 reference cell. The power ring is created on horizontal layer METAL5 with a ring width of 1.0 micron and an offset of 0.6 micron and on vertical layer METAL6 with a ring width of 1.0 micron and an offset of 0.6 microns.

```
prompt> set_fp_block_ring_constraints \  
-add \  
-block_type master \  
-block R16X16M1 \  
-horizontal_layer METAL5 \  
-horizontal_width 1.0 \  
-vertical_layer METAL6 \  
-vertical_width 1.0 \  
-horizontal_offset 0.6 \  
-vertical_offset 0.6 \  
-nets VDD
```

SEE ALSO

```
set_fp_rail_constraints(2)  
report_fp_rail_constraints(2)  
synthesize_fp_rail(2)
```

set_fp_clock_plan_options

Sets options for the clock planning clock tree synthesis engine.

SYNTAX

```
status set_fp_clock_plan_options
[-output_directory directory]
[-no_feeds_plan_group plan_groups]
[-clock_nets clock_nets]
-anchor_cell cell
[-route_mode detailed | global | none]
[-keep_block_tree true | false]
[-update_clock_latency true | false]
```

Data Types

<i>directory</i>	string
<i>plan_groups</i>	collection
<i>clock_nets</i>	collection
<i>cell</i>	collection

ARGUMENTS

-output_directory *directory*
Specifies the directory in which to write all constraint files, log files, and reports. By default, the tool uses a directory called cp_output.

-no_feeds_plan_group *plan_groups*
Specifies a list of plan groups in which feedthroughs are not allowed. By default, feedthroughs are allowed in all plan groups.

-clock_nets *clock_nets*
Specifies a list of clock nets on which to do clock planning.

-anchor_cell *cell*
Specifies a list of the cells that are inserted as the anchor cells for all plan groups. By default, the tool generates an error message.

-route_mode detailed | global | none
Specifies the type of routing to be performed on top-level clock nets. Use **detailed** to perform detailed routing on top-level clock nets. Use **global** to perform global routing. Use **none** to perform no routing. By default, the tool performs no routing.

-keep_block_tree true | false
Specifies whether the block-level clock tree built during clock planning will be kept. Use **true** to keep that block-level clock tree. Use **false** to remove this block-level clock tree when clock planning is finished. By default, the tool removes this block-level clock tree when clock planning is finished.

-update_clock_latency true | false
Specifies whether update_clock_latency will be done at the end of clock

planning. Use **true** to update clock latency. Use **false** to avoid latency updating. By default, clock planning finishes without updating clock latency.

DESCRIPTION

This command sets the proper environment before running the **compile_fp_clock_plan** command. For best results, specify **-no_feeds_plan_group** for all plan groups on which clock planning is performed.

The anchor cell is the buffer cell that is inserted in each plan group. All plan groups should use the same buffer type. Clock planning supports only buffer cells as anchor cells (inverters are not supported as anchor cells). You can use the **report_fp_clock_plan_options** command to report clock planning settings.

Multicorner-Multimode Support

This command is not supported in a multi-scenario design flow.

EXAMPLE

The following example sets the output directory to the name tmp_dir.

```
prompt> set_fp_clock_plan_options\  
-output_directory tmp_dir
```

The following example specifies the plan groups in which feedthroughs are not allowed.

```
prompt> set_fp_clock_plan_options\  
-no_feed_plan_groups [get_plan_groups *]
```

The following example sets the anchor cell to the name CLKBUF12 and selects global routing for the top-level clock nets.

```
prompt> set_fp_clock_plan_options\  
-clocks_nets CLK1 -anchor_cell CLKBUF12\  
-route_mode global
```

SEE ALSO

```
compile_fp_clock_plan(2)  
report_fp_clock_plan_options(2)
```

set_fp_flow_strategy

Sets the strategy for the hierarchical floorplanning flow. These settings are not persistent in the Milkyway database.

SYNTAX

```
status set_fp_flow_strategy  
[-plan_group_aware_routing true | false]  
[-top_level_routing_only true | false]
```

ARGUMENTS

`-plan_group_aware_routing true | false`

Sets to *true* to enable plan group-aware routing. When set to true, the subsequent **route_global**, **route_fp_proto**, and **place_fp_pins** commands read in any plan group definitions, and honor any constraints placed on the plan group. Constraints can be prohibitions on feedthroughs, TDF constraints, or pin blockages. In the virtual flat or pin cutting flow, you must set this value to *true*.

Sets to *false* to disable plan group-aware routing. When set to false, the subsequent **route_global**, **route_fp_proto**, and **place_fp_pins** commands do not read in any plan group definitions. Routing is run without any awareness of the plan groups and therefore, any constraints placed on the plan groups will have no effect.

This argument can affect the **place_fp_pins** command if, in addition to the soft macros for which you are placing pins, there are uncommitted plan groups. In this case, it may be important to take the plan groups into account. For example, if feedthroughs on the plan groups are prohibited, you want the soft macro pin placement to take that into account.

The default value is *false*.

`-top_level_routing_only true | false`

Sets to *true* to instruct the subsequent **route_global**, **route_fp_proto**, and **place_fp_pins** commands not to route any nets that are entirely contained within plan groups. Note that for this option to be effective, you must set the **-plan_group_aware_routing** option to *true*.

Sets to *false* to instruct the subsequent **route_global**, **route_fp_proto**, and **place_fp_pins** commands to route all signal nets.

The purpose of the **top_level_routing_only option** is to increase the routing speed. Intra-plan group routing is not strictly needed for creating feedthroughs and pin positions in the subsequent **analyze_fp_routing** and **commit_fp_plan_group** commands. If routing congestion inside the plan groups is significant, ignoring the intra-plan group nets could lower the quality of the pin placement.

DESCRIPTION

This command sets the parameters that affect the strategy for the hierarchical floorplanning flow.

When routing is plan group-aware, the tool not only honors any special constraints

placed on the plan groups such as pin spacing or TDF constraints, but it also enforces the hierarchy rules for boundary crossings. For example, an intra-plan group net (a net that has connections only inside a plan group) is not allowed to cross the plan group boundary but must be routed entirely within the plan group.

Setting the **-plan_group_aware_routing** option to true also gives priority to clock nets. During floorplanning, you may want to route the clock flat to generate clock pins on module boundaries and buffer these clocks later. The clock nets will be routed before any other nets so that their pin placement has priority over normal signal pins.

In the virtual flat or pin cutting flow, you must specify **-plan_group_aware_routing true**. If you route without awareness of plan groups, the **analyze_fp_routing** and **commit_fp_plan_groups** commands will not work properly.

The settings for this command are valid in the current session only.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

In the following example, the router reads in the plan group definitions, enabling the constraint prohibiting feedthroughs on plan groups to be effective.

```
prompt> set_fp_flow_strategy -plan_group_aware_routing true
prompt> set_fp_pin_constraints -allow_feedthroughs off
prompt> route_global
```

In the following example, the router does not read in the plan group definitions. Therefore the constraint prohibiting feedthroughs on plan groups have no effect. The router runs in its normal flat mode, without any regard for plan group boundaries.

```
prompt> set_fp_flow_strategy plan_group_aware_routing false
prompt> set_fp_pin_constraints -allow_feedthroughs off
prompt> route_global
```

In the following example, the router reads in the plan group definitions and does not route any nets that are entirely contained within single plan groups (intra-plan group nets). Note that the rest of the flow including the **analyze_fp_routing** and **commit_fp_plan_groups** commands, does not require intra-plan group routing. The flow is unaffected by this setting, with the exception for any impact on the quality of results.

```
prompt> set_fp_flow_strategy -plan_group_aware_routing true \
      -top_level_routing_only true
prompt> route_global
prompt> analyze_fp_routing -finalize_pins_and_feedthroughs plan_group
prompt> commit_fp_plan_groups
```


SEE ALSO

`route_global(2)`
`route_fp_proto(2)`
`set_fp_pin_constraints(2)`
`analyze_fp_routing(2)`
`commit_fp_plan_groups(2)`
`place_fp_pins(2)`

set_fp_macro_array

Specifies an array of macro cells.

SYNTAX

```
status set_fp_macro_array
-name string
[-elements collection_of_macro_cell_objects]
[-align_edge t | b | l | r | c | top | bottom | left | right | center]
[-align_pins {list of two pin objects}]
[-x_offset float]
[-y_offset float]
[-use_keepout_margin]
[-vertical]
[-rectilinear]
[-align_2d lb | lc | lt | rb | rc | rt | cb | cc | cr | left-bottom | left-
center | left-top | right-bottom | right-center | right-top | center-
bottom | center-center | center-top]
[-reset]
```

ARGUMENTS

-name *string*
Specifies name for the macro array.

-elements *collection_of_macro_cell_objects*
Specifies a collection or list-of_collection of macro cells which will compose the macro array. Each sublist represents a row in the array. An array starts from top most row. A row starts from left. For example, to describe the following array,
A11 A12 A13
A21 A22 A23
A31 A32 A33
specify this argument as shown below.
-elements [list [get_cells "A11 A12 A13"] \
[get_cells "A21 A22 A23"] \
[get_cells "A31 A32 A33"]]
This is a required argument when creating a macro array. It is not required when the **-reset** option is used to remove a macro array.

-align_edge t | b | l | r | c | top | bottom | left | right | center
Specifies an edge for the macro cells in one row to align with. Default is to align to the centers.

-align_pins {*list of two pin objects*}
Specifies list of pins to align two macro cells with pins. In the list, the first pin is a pin on reference cell (first cell in the array) and the second pin is a pin from second cell. This option takes effect only if specified array has two macro cells.

-x_offset *float*
Specifies the distance between two adjacent edges of macro cells on one row.

Unit is micron. The default value is 0, which means the cells abut to each other.

-y_offset *float*

Specifies the distance between two adjacent edges of macro cells on the same column of the next row. Unit is micron. The default value is 0, which means the cells abut to each other.

-use_keepout_margin

Instructs the tool to use the keepout margin to determine the spacing between two adjacent rows and/or columns of the array. This option is mutually exclusive with the **-x_offset** and **-y_offset** options.

-vertical

Instructs the tool to arrange an array vertically. If specified for an one-dimensional array, the tool arranges one dimensional macros in a vertical column. By default these arrays are created as horizontal rows. If specified for a two-dimensional array, the tool arranges the array in column based structure, instead of row based structure. It is like rotate a two dimensional array 90 degrees counter-clock-wise.

-rectilinear

Specifies the overall rectilinear shape of a two dimensional array. For a rectilinear macro array, you can specify different number of cells in different rows. By default, in design planning, a two-dimensional macro array is placed as a single object that is rectangular in shape, and all the rows(or columns) in the array have the same number of cells.

-align_2d lb | lc | lt | rb | rc | rt | cb | cc | cr | left-bottom | left-center | left-top | right-bottom | right-center | right-top | center-bottom | center-center | center-top

Specifies the edges for the macro cells in a row or column to be aligned with, in a heterogeneous two-dimensional macro array. This argument is optional. By default, the tool aligns the cells in the same column by their left edge, and aligns the cells in the same row by their bottom edge.

-reset

Removes the macro array constraint. This option is mutually exclusive with all other options except for **-name**.

DESCRIPTION

The **set_fp_macro_array** command defines an array of macro cells. The tool places macro cells in the array according to the parameters specified for the array.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows how to specify a 2x2 array for cell A11 A12 A21 A22 with

10 microns space between rows and columns.

```
prompt>set_fp_macro_array -name AA \  
        -elements [list [get_cells "A11 A12"] [get_cells "A21 A22"]] \  
        -x_offset 10 -y_offset 10
```

The following example shows how to remove a previously defined macro array ICRAMARRAY1.

```
prompt>set_fp_macro_array -name ICRAMARRAY1 -reset
```

SEE ALSO

```
create_fp_placement(2)  
set_fp_macro_options(2)
```

set_fp_macro_options

Sets constraints on the specified floorplan macro cells and macro cell arrays in design planning.

SYNTAX

```
status set_fp_macro_options  
collection_of_macro_objects  
[-legal_orientations list]  
[-anchor_bound t1 | t | tr | r | br | b | bl | l | tm | bm | lm | rm | c]  
[-x_offset float]  
[-y_offset float]  
[-align_pins list]  
[-side_channel {left right top bottoa}]  
[-reset]
```

Data Types

<i>collection_of_macro_objects</i>	collection
<i>left</i>	integer
<i>right</i>	integer
<i>top</i>	integer

ARGUMENTS

collection_of_macro_objects

Specifies the macro cells or macro cell arrays on which to set constraints. Note that some constraints apply only to macro cells and not to macro cell arrays.

Use the **get_cells** command to obtain the macro cell objects. For example, **[get_cells Input*]**.

-legal_orientations *list*

Specifies the orientations that the placement tool can use for the specified macro cells. You cannot specify this option for macro cell arrays.

You specify the legal orientations by using the following DEF syntax to specify one or more of the 8 possible orientations:

N, W, S, E, FN, FW, FS, FE

If you do not specify this option, the placement tool uses the legal orientations specified for the macro cell in the physical library.

Otherwise, the placement tool uses the common subset of orientations that are allowed by both this option and the physical library. If the common subset is empty because the orientations specified by this option conflict with the legal orientations defined in the physical library, the placement tool ignores the orientation constraint specified by this option and uses the legal orientations specified in the physical library.

You can specify a single legal orientation to force the placement tool to place the cell in the fixed orientation.

-anchor_bound t1 | t | tr | r | br | b | bl | l | tm | bm | lm | rm | c

Specifies the anchor bound (a kind of placement move bound) for the specified macro cells or macro cell arrays.

The valid keywords and their definitions are shown below:

Keyword	Definition
-----	-----
tl	top-left corner
t	top
tr	top-right corner
r	right
br	bottom-right corner
b	bottom
bl	bottom-left corner
l	left
tm	top-middle
bm	bottom-middle
lm	left-middle
rm	right-middle
c	center

An anchor bound constrains the placement tool so that it places the macro cell in one of the boundary areas of the block. This forces the macro cell to stay at one of the boundaries of the block. The default move bound is one-half or one-quarter of the core area associated with the specified anchor boundary. For example, if you specify **-anchor_bound** as **t**, the move bound is the top half of the core area. If you specify **-anchor_bound** as **tr**, the move bound is the top-right area of the core area. If the default move bound is not large enough to hold the macro cell in any dimension, the move bound is set to the cell width or height in that dimension.

An anchor bound is a soft constraint, which means that placement can always move the macro cell out of the bound if there is no space and the cost is too high.

-x_offset *float*

-y_offset *float*

Specifies the distance between the specified macro cells or macro cell arrays and the core edges.

If you set an anchor bound of top, bottom, top-middle, or bottom-middle, you should specify only the **-y_offset** option. If you set an anchor bound of left, right, left-middle, or right-middle, you should specify only the **-x_offset** option. If you set an anchor bound of top-left, top-right, bottom-left, or bottom-right, you can specify the **-x_offset** option, the **-y_offset** option, or both options.

-align_pins *list*

Aligns a specific port to a pin of the specified macro cells.

The argument for this option is a list of two pins: the reference port followed by the constrained pin. For example,

```
set_fp_macro_options [get_cell A1] -anchor_bound t \  
    -align_pins [list [get_ports "PD[0]"] [get_pins "A1/DATA[0]"]]
```

-side_channel {*left right top bottoa*}

Specifies the distance at which to place the specified macro cells away from the core edge. The values in the list represent the channel size in microns for the left, right, top, and bottom sides, respectively. You must specify a value for all four sides. Use 0 if a specific side does require a side channel.

set_fp_macro_options

-reset

Removes the constraints set with previous uses of the **set_fp_macro_options** command.

DESCRIPTION

The **set_fp_macro_options** command sets constraints for macro cells and macro cell arrays for virtual flat placement. This command constrains the **create_fp_placement** command in placing the macro cells and macro cell arrays in the floorplan. You use this command after you load the design and before you run the **create_fp_placement** command.

You can use this command to restrict macro cells and macro cell arrays in the following ways:

- To a particular region, by specifying the **-anchor_bound** option.
- To particular orientations (macro cells only), by specifying the **-legal_orientations** option.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows how to place the RAM1 cell in the top-right corner of the core area and in the south orientation.

```
prompt> set_fp_macro_options [get_cell "RAM1"] \  
-legal_orientations S -anchor_bound tr
```

The following example shows how to place the user-defined ICRamMacroArray1 macro array in the bottom-left corner of the core area.

```
prompt> set_fp_macro_options ICRamMacroArray1 \  
-anchor_bound bl
```

The following example shows how to place the user-defined ICRamMacroArray2 macro array at least 20 microns away from the left edge of the core area, 30 microns from its right edge, and 30 microns from its top edge.

```
prompt> set_fp_macro_options ICRamMacroArray2 \  
-side_channel {20 30 30 0}
```

SEE ALSO

create_fp_placement(2)
report_fp_macro_options(2)
set_fp_macro_array(2)

set_fp_pin_constraints

Sets pin assignment constraints that are honored during pin cutting and pin assignment.

SYNTAX

```
status set_fp_pin_constraints
[-allowed_layers layers]
[-pin_spacing pin_spacing_number]
[-hard_constraints {off | spacing | location | layer}]
[-pin_preroute_spacing preoute_spacing_number]
[-no_stacking stacking_allowed | pg_pins_only | signal_pins_only | all]
[-corner_keepout_num_wiretracks wiretracks_number
 | -corner_keepout_percent_side keepout_percentage]
[-exclude_sides side_numbers]
[-allow_feedthroughs off | on]
[-exclude_network]
[-exclude_clock_feedthroughs off | on]
[-exclude_scan_chain_net_feedthroughs off | on]
[-exclude_hfn_feedthroughs hfn_number]
[-exclude_feedthroughs nets]
[-incremental off | on]
[-nets nets | -exclude_nets nets]
[-keep_buses_together off | on]
[-bus_ordering lsb_to_msb | msb_to_lsb | scrambled | consistent_wirelengths]]
[-reserved_channel_threshold channel_size]
[-reserved_channel_nets nets]
[-scramble_skip skip_number]
[-use_physical_constraints off | on]
[-block_level]
[blocks]
```

Data Types

<i>layers</i>	collection
<i>pin_spacing_number</i>	integer
<i>preoute_spacing_number</i>	integer
<i>wiretracks_number</i>	integer
<i>keepout_percentage</i>	real
<i>side_numbers</i>	string
<i>hfn_number</i>	integer
<i>nets</i>	collection
<i>channel_size</i>	real
<i>skip_number</i>	integer
<i>blocks</i>	collection

ARGUMENTS

-allowed_layers *layers*

Specifies the set of metal layers allowed for pin placement. The argument is a collection of consecutive metal layers. A collection that contains nonconsecutive metal layers is not supported.

By default, the metal layers from M2 to the maximum metal layer specified earlier in the floorplan planner are allowed for pin placement. If the maximum metal layer has not been specified, the maximum available metal layer specified in the technology file is used as the default. If the maximum layer specified is beyond the maximum metal layer for the current design then the latter figure is used.

-pin_spacing *pin_spacing_number*

Specifies the number of wire tracks between adjacent pins or between a pin and a preroute wire that cuts across a soft macro edge (or a top-level cell edge, if **-block_level** is used) in the normal direction. Note that created pins are always snapped to wire tracks. The default pin-to-pin spacing is one wire track.

-hard_constraints {off | spacing | location | layer}

Specifies whether the pin spacing constraints, location constraints, and metal layer constraints are hard or soft constraints.

You can specify one or more of the following values:

- **off** (default)

The pin-to-pin spacing constraint specified by the **-pin_spacing** option is treated as a soft constraint.

- **spacing**

The pin-to-pin spacing constraint specified by the **-pin_spacing** option is treated as a hard constraint. An error occurs during pin assignment if there are insufficient pin slots to honor the constraint.

- **location**

The pin location constraint specified by the **read_io_constraints** or **set_pin_physical_constraints** command is treated as a hard constraint. Pins are created at the exact locations and pin assignment gives warnings if there are pin spacing violations.

- **layer**

The pin layer constraint specified by the **read_io_constraints** or **set_pin_physical_constraints** command is treated as a hard constraint. Pins are created at the layer based on the physical constraints and pin placement gives warnings if the specified pin layer is not within the allowed layer constraints.

The pin layer constraint can be treated as a hard constraint only when the pin location constraint is also treated as a hard constraint. See the **EXAMPLES** section for an example of setting hard pin layer constraints.

By default, the constraints are soft constraints.

-pin_preroute_spacing *preoute_spacing_number*

Specifies the number of wire tracks between pins and preroutes.

To allow enough space for routing, no pins are created on the section of a soft macro edge (or top-level cell edge, if **-block_level** is used) that is parallel to a preroute if that edge is closer to the preroute than to the specified number of wire tracks.

IMPORTANT: Please note that this spacing pertains to the distance between a macro edge and an edge of a preroute that runs parallel to the macro edge. When a preroute runs into a soft macro across a macro edge in the normal perpendicular direction, the distance between soft macro pins on that macro

edge and the preroute is dictated by the **-pin_spacing** option.
The default is three wire tracks.

-no_stacking *stacking_allowed | pg_pins_only | signal_pins_only | all*
Specifies how to handle pin overlaps across layers. The default method is *stacking_allowed*.
The supported methods are described below.

- **stacking_allowed** (default)

Signal and power and ground pins can be stacked.

- **pg_pins_only**

Power and ground pins cannot be stacked. Signals pins can be stacked.

- **signal_pins_only**

Signal pins cannot be stacked. Power and ground pins cannot be stacked.

- **all**

Signal and power and ground pins cannot be stacked.

Note: The **set_fp_pin_constraints** command operates only on signal pins.

-corner_keepout_num_wiretracks *wiretracks_number*

Specifies the distance in wire tracks from the corners of soft macros (or the top-level cell, if **-block_level** is used) where pin placement should begin. Pin assignment and pin cutting multiplies the specified argument by the maximum metal layer pitch (the maximum wire track distance for metal layers from minimum to maximum) and uses the resulting distance as the corner spacing. In a design with soft macros, for example, if the minimum layer is metal3 and the maximum layer is metal6 and the metal6 pitch is the largest, the corner spacing is calculated as the metal6 pitch multiplied by the number of wire tracks from the corners of soft macros.

This option can not be used together with the **-corner_keepout_percent_side** option. If neither option is specified, the default distance is five wire tracks.

-corner_keepout_percent_side *keepout_percentage*

Specifies the distance as a percentage of edge length from the corners of soft macros (or the top-level cell, if **-block_level** is used) where pin placement should begin. This option is similar to the **-corner_keepout_num_wiretracks** option, except that the corner keepout spacing is specified as a percentage of the edge length.

This option can not be used together with the **-corner_keepout_num_wiretracks** option. If neither option is specified, the default distance is five wire tracks.

-exclude_sides *side_numbers*

Specifies the soft macro edges or the top-level cell edges, **-block_level** is used. on which pins cannot be placed. The argument is a string consisting of a number or a collection of numbers separated by commas. Each number corresponds to a macro or plan group edge or top-level cell edge, if **-block_level** is used. The leftmost edge is numbered 1 (one). Other edges are numbered according to their order of traversal in the clockwise direction starting from edge 1. If there are multiple leftmost edges then edge 1 is the lowest leftmost edge. For example, to exclude sides 2, 4, and 6, enter

"2,4,6"

The default is 0 (no sides are excluded). IMPORTANT: The sides are numbered according to the actual placement of individual cell instances, not cell masters.

`-allow_feedthroughs off | on`

Specifies whether feedthrough ports can be created.

By default, this option is **off** (no feedthrough ports are created).

If set to **on**, pin assignment and cutting creates at least two feedthrough ports in the child cell. Each top-level net for which a feedthrough port is created is split into a set of new top-level nets and child-level nets (if feedthrough nets were created for the child net). Directions are assigned for the newly created feedthrough ports. Because new ports are created in the soft macros, the netlist is modified as well.

This option is silently ignored if `-block_level` is used.

If a plangroup boundary is coincident with voltage area boundary, and this feedthrough constraints setting are conflict with `set_fp_voltage_area_constraints`, voltage area's feedthrough constraints will be ignored.

`-exclude_network`

Propagates feedthrough exclusion nets through combinational circuits.

By default, feedthrough exclusions are not propagated through combinational circuits.

When used, all nets connected by combinational logic to nets specified as feedthrough exclusions are also excluded. For example, if nets A and B are excluded from feedthroughs due to the use of one or more of -

exclude_clock_feedthroughs, **-exclude_hfn_feedthroughs**, -

exclude_feedthroughs, and **-exclude_scan_chain_net_feedthroughs**, all nets connected to A and B via combinational logic are similarly excluded from feedthrough creation.

This option is not additive in the sense that **set_fp_pin_constraints** does not have any memory of the use of this option in previous invocations.

This option is ignored unless it is used together with the -

allow_feedthroughs on option.

`-exclude_clock_feedthroughs off | on`

Specifies whether feedthrough port creation is excluded on clock nets.

By default, this option is **on** (clock nets are excluded).

This option is ignored unless it is used together with the -

allow_feedthroughs on option.

`-exclude_scan_chain_net_feedthroughs off | on`

Specifies whether feedthrough port creation is excluded on scan chain nets.

By default, this option is **on** (scan chain nets are excluded).

This option is ignored unless it is used together with the -

allow_feedthroughs on option.

`-exclude_hfn_feedthroughs hfn_number`

Excludes feedthrough port creation on nets with a fanout greater than the specified number.

By default, feedthrough port creation is allowed on all nets (fanout limit of -1).

This option is ignored unless it is used together with the -

allow_feedthroughs on option.

-exclude_feedthroughs nets

Excludes feedthrough port creation on the specified nets.

The effect of this option is not cumulative. A subsequent call to

set_fp_pin_constraints that uses this option overrides the previous setting. For example, after the following successive calls to **set_fp_pin_constraints**, feedthrough exclusion applies only to net B:

```
set_fp_pin_constraints -allow_feedthroughs on \
```

```
  -exclude_feedthroughs [get_nets A]
```

```
set_fp_pin_constraints -allow_feedthroughs on \
```

```
  -exclude_feedthroughs [get_nets B]
```

Note that subsequent calls to **set_fp_pin_constraints** that do not use the **-exclude_feedthroughs** option do not change the existing setting.

To remove all nets previously specified with this option from feedthrough exclusion, use this option with an empty collection as argument, **-exclude_feedthroughs {}**.

This option is ignored unless it is used together with the **-allow_feedthroughs on** option.

-incremental off | on

Specifies whether the pin assignment or pin cutting is incremental. By default, this option is **off**.

In the top-down pin assignment flow, if the value is on, the **place_fp_pins** command retains the existing soft macro pins. If the value is off, it deletes and reassigns the soft macro pins.

In the block-level pin assignment flow, if the value is on, the **place_fp_pins** command retains the existing terminals. If the value is off, it deletes and reassigns the terminals.

In the pin cutting flow, if the value is on, the previously finalized routes generated by the **analyze_fp_routing -finalize_pins_feedthroughs plan_groups** command are retained. If the value is off, the previously finalized routings are abandoned.

Note: Specifying the **-incremental on** option nullifies the previous use of the **-nets** and **-exclude_nets** options.

-nets nets

Specifies the nets for which to perform pin assignment.

By default, pin assignment is performed on all nets.

This option and the **-exclude_nets** option are mutually exclusive; specify only one. This option nullifies the previous use of the **-exclude_nets** option.

This option is nullified by subsequent calls to the **set_fp_pin_constraints** command under the following conditions: when the **-exclude_nets** option is used, when the **-incremental on** option is used, or when an empty collection is used as the argument to this option, as shown in the following example:

```
set_fp_pin_constraints -nets []
```

The construct shown above is a convenient way to nullify the effect of this option without using the **remove_fp_pin_constraints** command.

Note: The argument set of this option is not incremental. Arguments supplied in an earlier use of this option are overwritten by the argument set specified in a subsequent use of this option. In the following example, pins are placed only for net B after the second call to the **set_fp_pin_constraints** command:

```
set_fp_pin_constraints -nets [get_flat_nets A]
```

```
set_fp_pin_constraints -nets [get_flat_nets B]
```

-exclude_nets nets

Specifies the nets for which pin assignment is not performed.

By default, pin assignment is performed on all nets. This option and the **-nets** option are mutually exclusive; specify only one. This option nullifies the previous use of the **-nets** option. This option is nullified by subsequent calls to the **set_fp_pin_constraints** command under the following conditions: when the **-nets** option is used, when the **-incremental on** option is used, or when an empty collection is used as the argument to this option, as shown in the following example:

```
set_fp_pin_constraints -exclude_nets []
```

The construct shown above is a convenient way to nullify the effect of this option without using the **remove_fp_pin_constraints** command.

Note: The argument set of this option is not incremental. Arguments supplied in an earlier use of this option are overwritten by the argument set specified in a subsequent use of this option. In the following example, pins are excluded only for net B after the second call to the **set_fp_pin_constraints** command:

```
set_fp_pin_constraints -exclude_nets [get_flat_nets A]
set_fp_pin_constraints -exclude_nets [get_flat_nets B]
```

-keep_buses_together off | on

Specifies whether the pins associated with bus bits are grouped together. By default, the bus bits are not kept together (**-keep_buses_together off**).

-bus_ordering lsb_to_msb | msb_to_lsb | scrambled | consistent_wirelengths

Specifies the method used for ordering the bus bits. This option is valid only if the **-keep_buses_together** option is enabled.

The following describes the supported bus ordering methods.

- **lsb_to_msb** (default)

Pins are ordered from the least significant bit to the most significant bit, and from left to right on a horizontal edge and bottom to top on a vertical edge. For example, a[0], a[1], a[2], ... a[31] for the 32-bit bus "a".

- **msb_to_lsb**

Pins are ordered from the most significant bit to the least significant bit, and from left to right on a horizontal edge and bottom to top on a vertical edge. For example, a[31], a[30],

- **scrambled**

Bits are first sorted in increasing order and then ordered using the skip count specified in the **-scramble_skip** option. For example, using a skip count of 1, the bits are ordered as

```
a[0], a[2], a[4], ... a[30], a[1], a[3],
a[5], ... a[31]
```

Using a skip count of 2, the bits are ordered as

```
a[0], a[3], a[6], ... a[1], a[4], a[7],
... a[2], a[2], a[5], a[8], ...
```

- **consistent_wirelengths**

Bits are ordered to equalize the wire lengths of the bus bits between soft macros. This is not a valid selection if the **-block_level** option is used for block-level pin assignment.

-reserved_channel_threshold *channel_size*

This feature is targeted at floorplans where there are only narrow channel spaces between the plan groups. In this case, feedthroughs should generally

be allowed, except for the special nets for which feedthroughs are excluded. Also note that this feature is implemented for Zroute based pin cutting only. Specifies the maximum size of a channel (space between two plan groups) that is to be reserved for special nets, such as clocks. The default is 0.0 (this feature is turned off). The units are microns. Negative numbers are not valid. The special nets (see **-reserved_channel_nets**) can route freely through any space between plan groups. If the space is less than or equal to the threshold, other nets are prevented from routing parallel to the direction of the channel, if at all possible. Pins in the channel area, or other constraints, might force other nets to enter the channel. In pin assignment, the effect of this is twofold: (1) Pins are discouraged on plan groups that face the channel unless they are perfectly aligned across the channel or are pins belonging to the special nets, such as clocks. (2) Nets that are not in the special group find other routing paths, creating feedthroughs instead of entering the channels. This creates feedthrough pins for these nets.

-reserved_channel_nets *nets*

Specifies the collection of nets that are permitted in the reserved channels. By default, this is the set of all nets that have been marked as clocks in the database. If you use this option, the collection must include all nets that you want in the reserved channels. Setting this option overrides the default or any previous selection. These nets are implicitly marked as "exclude feedthroughs" because the intent is to route them through the channels.

If you wish to exclude all nets from the reserved channels, you can do so by specifying an empty set of **reserved_channel_nets**. For example, to discourage routing in any channel space smaller than 20 microns:

```
prompt> set_fp_pin_constraints -reserved_channel_threshold 20 \
    -reserved_channel_nets [get_nets {}]
```

-scramble_skip *skip_number*

Specifies the skip count for scrambled bus bit ordering. This option is valid only if the **-bus_ordering** option is enabled using the scrambled method. The default skip count is 1.

-use_physical_constraints off | on

Specifies whether pin assignment should honor pin physical constraints set by the **set_pin_physical_constraints** command or read with the **read_io_constraints** command.

By default, pin assignment does not use physical constraints (**-use_physical_constraints off**).

If this option is set to **on**, preexisting physical constraints loaded with the **set_pin_physical_constraints** or **read_io_constraints** command are enforced.

-block_level

Applies constraints on block-level pins instead of soft macro pins.

blocks

Specifies the soft macros and plan groups to which the specified constraints apply.

By default, the specified constraints apply to all soft macros and plan groups unless **-block_level** is used, in which case the constraints apply to the top-level cell itself.

DESCRIPTION

This command allows you to set pin assignment constraints to be honored during pin cutting and pin assignment (see the man pages for **assign_fp_pins** and **commit_fp_plan_groups**). The pin assignment constraints are saved in the Milkyway database.

The **set_fp_pin_constraints** command is additive. The constraint values set in previous calls are retained unless they are explicitly overridden by subsequent calls.

Use the **remove_fp_pin_constraints** command to reset all constraint values to their defaults.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command allows feedthrough pins to be created on soft macros A, B, and C. All other pin constraints for A, B, and C are retained. If pin constraints were not previously defined for these soft macros, default values are used for all other pin constraints. If any of A, B, or C are plan groups, the pin constraints are migrated to their corresponding soft macros after you run the **commit_fp_plan_groups** command.

```
prompt> set_fp_pin_constraints -allow_feedthroughs on \  
[get_cells A B C]
```

The following command restricts pins to be on layers M2 through M5 on all soft macros and plan groups.

```
prompt> set_fp_pin_constraints -allowed_layers {M2 M3 M4 M5}
```

The following commands allow terminals to be created at the exact location and layers based on the preexisting physical constraints loaded with the **set_pin_physical_constraints** or **read_io_constraints** command.

```
prompt> set_fp_pin_constraints -block_level \  
-use_physical_constraints on \  
-hard_constraints {layer location}
```

SEE ALSO

```
place_fp_pins(2)  
commit_fp_plan_groups(2)  
read_io_constraints(2)  
set_pin_physical_constraints(2)  
analyze_fp_routing(2)
```

set_fp_placement_strategy

Sets parameters for controlling the **create_fp_placement** command.

SYNTAX

```
status set_fp_placement_strategy
[-default]
[-macro_orientation automatic | all | N]
[-auto_grouping none | user_only | low | high]
[-macro_setup_only on | off]
[-macros_on_edge on | off | auto]
[-snap_macros_to_user_grid on | off]
[-sliver_size distance]
[-fix_macros none | soft_macros_only | all]
[-congestion_effort low | medium | high]
[-IO_net_weight float]
[-plan_group_interface_net_weight float]
[-voltage_area_interface_net_weight float]
[-voltage_area_net_weight_LS_only on | off]
[-legalizer_effort low | high]
[-spread_spare_cells on | off]
[-virtual_IPO on | off]
[-pin_routing_aware on | off]
[-adjust_shapes on | off]
[-force_auto_detect_hierarchy on | off]
```

ARGUMENTS

-default

Set all the default parameter values.

-macro_orientation automatic | all | N

Controls how macro orientations will be determined. Automatic means that the tool will analyze the macro's reference library cell and determine a good orientation for each reference library cell based on the pins on the reference cell. All means macros can rotate to any legal orientation during placement. N means all macros will be in a LEF/DEF N (ie PDEF 0) orientation or a flipped version of N (that is, one of N, S, FN, FS). Default is automatic. Note that any reference library or user set orientation constraint will override this parameter.

-auto_grouping none | user_only | low | high

Controls the amount of array packing done. None means that no macro arrays will be generated. User_only means that only user-defined macro arrays will be generated. Low is the default value and means that automatic macro arrays will be generated from small macros only. High means automatic macro arrays will be generated from all macros.

-macro_setup_only on | off

Causes the placer to only create the macro arrays outside the chip and place macros according to their relative location constraints, and then stop without doing placement. You can then manually place the macro arrays if

desired. Default is off.

`-macros_on_edge on | off | auto`

Determines whether macros are placed along the edges of the chip or plan group. If set to on, all macros are placed along the edge; if set to auto, the tool will automatically decide which macros are to be placed along the edge. In a hierarchical design, it is recommended that you set this option to off until plan groups are shaped/placed inside the chip core. This is to avoid biasing the plan group locations to be on the edge. The default is auto.

`-snap_macros_to_user_grid on | off`

Snaps the lower-left corner of a macro's bounding box onto a user-defined grid. Use the **set_user_grid** command to create a user grid. For details, see the `set_user_grid` man page. The default is off.

`-sliver_size distance`

Controls the minimum channel size between macros which allows standard cells to be placed within the channel. The value is in microns. For example, if `sliver_size` is 10, then a channel of height 9 would not have any standard cells inside it. Default value is 0. Allowed range is ≥ 0 .

`-fix_macros none | soft_macros_only | all`

Controls whether macros should be temporarily fixed for **create_fp_placement**. None means all macros are allowed to move during placement. `Soft_macros_only` means all soft macros will not move during placement. All means all macros will not move during placement. Default is none. This control does not change the "fixed" attribute on macros.

`-congestion_effort low | medium | high`

Controls the congestion map used in the congestion driven placement. If the effort level is high, the global router will be called and the routing congestion map will be used. If the effort level is medium, the proto router will be called and the corresponding routing congestion map is used. The default value is low, which uses the placement congestion map.

`-IO_net_weight float`

Controls the net weight on nets connected to IOs. Default value is 1.0. Allowed range is between 0.0 and 10.0.

`-plan_group_interface_net_weight float`

Controls the net weight on interface nets of plan groups. Default value is 1.0. Allowed range is between 0.0 and 10.0.

`-voltage_area_interface_net_weight float`

Controls the net weight on interface nets of voltage areas. Default value is 1.0. Allowed range is between 0.0 and 10.0.

`-voltage_area_net_weight_LS_only on | off`

Controls the net weight set by the option `-voltage_area_interface_net_weight` to apply to nets connected to level shifter or isolation cells only. Default is off.

`-legalizer_effort low | high`

Controls the effort level of legalizer. Use low if the legalizer is running too long. The default is high.

`-spread_spare_cells on | off`
Spreads cells which have no net connectivity, across the design. Default is on.

`-virtual_IPO on | off`
Controls whether to use virtual in-place optimization during timing-driven placement. Default is off.

`-pin_routing_aware on | off`
Determines whether macro placement should consider reserving the area for routing resources to pins of a block or plan group. Default is off.

`-adjust_shapes on | off`
Determines whether an incremental adjustment of the plan group shapes will be applied during the congestion-driven placement. If `-adjust_shapes` on is set and all the plan groups are inside the core, incremental adjustment is applied. The default is off.

`-force_auto_detect_hierarchy on | off`
If your design contains existing plan groups either within the core area or outside the core area, and this parameter is set to on, the placement engine analyzes the logical hierarchy nodes that are left at the top level of the design to determine if they are candidates for hierarchy gravity. If the design does not contain plan groups, this option has no effect. The default is off.

DESCRIPTION

This command sets parameters for controlling the **create_fp_placement** and **legalize_fp_placement** design planning commands. These settings are not applicable to other placement commands or other parts of the flow. These settings are not persistent in the database.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLE

The following examples sets the sliver size to 10.5 microns.

```
prompt> set_fp_placement_strategy -sliver_size 10.5
INFO: Setting sliver_size = 10.50
```

SEE ALSO

```
report_fp_placement_strategy(2)
create_fp_placement(2)
set_user_grid(2)
```

set_fp_power_pad_constraints

Defines the power pad synthesis constraints.

SYNTAX

```
status set_fp_power_pad_constraints  
[-honor_existing_pads | -honor_even_space]  
[-target_pad_current current]  
[-maximum_number_of_pads number]  
[-save_file file_name]
```

Data Types

<i>current</i>	float
<i>number</i>	integer
<i>file_name</i>	string

ARGUMENTS

-honor_existing_pads

Honors all existing power and ground pads during pad synthesis. By default, the tool ignores existing power and ground pads during pad synthesis. However, virtual pads that you create are always honored during pad synthesis. This option and the **-honor_existing_space** option are mutually exclusive; you can specify only one.

-honor_even_space

Honors existing pads that have an even amount of space between them during pad synthesis. By default, the tool ignores existing power and ground pads during pad synthesis. However, virtual pads that you create are always honored during pad synthesis.

-target_pad_current *current*

Specifies the target pad current value (in mA). Power pad synthesis adds power pads and attempts to place them such that both the the maximum pad current (and voltage drop) and the maximum number of power pads are simultaneously satisfied. There is no default value.

-maximum_number_of_pads *number*

Specifies the maximum number of pads to be considered during pad synthesis. The default value is 40.

-save_file *file_name*

Specifies the name of the virtual pad file that is generated during pad synthesis. This virtual pad file can be read in by the **fp_create_virtual_pad -load_file** command. The default file name is default.vpad.

DESCRIPTION

Power pad synthesis determines the number of power pads and their locations based on the chip area and a target IR drop for boundary I/O designs. It can solve IR drop problems without modifying the existing design's power structure. Power pad synthesis can be performed simultaneously with power network synthesis.

Multicorner-Multimode Support

This command is not supported in a multi-scenario design flow.

EXAMPLES

The following example shows how to set power pad synthesis constraints that honor existing power and ground pads in the design, limit the pad current to be under 60mA and save the constraints to foo.vpad.

```
prompt> set_fp_power_pad_constraints\  
-honor_existing_pads -target_pad_current 60.0\  
-save_file foo.vpad
```

SEE ALSO

set_fp_rail_constraints

Defines power network synthesis (PNS) constraints, including layer constraints, power ring and strap constraints, and global constraints.

SYNTAX

```
status set_fp_rail_constraints
[-add_layer
  | -remove_layer
  | -remove_all_layers
  | -set_ring
  | -skip_ring
  | -set_global]
[-layer layer]
[-direction vertical | horizontal]
[-max_strap number]
[-min_strap number]
[-max_pitch distance]
[-min_pitch distance]
[-max_width distance]
[-min_width distance]
[-spacing distance | minimum | interleaving]
[-offset distance]
[-nets nets]
[-horizontal_ring_layer layer]
[-vertical_ring_layer layer]
[-ring_width distance]
[-ring_max_width distance]
[-ring_min_width distance]
[-ring_spacing distance]
[-ring_offset distance]
[-extend_strap core_ring | boundary | pad_ring]
[-keep_floating_segments]
[-no_stack_via]
[-no_same_width_sizing]
[-optimize_tracks]
[-keep_ring_outside_core]
[-no_routing_over_hard_macros]
[-no_routing_over_plan_groups]
[-no_routing_over_soft_macros]
[-ignore_blockages]
```

Data Types

<i>layer</i>	collection or list
<i>number</i>	integer
<i>distance</i>	float
<i>nets</i>	collection or list

ARGUMENTS

[-add_layer | -remove_layer | -remove_all_layers | -set_ring | -skip_ring | -

set_global

Specifies the type of constraint being set. These options are mutually exclusive; specify only one.

Specify **-add_layer** to add power strap layer constraints.

or

Specify **-remove_layer** to remove power strap layer constraints.

or

Specify -

remove_all_layers to remove all the power strap layer constraints.

or

Specify **-set_ring** to add power ring constraints.

or

Specify **-skip_ring** to disable power ring generation.

or

Specify **-set_global** to add global constraints.

-layer *layer*

Specifies the layer on which to create the power grid. This option is valid only when you also specify the **-add_layer** or **-remove_layer** option.

-direction *vertical* | *horizontal*

Specify the direction for the power and ground wires.

This option is valid only when you also specify the **-add_layer** option.

By default, the preferred routing direction of the specified layer is used.

-max_strap *number*

Controls the power strap density by setting the maximum number of straps. This option is valid only when you also specify the **-add_layer** option. The default is 128. This option and **-min_strap** option are mutually exclusive to **-max_pitch** and **-min_pitch** options.

-min_strap *number*

Controls the power strap density by setting the minimum number of straps. This option is valid only when you also specify the **-add_layer** option. The default is 16. This option and **-max_strap** option are mutually exclusive to **-max_pitch** and **-min_pitch** options.

-max_pitch *distance*

Specifies the maximum pitch (in microns).

This option is valid only when you also specify the **-add_layer** option.

This option and **-min_pitch** option are mutually exclusive to **-max_strap** and **-min_strap** options.

-min_pitch *distance*

Specifies the minimum pitch (in microns).

set_fp_rail_constraints

This option is valid only when you also specify the **-add_layer** option. This option and **-max_pitch** option are mutually exclusive to **-max_strap** and **-min_strap** options.

-max_width *distance*

Specifies the maximum width of the power wires (in microns). This option is valid only when you also specify the **-add_layer** option.

If you do not specify this option, the tool uses the maximum width that is defined in the technology file.

-min_width *distance*

Specifies the minimum width of the power wires (in microns). This option is valid only when you also specify the **-add_layer** option.

If you do not specify this option, the tool uses the minimum width that is defined in the technology file.

-spacing *distance | minimum | interleaving*

Constrains the spacing between the power and ground wires, so that signals may be able to route between the P/G networks. User can choose minimum spacing(**minimum**) or interleaving (**interleaving**), or explicitly specify the *distance* in the unit of microns. The default is **minimum**. This option is valid only when you also specify the **-add_layer** option.

-offset *distance*

Specifies the offset from the I/O pads to the nearest power straps (in microns). This option is valid only when you also specify the **-add_layer** option.

-nets *nets*

Specifies the nets that make up the core rings. To generate multiple core rings, specify multiple nets. The first net is be the outermost ring. This option is valid only when you also specify the **-set_ring** option.

-horizontal_ring_layer *layer*

Specifies the horizontal metal layers for the power ring. This option is valid only when you also specify the **-set_ring** option.

The default horizontal layer is the highest horizontal layer in the database.

-vertical_ring_layer *layer*

Specifies the vertical metal layers for the power ring. This option is valid only when you also specify the **-set_ring** option.

The default vertical layer is the highest vertical layer in the database.

-ring_width *distance*

Specifies the width of the power rings (in microns).
This option is valid only when you also specify the **-set_ring** option.

If this option is not set, by default the tool will automatically determine the ring width to meet the IR drop requirements.

You cannot specify this option together with the **-ring_max_width** or **-ring_min_width** option.

-ring_max_width *distance*

Specifies the maximum allowable core ring width (in microns).
This option is valid only when you also specify the **-set_ring** option.

If you specify this option, you cannot specify the **-ring_width** option.

-ring_min_width *distance*

Specifies the minimum allowable core ring width (in microns).
This option is valid only when you also specify the **-set_ring** option.

If you specify this option, you cannot specify the **-ring_width** option.

-ring_spacing *distance*

Specifies the spacing between the core rings (in microns).
This option is valid only when you also specify the **-set_ring** option.

-ring_offset *distance*

Specifies the offset from the I/O pads to the power rings (in microns).
This option is valid only when you also specify the **-set_ring** option.

-extend_strap *core_ring | boundary | pad_ring*

If you choose to generate a power plan without a core ring, you must tell PNS how to connect (extend) the generated horizontal and vertical power straps.

This option is valid only when you also specify the **-set_ring** or **-skip_ring** option.

There are three options:

core_ring (default)

PNS extends the power straps to the existing core ring.

boundary

PNS extends the power straps to the top-level cell boundary and creates power pins.

pad_ring

PNS extends the power straps to an existing pad ring.

-keep_floating_segments

Instructs PNS to keep dangling wires cut by hard macros or blockages. This option is valid only when you also specify the **-set_global** option.

By default, dangling wires cut are removed.

-no_stack_via

Prevents PNS from using stack via (vias are dropped only on adjacent layers). This option is valid only when you also specify the **-set_global** option.

By default, PNS uses stack vias.

-no_same_width_sizing

Allows PNS to create wires that have different widths for power and ground nets. This option is valid only when you also specify the **-set_global** option.

By default, PNS creates wires with the same width for power and ground nets.

-optimize_tracks

PNS sizes wires so that it leaves enough space for the power and ground wire's adjacent tracks to be used for signal routes.

By default, no sizing is performed on P/G wires.

This option is valid only when you also specify the **-set_global** option.

-keep_ring_outside_core

PNS creates the core ring outside of the core area. This option is valid only when you also specify the **-set_global** option.

-no_routing_over_hard_macros

Prevents PNS from routing over hard macros. This option is valid only when you also specify the **-set_global** option.

-no_routing_over_plan_groups

Prevents PNS from routing over plan groups, despite the absence of blockages. This option is valid only when you also specify the **-set_global** option.

-no_routing_over_soft_macros

Prevents PNS from routing over soft macros, despite the absence of blockages. This option is valid only when you also specify the **-set_global** option.

-ignore_blockages

Allows PNS to create power wires despite hard macro blockages. This option is valid only when you also specify the **-set_global** option.

By default, PNS honors hard macro blockages to create a DRC-clean power plan.

DESCRIPTION

This command sets the power network synthesis (PNS) constraints for power planning, including power strap layer constraints, power ring constraints, and global constraints. Multiple core rings and sandwich core rings can also be generated.

Multicorner-Multimode Support

This command is not supported in a multi-scenario design flow.

EXAMPLES

The following example shows how to set the power network synthesis layer constraints. The horizontal layer is METAL5 with a maximum of 20 straps and a minimum of 5 straps; the vertical layer is METAL6 with a maximum of 40 straps and a minimum of 8 straps.

```
prompt> set_fp_rail_constraints \  
-add_layer -layer METAL5 \  
-direction horizontal \  
-max_strap 20 \  
-min_strap 5 \  

```

```
prompt> set_fp_rail_constraints \  
-add_layer -layer METAL6 \  
-direction vertical \  
-max_strap 40 \  
-min_strap 8 \  

```

SEE ALSO

```
report_fp_rail_constraints(2)  
synthesize_fp_rail(2)
```

set_fp_rail_region_constraints

Defines the region where power plan synthesis creates a rectilinear power mesh.

SYNTAX

```
status_value set_fp_rail_region_constraints  
[-voltage_area voltage_area  
 | -polygon  
-load_file file_name  
 | -remove]  
[-save_file file_name]
```

Data Types

<i>voltage_area</i>	collection or list of one object
<i>polygon_area</i>	list
<i>file_name</i>	string

ARGUMENTS

-voltage_area *voltage_area*

Defines the PNS region based on the specified voltage area. You can specify only a single voltage area.

By default, the command creates a region based on the DEFAULT_VA voltage area. The DEFAULT_VA voltage area includes the whole core area excluding existing voltage areas.

This option and the **-polygon**, **-load_file**, and **-remove** options are mutually exclusive; you must specify only one.

-polygon {*polygon_area*}

Defines a rectilinear PNS region based on the user specified polygon area. A polygon is identified by specifying the coordinates of all its points. The format is {{x1 y1} {x2 y2} ...}.

This option and the **-voltage_area**, **-load_file**, and **-remove** options are mutually exclusive; you must specify only one.

-load_file *file_name*

Defines a rail region by loading the specified region definition file.

This option and the **-voltage_area**, **-polygon**, and **-remove** options are mutually exclusive; you must specify only one.

-remove

Deletes the currently defined rail region. Note that after commit, all defined rail regions are automatically deleted.

This option and the **-voltage_area**, **-polygon**, and **-load_file** options are mutually exclusive; you must specify only one.

-save_file *file_name*

Saves the defined rail region to the specified file.

This option is optional. The default file name is default.region.

DESCRIPTION

Defines the region where power plan synthesis creates a rectilinear power mesh.

Multicorner-Multimode Support

This command is not supported in a multi-scenario design flow.

EXAMPLES

The following example defines a region for power network synthesis with the coordinates {1200.00 952.00 1200.00 650.00 1400.00 650.00 1400.00 952.00} and saves the definition to foo.region.

```
prompt> set_fp_rail_region_constraints \  
        -polygon {1200 952 1200 650 1400 650 1400 952} \  
        -save_file foo.region
```

SEE ALSO

create_voltage_area(2)

set_fp_rail_strategy

Sets the strategy for power network synthesis (PNS) and power network analysis (PNA). These settings are not persistent in the Milkyway database.

SYNTAX

```
status set_fp_rail_strategy
[-reset]
[-use_lm_view true | false]
[-use_tluplus true | false]
[-pna_ultra_solver true | false]
[-virtual_pad_wire_width virt_wire_width]
[-define_pad_connection connection_config_file_name]
[-pad_resistance_file pad_resistance_file_name]
[-pns_skip_ir true | false]
[-honor_macro_strap_config configuration_file]
[-pns_hor_relative_offset {h_layer h_rel_offset}]
[-pns_ver_relative_offset {v_layer v_rel_offset}]
[-pns_ignore_via_cut_to_edge true | false]
[-pns_ignore_soft_macro_blockage true | false]
[-pns_clip_top_boundaries true | false]
[-cut_plangroup_edge_layers layers]
[-pna_via_cut_row_column number]
[-honor_macro_route_constraints routing_constraints_file_name]
[-align_strap_with_top_pin true | false]
[-align_strap_with_mtcmos_cells alignment_constraints_file_name]
[-align_strap_with_bump_cells alignment_constraints_file_name]
[-align_strap_with_m1_rail true | false]
[-put_strap_in_std_cell_row true | false]
[-commit_fast true | false]
[-create_hierarchical_pns_script true | false]
```

Data Types

<i>virt_wire_width</i>	float
<i>connection_config_file_name</i>	string
<i>pad_resistance_file_name</i>	string
<i>configuration_file</i>	string
<i>h_layer</i>	list
<i>h_rel_offset</i>	float
<i>v_layer</i>	list
<i>v_rel_offset</i>	float
<i>number</i>	integer
<i>routing_constraints_file_name</i>	string
<i>alignment_constraints_file_name</i>	string

ARGUMENTS

-reset
Reset all the PNS/PNA strategy.

`-use_lm_view true | false`
 Uses the LM view for power calculations. The default value is **false**.

`-use_tluplus true | false`
 Specifies for power network synthesis (PNS) and power network analysis (PNA) to use the TLU+ model. PNA and PNS use the TLU+ model when either of the following conditions are met:

- This option is set to **true**. Resistance values in technology files are not defined. The default value is **false**.

`-pna_ultra_solver true | false`
 Enables PNA to use the ultra-solver, which uses less memory but more CPU time. The default value is **false**.

`-virtual_pad_wire_width virt_wire_width`
 Defines the width of the virtual wires used to connect the virtual pads and their neighboring wires. By default, the PNS/PNA virtual pad uses the closest wire width for a virtual connection.

`-define_pad_connection connection_config_file_name`
 Enables PNA to make virtual connections from power and ground pads to power and ground wires based on a pre-defined connection configuration specified by the *connection_config_file_name*. The file includes pad master/instance name, connected pin layers, and routing layers. The format of the configuration file is as follows:

```

Instance|Master <name>

Route Pins layer <layer_number> | all

Primary Routing Layer Preferred low | high
Or
Primary Routing Layer Pin
Or
Primary Routing Layer Specified Horizontal layer_number
Primary Routing Layer Specified Vertical layer_number
I. Pad Instance/Master name: Specifies the instance or master name of pads.
II. Connected pin layers: Connects only pins on All layers or the specified layer.
III. Primary Routing Layer: Specifies how to pick the layer on which to route.
Valid keywords are: preferred | pin | specified.
Use preferred to route on preferred metal layers. You can further refine routing by using the low or high keyword, where high routes on the next available higher preferred layer. If a higher preferred layer is not available, use low to route on an available lower preferred layer. Use pin to route on the same layer on which the pins are located. Use specified to route on layers. Use the keywords horizontal layer_number or vertical layer_number to define horizontal and vertical routing layers. See an example of using the -define_pad_connection option in the EXAMPLE section.
```

`-pad_resistance_file pad_resistance_file_name`
 Specifies the file from which to read pad resistance. This file is used to add extra resistance for power and ground pads.

`-pns_skip_ir true | false`

Causes PNS to skip IR drop simulation by specifying **true**. By default, PNS always performs IR drop simulation before committing the results. When this option is set to **true**, the number and width of the specified straps for all defined layers are fixed (maximum is equal to minimum). The default value is **false**.

`-honor_macro_strap_config configuration_file`

Enables PNS to accept user-defined power and ground configurations with layer, direction, width, pitch, and offset on the specified hard macros or rectangle regions. PNS synthesizes the remaining area based on your constraints.

The format of the configuration file is as follows:

`<region_type> <name or polygon>`

`Layer <layer_name> Direction <direction> Width <width>`

`Pitch <pitch> Offset <offset>`

User one of the keywords: **Instance** or **Master** or **Region** to specify the region type. For Instance or Master, specify its name next; for user-defined region, specify the region polygon. This must be specified in a separate line from below. *layer_name* can be either layer name or mask name; *direction* can be one of the keywords: **Horizontal** or **Vertical**; *width*, *pitch* and *offset* are floating numbers. User can specify multiple layers for one region. See the examples in the EXAMPLE section for more detailed usage of this strategy.

`-pns_hor_relative_offset {h_layer h_rel_offset}`

Specifies the relative offsets from the top-most horizontal layer to the specified layer so that PNS synthesizes nonoverlapping straps with the same directional layers. This option is used when there are more than two layers of power and ground straps.

This option has two arguments, so you must enclose them in curly brackets {}. The first argument, *h_layer*, is the layer name. The second argument, *h_rel_offset*, is the offset (in microns) from the top-most layer to the specified layer.

If the offset is a positive value, the straps for the specified layer are above the corresponding top-layer straps. If the offset is a negative value, the straps for the specified layer are below the corresponding top-layer straps. You can reset this strategy by specifying an empty string. By default, PNS generates overlapping straps on different layers with the same direction.

`-pns_ver_relative_offset {v_layer v_rel_offset}`

Specifies the relative offsets from the top-most vertical layer to the specified layer so that PNS synthesizes nonoverlapping straps with the same directional layers. This option is used when there are more than two layers of power and ground straps.

This option has two arguments, so you must enclose them in curly brackets {}. The first argument, *v_layer*, is the layer name. The second argument, *v_rel_offset*, is the offset (in microns) from the top-most layer to the specified layer.

If the offset is a positive value, the straps for the specified layer are to the right of the corresponding top-layer straps. If the offset is a negative value, the straps for the specified layer are to the left of the corresponding top-layer straps. You can reset this strategy by specifying an empty string. By default, PNS generates overlapping straps on different layers with the same direction.

`-pns_ignore_via_cut_to_edge true | false`
 Causes the synthesized power and ground wires to ignore the via's cut to edge spacing when you set this option to the **true** value. This strategy enables PNS to use smaller spacing values, (defined in technology file), but it might result in a design rule checking (DRC) violation. The default is **false**.

`-pns_ignore_soft_macro_blockage true | false`
 Ignores soft macro blockages when you set this option to the **true** value. The default is **false**.

`-pns_clip_top_boundaries true | false`
 Causes PNS to turn on the "clip at top-cell boundaries" option in the preroute engine when committing power networks when you set this option to the **true** value. The default is **false**.

`-cut_plangroup_edge_layers layers`
 Specifies the layer on which synthesized straps are removed if they overlap with plangroup edges. Users can use the keyword "all" to cut synthesized straps that overlap plangroup edges on all layers. By default, this strategy is off.

`-pna_via_cut_row_column number`
 Breaks a large square or a rectangular via array into smaller square via arrays based on the value specified by *number*.

`-honor_macro_route_constraints routing_constraints_file_name`
 Specifies routing constraints for macros in an input file that identifies the constraints. The *routing_constraints_file_name* input file should define the macro type (master or instance), macro name (master name or instance name), net name (the net to block), and layer name (the layer to block, or use keyword **all** to block routing on all metal layers). By default, the command uses no routing constraints.

`-align_strap_with_top_pin true | false`
 If this strategy is specified, PNS will align power and ground straps with power and ground pins on block boundary. The alignment strap layer will be the layer of the pin and the strap width will be the pin width. In addition, PNS will also generate other non-aligned straps to satisfy the target IR drop. However, if user does not want PNS to generate other straps except for the aligned straps, user can set the Tcl variable **pns_pin_alignment_strap_only** to *true*.
 This strategy is especially useful in block-level PNS. By default this strategy is **false**.

`-align_strap_with_mtcmos_cells alignment_constraints_file_name`
 Causes PNS to align the power straps with MTCMOS cells and connect straps to the corresponding pins in the MTCMOS cells based on the constraint file specified by the *alignment_constraints_file_name* argument. This strategy is used to configure primary power straps in MVDD PNS for power-gated voltage areas.
 You can specify the following information in the configuration file, one net per line:
net_name direction width layer VA_name
 where

- *net_name* is the name of the primary power net to align.
- *direction* is the alignment direction, the value of which can be one of **horizontal**, **vertical**, or **ring**.
- *width* is the width of aligned primary power straps
- *layer* is the layer name of aligned primary power straps (in the ring MTCMOS placement style, *layer* should include both the horizontal and vertical ring layer names).
- *VA_name* specifies which voltage area this strategy is applied to.

The following example shows how to specify a configuration file. In this example, there are two voltage areas: VA1 and VA2. Both voltage areas are power-down voltage areas. VA1 uses the VDD1 primary power net and VA2 uses the VDD2 primary power net. VA1 uses array-style MTCMOS placement and VA2 uses ring-style MTCMOS placement.

The content of file `example.align` should look like the following:

```
VDD1 vertical 1.5 METAL6          VA1
VDD2 ring      1.0 METAL5,METAL6  VA2
```

Note that, for the *width* and *layer* fields, if you want power network synthesis to use the default value, you can use the **default** keyword to specify this, as shown in the following example:

```
VDD1 vertical default METAL6 VA1
```

You can reset this strategy by using an empty file name. By default, this strategy is set to NULL.

`-align_strap_with_bump_cells alignment_constraints_file_name`

If this strategy is specified, PNS will align power/ground straps with bump cells in the design based on the specification in the *alignment_constraints_file_name*.

The format of the constraint file is as follows:

```
net_name direction width layer
```

where

- *net_name* is the name of the power/ground net to align.
- *direction* is the alignment direction. The value is one of the string "horizontal" or "vertical".
- *width* is the width of aligned power/ground straps.
- *layer* is the layer name of aligned power/ground straps.

User can specify multiple lines in the file. The following example shows how to specify a configuration file. In this example, VDD and VSS are aligned with bump cells vertically on METAL6 with strap width 1.5 microns.

```
VDD vertical 1.5 METAL6
VSS vertical 1.5 METAL6
```

Note that, for the `<width>` and `<layer>` field, if user want PNS to use the default value, user could use keyword "default" to specify as follows:

```
VDD vertical default METAL6
```

You can reset this strategy by using an empty file name. By default this strategy is set to NULL.

`-align_strap_with_m1_rail true | false`

This option enables PNS to align synthesized power and ground straps in the lowest horizontal layer with the horizontal power and ground rails of standard cells to avoid horizontal power (or ground) straps overlapping with horizontal ground (or power) standard cell rails when you set this option to **true**. The default is **false**. Note that the fixed strap pitch, fixed strap offset and fixed power ground spacing might not be honored due to the alignment. Some straps might be discarded due to the limited number of

available standard cell rows. You can select "put core ring outside of core area" to avoid standard cell rails blocked by the core ring. This option is mutually exclusive with **put_strap_in_std_cell_row** option, i.e. only one of them could be set to true.

`-put_strap_in_std_cell_row true | false`

This option enables PNS to put synthesized power and ground straps in the lowest horizontal layer between standard cell power and ground rails inside standard cell rows when you set this option to **true**. The default is **false**. Therefore, straps may not be overlapped with standard cell rails even if they may be in the same logic net. Note that the fixed strap pitch, fixed strap offset and fixed power ground spacing might not be honored to avoid rail blockage. Some straps might be discarded due to the limited number of available standard cell rows. At most one strap may be laid within one standard cell row. You can select "put core ring outside of core area" to avoid standard cell rails blocked by the core ring. This option is mutually exclusive with **align_strap_with_m1_rail** option, i.e. only one of them could be set to true.

`-commit_fast true | false`

This strategy enables PNS to call the **create_power_straps** preroute command once for each layer when committing the synthesized power plan. The default is **false**, meaning that PNS calls **create_power_straps** once for each segment during the commit process.

`-create_hierarchical_pns_script true | false`

This strategy enables power network synthesis at the top-level to create power network synthesis replay scripts for each plan group in the design such that after the plangroups are committed to soft macros, you can resynthesize power networks in each soft macro with the same power budget allocated from top-level and with a similar power network as pushed down from the top-level. The script considers voltage drops at the PG pins on the soft macro boundary during IR drop simulation in the block-level power network synthesis.

If this strategy is set to true, a subdirectory with the name *plangroup_name_PNS_script* is created for each plan group in the *pna_output* directory. In each *plangroup_name_PNS_script* subdirectory two files are generated:

plangroup_name.vin contains the voltage information for each pin on the plan group boundary.

plangroup_name_replay.tcl is the replay tcl file that can resynthesize the power network in the soft macro. Note that the *plangroup_name.vin* is called from this Tcl file.

By default, this strategy is set to false.

DESCRIPTION

This command sets parameters for power network synthesis (PNS) and power network analysis (PNA). These settings are not persistent in Milkyway database.

Multicorner-Multimode Support

This command is not supported in a multi-scenario design flow.

EXAMPLES

The following examples show how to define a pad connection file.

```
prompt> set_fp_rail_strategy\  
-define_pad_connection "connection_file"
```

Example I: For pads with master name *KMVDDIOGS.FRAME*, pick pins with layer number 1, and use the same layers with pin to make connection.

<connection_file>:

```
Master KVVDDIOGS.FRAME  
Route Pins Layer 1  
Primary Routing Layer Pin
```

Example II: For pads with master name *KMVDDIOGS.FRAME*, pick pins with all layers, and use layer 1 as horizontal routing, and layer 11 vertical routing.

<connection_file>:

```
Master KVVDDIOGS.FRAME  
Route Pins All  
Primary Routing Layer Specified Horizontal 1  
Primary Routing Layer Specified Vertical 11
```

Example III: For pads with master name *KMVDDIOGS*, pick pins with all layers, and use the same layers with pin to make connection. The connection rule is the same as the one in *preroute_instances*

<connection_file>:

```
Master KVVDDIOGS  
Route Pins All  
Primary Routing Layer Pin
```

The following example specifies the offset of 15 microns from the top edge of *metal3* power nets to the top edge of *metal5* power nets. Because the offset is a positive number, the *metal3* straps are above the corresponding *metal5* straps.

```
prompt> set_fp_rail_strategy\  
-pns_hor_relative_offset "metal3 15"
```

The following example shows how to define customized P/G routing regions in PNS. In this example, P/G network on all the macros with master name *RAM* will be created vertically on *metal6* with width 4 microns, pitch 80 microns and offset 10 microns, horizontally on *metal5* with width 4 microns, pitch 60 and offset 10 microns.

```
prompt> set_fp_rail_strategy\  
-honor_macro_strap_config "configuration_file1"
```

<configuration_file1>:

```
Master RAM  
Layer metal6 Direction Vertical Width 4 Pitch 80 Offset 10  
Layer metal5 Direction Horizontal Width 5 Pitch 60 Offset 10
```

The following example shows how to define customized P/G routing regions in PNS. In this example, P/G network in the defined region (7320 100 9110 1410) will be created

vertically on *M8* with width 8.2 microns, pitch 60.5 microns and offset 6.2 microns, horizontally on *M7* with width 6 microns, pitch 60 and offset 6 microns.

```
prompt> set_fp_rail_strategy\  
-honor_macro_strap_config "configuration_file2"  
<configuration_file2>:  
Region 7320 100 9110 1410  
Layer M8 Direction Vertical Width 8.2 Pitch 60.5 Offset 6.2  
Layer M7 Direction Horizontal Width 6 Pitch 60 Offset 6
```

The following example shows how to define a macro routing constraint in a file. In this example, *VDD* net will not route over all instances of cell master *BLENDER_1.CEL* on *METAL3*; *VDD* net will not route over all instances of cell master *BLENDER_2.CEL* on *METAL3* and *METAL4*; *VSS* net will not route over cell instance *I_ORCA_TOP/I_BLENDER_3* on all the metal layers.

```
prompt> set_fp_rail_strategy\  
-honor_macro_route_constraints "route_constr_file"  
<route_constr_file>:  
master BLENDER_1.CEL VDD METAL3  
master BLENDER_2.CEL VDD {METAL3,METAL4}  
instance I_ORCA_TOP/I_BLENDER_3 VSS all
```

The following example shows how to specify MTCMOS alignment constraint in a file. In this example, *VDD* strap of width 2 microns on *METAL8* will be vertically aligned with all MTCMOS cells in voltage area *MULTIPLIER*, and *VSS* straps of width 1.5 microns on *default* metal layers will be horizontally aligned with instances of cell master *FOOTER.CEL* in all voltage areas. Here the default layer is the topmost metal layer on the specified direction.

```
prompt> set_fp_rail_strategy\  
-align_strap_with_mtcmos_cells "align_constr_file"  
<align_constr_file>:  
VDD vertical 2 METAL8 MULTIPLIER  
VSS horizontal 1.5 default all FOOTER.CEL
```

The following example shows how to specify bump cell alignment constraint in a file. In this example, *VDD* strap of width 2.0 microns on *METAL8* will be vertically aligned with *VDD* bump cells; *VSS* straps of width 2.0 microns on *default* metal layers will be horizontally aligned with *VSS* bump cells. The default layer is the topmost metal layer of the specified direction.

```
prompt> set_fp_rail_strategy\  
-align_strap_with_bump_cells "align_constr_file"  
<align_constr_file>:  
VDD vertical 2.0 METAL8  
VSS horizontal 2.0 default
```

SEE ALSO

report_fp_rail_strategy(2)

set_fp_rail_voltage_area_constraints

Defines power network synthesis (PNS) constraints for the specified voltage area. This command specifies four groups of PNS constraints: synthesis constraints, layer constraints, ring constraints, and global constraints. Note that these constraints cannot be specified together in command; they must be specified in separate commands.

SYNTAX

```
status set_fp_rail_voltage_area_constraints
-voltage_area voltage_area
[[-nets pg_nets]
-voltage_supply supply_voltage
[-power_budget power]
[-target_voltage_drop voltage]
[-power_switch power_switch_or_lib_cell]] |
[-layer pg_layer
-direction vertical | horizontal
[[-max_strap max_strap_number]
[-min_strap min_strap_number]] |
[[-max_pitch max_pitch_distance]
[-min_pitch min_pitch_distance]]
[-max_width max_width_distance]
[-min_width min_width_distance]
[-spacing minimum | interleaving | distance]
[-offset offset_distance]
[-mtcmos_net_type permanent | virtual]] |
[[-ring_nets ring_nets | -skip_ring]
[-horizontal_ring_layer h_ring_layer]
[-vertical_ring_layer v_ring_layer]
[-ring_width pg_ring_width]
[-ring_max_width max_ring_width]
[-ring_min_width min_ring_width]
[-ring_spacing distance_between_rings]
[-ring_offset distance_to_boundary]
[-extend_strap voltage_area_ring | core_ring | boundary]
[-num_extend_straps number_extended]
[-extend_strap_direction directions_to_extend]] |
[-global
[-keep_floating_segments]
[-no_stack_via]
[-no_same_width_sizing]
[-optimize_tracks]
[-no_routing_over_hard_macros]
[-no_routing_over_plan_groups]
[-no_routing_over_soft_macros]
[-ignore_blockages]
[-allow_routing_over_voltage_area]]]
```

Data Types

<i>voltage_area</i>	collection or list of one item
<i>pg_nets</i>	collection or list

<i>supply_voltage</i>	float
<i>power</i>	float
<i>voltage</i>	float
<i>pg_layer</i>	list
<i>max_strap_number</i>	integer
<i>max_pitch_distance</i>	float
<i>min_pitch_distance</i>	float
<i>max_width_distance</i>	float
<i>min_width_distance</i>	float
<i>offset_distance</i>	float
<i>ring_nets</i>	collection or list
<i>h_ring_layer</i>	list
<i>v_ring_layer</i>	list
<i>pg_ring_width</i>	float
<i>max_ring_width</i>	float
<i>min_ring_width</i>	float
<i>distance_between_rings</i>	float
<i>distance_to_boundary</i>	float
<i>number_extended</i>	integer

ARGUMENTS

-voltage_area *voltage_area*

Specifies the voltage area to which the constraints apply.
This is a required option.

-nets *pg_nets*

Specifies the power and ground (PG) nets to be synthesized in the specified voltage areas. For power-down voltage areas, specify the permanent and virtual PG net as "VDD_Permanent+VDD_Virtual". PNS synthesizes both nets together connected through power switch cells. If UPF is defined, PNS uses the primary power and ground net from the UPF.
This option belongs to the synthesis constraints group.

-voltage_supply *supply_voltage*

Specifies the supply voltage in the specified voltage areas. The *supply_voltage* value is in volts.
This option belongs to the synthesis constraints group and is required when specifying synthesis constraints.

-power_budget *power*

Specifies the power budget in the specified voltage areas. The *power* value is in milliwatts.
By default, power budgets are automatically assigned to each voltage area based on the total power budget specified by using the **synthesize_fp_rail** command.
This option belongs to the synthesis constraints group.

-target_voltage_drop *voltage*

Specifies the target voltage (IR) drop value in the specified voltage areas. The *voltage* value is in millivolts.
By default, the target IR drop value is 10 percent of the supply voltage in that voltage area.
This option belongs to the synthesis constraints group.

set_fp_rail_voltage_area_constraints

-power_switch *power_switch_or_lib_cell*
 Specifies the library cell to be used as the power switch in the specified voltage areas. In UPF mode, use the power switch cell name defined in the UPF. In non-UPF mode, use the library reference cell name. Use this option only for power switch synthesis. For more information, see the man page for the **synthesize_fp_rail** command. This option belongs to the synthesis constraints group.

-layer *pg_layer*
 Specifies the metal layers for the power and ground straps in the specified voltage areas. This option belongs to the layer constraints group and is required when specifying layer constraints.

-direction *vertical | horizontal*
 Specifies the direction of power and ground straps on the specified layers in the specified voltage areas. This option belongs to the layer constraints group and is required when specifying layer constraints.

-max_strap *max_strap_number*
 Specifies the maximum number of power and ground straps in the specified voltage areas. By default, the maximum number of straps is 128. You cannot use this option together with the **-max_pitch** or **-min_pitch** option. This option belongs to the layer constraints group.

-min_strap *min_strap_number*
 Specifies the minimum number of power and ground straps in the specified voltage areas. By default, the minimum number of straps is 16. You cannot use this option together with the **-max_pitch** or **-min_pitch** option. This option belongs to the layer constraints group.

-max_pitch *max_pitch_distance*
 Specifies the maximum pitch of the power and ground straps in the specified voltage areas. The *max_pitch_distance* value is in microns. You cannot use this option together with the **-max_strap** or **-min_strap** options. This option belongs to the layer constraints group.

-min_pitch *min_pitch_distance*
 Specifies the minimum pitch of the power and ground straps in the specified voltage areas. The *min_pitch_distance* value is in microns. You cannot use this option together with the **-max_strap** or **-min_strap** option. This option belongs to the layer constraints group.

-max_width *max_width_distance*
 Specifies the maximum width of the power and ground straps in the specified voltage areas. The *max_width_distance* value is in microns. By default, PNS uses the maximum width defined in the technology file. This option belongs to the layer constraints group.

-min_width *min_width_distance*
 Specifies the minimum width of the power and ground straps in the specified

voltage areas. The *min_width_distance* value is in microns.
By default, PNS uses the minimum width defined in the technology file.
This option belongs to the layer constraints group.

-spacing *minimum | interleaving | distance*
Specifies the spacing between power and ground straps in the specified voltage areas. Use **minimum** for minimum spacing, **interleaving** for interleaving, or *distance* for explicitly specified spacing in microns.
By default, PNS uses minimum spacing.
This option belongs to the layer constraints group.

-offset *offset_distance*
Specifies the distance from the voltage area boundary to the left-most and bottom-most power strap. The *offset_distance* value is in microns.
By default, PNS calculates the offset distance based on the number of straps, the strap pitch, and the strap width.
This option belongs to the layer constraints group.

-mtcmos_net_type *permanent | virtual*
Specifies the MTCMOS net type on the layer in power switch array synthesis. In power switch array synthesis, only one power net, either a permanent power net or a virtual power net, can be synthesized on one layer. The ground net, however, can be synthesized on any layer. This option is used only for power switch array synthesis (***synthesize_fp_rail -synthesize_power_switch_array***).
This option belongs to the layer constraints group.

-ring_nets *ring_nets*
Specifies the power and ground ring nets to be generated inside the specified voltage areas. To generate the core ring for each voltage area, you must first specify the ring nets in *DEFAULT_VA*, as shown in the EXAMPLES section.
By default, the *ring_nets* are the same as the nets specified in the synthesis constraints for the voltage area.
This option belongs to the ring constraints group.

-skip_ring
Does not generate a power and ground ring in the voltage area.
The **-skip_ring** and **-ring_nets** options are mutually exclusive; use only one.
This option belongs to the ring constraints group.

-horizontal_ring_layer *h_ring_layer*
Specifies the horizontal metal layer for the power and ground ring in the voltage area. By default, PNS uses the highest horizontal metal layer.
This option belongs to the ring constraints group and must be specified with the **-ring_nets** option.

-vertical_ring_layer *v_ring_layer*
Specifies the vertical metal layer for the power and ground ring in the voltage area. By default, PNS uses the highest vertical metal layer.
This option belongs to the ring constraints group.

-ring_width *pg_ring_width*
Specifies the width of the power and ground ring in the voltage area. The *pg_ring_width* value is in microns.
By default, the ring width is twice the strap width in the voltage area, except for *DEFAULT_VA*. In *DEFAULT_VA*, the default ring width is ten times the

minimum width defined in the technology file for the specified ring metal layer.

The **-ring_width**, **-ring_max_width**, and **-ring_min_width** options are mutually exclusive; you can use only one.

This option belongs to the ring constraints group and must be specified with the **-ring_nets** option.

-ring_max_width *max_ring_width*

Specifies the maximum allowable ring width in the voltage area. The *max_ring_width* value is in microns.

The **-ring_width**, **-ring_max_width**, and **-ring_min_width** options are mutually exclusive; you can use only one.

This option belongs to the ring constraints group and must be specified with the **-ring_nets** option.

-ring_min_width *min_ring_width*

Specifies the minimum allowable ring width in the voltage area. The *min_ring_width* value is in microns.

The **-ring_width**, **-ring_max_width**, and **-ring_min_width** options are mutually exclusive; you can use only one.

This option belongs to the ring constraints group and must be specified with the **-ring_nets** option.

-ring_spacing *distance_between_rings*

Specifies the distance between the power and ground rings in the voltage area. The *distance_between_rings* value is in microns.

By default, PNS uses the minimum spacing from the technology file.

This option belongs to the ring constraints group and must be specified with the **-ring_nets** option.

-ring_offset *distance_to_boundary*

Specifies the distance from the voltage area boundary to the closest ring net. The *distance_to_boundary* value is in microns.

By default, the ring offset is 0.

This option belongs to the ring constraints group and must be specified with the **-ring_nets** option.

-extend_strap *voltage_area_ring | core_ring | boundary*

Specifies where to extend power and ground nets in the voltage area. This option takes one of the following keywords:

* **voltage_area_ring**

Extends the power and ground straps to the ring inside the voltage area.

* **core_ring**

Extends the power straps to the core ring in DEFAULT_VA.

Note that, not all the straps in the voltage area are extended out to core ring. The number of straps extended from each side of the voltage area is controlled by the **-num_extend_straps** option.

* **boundary**

Extends the power straps to the top-level cell boundary and creates pins.

By default, PNS extends the power and ground straps to the rings inside the

voltage area (**-extend_strap voltage_area ring**). ring(s) inside.
This option belongs to the ring constraints group and must be specified with either the **-ring_nets** or **-skip_ring** option, if there are existing rings.

-num_extend_straps *number_extended*

Specifies the number of power straps extended out from each side of the voltage area to the core ring. The default is 4.

This option belongs to the ring constraints group and must be specified with either the **-ring_nets** or **-skip_ring** option, if there are existing rings.

-extend_strap_direction *directions_to_extend*

Specifies the directions in which to extend the straps from the voltage area to the core ring or boundary. You can specify one or more of the following keywords:

- * **left**
- * **right**
- * **top**
- * **bottom**

You can also use the keyword **all** to extend straps from all sides of the voltage area.

By default, PNS extends straps from all sides of the voltage area.

This option belongs to the ring constraints group and must be specified with either the **-ring_nets** or **-skip_ring** option, if there are existing rings.

-global

Specifies global constraints in the voltage area.

This option belongs to the global constraints group and is required when specifying global constraints.

-keep_floating_segments

Keeps dangling wires cut by hard macros or blockages in the voltage area.

By default, PNS removes dangling wires.

This option belongs to the global constraints group.

-no_stack_via

Disallows the use of stacked vias in the voltage area. Vias are dropped only on adjacent layers.

By default, PNS generates stacked via.

This option belongs to the global constraints group.

-no_same_width_sizing

Allows power and ground straps to have different widths in the voltage area.

By default, PNS creates straps with the same width for power and ground nets.

This option belongs to the global constraints group.

-optimize_tracks

Optimizes wire width in the voltage area to lower the routing usage by power and ground straps.

By default, wire width is not optimized.

This option belongs to the global constraints group.

-no_routing_over_hard_macros

Disables routing of power and ground straps across hard macros.

By default, power and ground straps can be routed across hard macros.

This option belongs to the global constraints group.

set_fp_rail_voltage_area_constraints

-no_routing_over_plan_groups
 Disables routing of power and ground straps across plan groups.
 By default, power and ground straps can be routed across plan groups.
 This option belongs to the global constraints group.

-no_routing_over_soft_macros
 Disables routing of power and ground straps across soft macros.
 By default, power and ground straps can be routed across soft macros.
 This option belongs to the global constraints group.

-ignore_blockages
 Generates power and ground straps despite the presence of blockages.
 By default, blockages are not ignored.
 This option belongs to the global constraints group.

-allow_routing_over_voltage_area
 Allows routing of power and ground straps of the voltage area to cross other voltage areas. This option is needed when synthesizing inner nested voltage areas.
 By default, routing of power and ground straps cannot cross other voltage areas.
 This option belongs to the global constraints group.

DESCRIPTION

This command defines power network synthesis (PNS) constraints for the specified voltage area. It is used for multivoltage power network synthesis and addresses four types of constraints: synthesis, layer, ring, and global.

You can specify only one group of constraints in each command. The options specifying the constraints for each group follow the required options. For synthesis constraints, you must specify the **-nets** option, unless you are applying synthesis constraints to the primary power and ground nets defined in the UPF. For layer constraints, you must specify the **-layer** option. For ring constraints, you must specify the **-ring_nets** option, unless you are defining ring constraints for the nets associated with the synthesis constraints, or you are defining constraints for the existing rings, in which case you must specify the **-skip_ring** option. For more information, see the descriptions for the ring group options. For global constraints, you must specify the **-global** option.

You can apply constraints to only one voltage area at a time.

Multicorner-Multimode Support

This command is not supported in a multi-scenario design flow.

EXAMPLES

The following example shows how to specify power network synthesis constraints to the MULTIPLIER, INST, and DEFAULT_VA voltage areas. The MULTIPLIER voltage area uses the VDD1 power supply at 1.1V. The INST voltage area is a power-down voltage area that uses the VDD2 power supply at 0.9V. It uses the permanent VDD2 power net and

the virtual VDD2_Virtual power net. The DEFAULT_VA voltage area uses the VDD power supply at 1.5V. The voltage areas have a common ground net, VSS.

```
prompt> set_fp_rail_voltage_area_constraints \  
-voltage_area MULTIPLIER -nets {VDD1 VSS} \  
-voltage_supply 1.1 -power_budget 120 \  
-target_voltage_drop 80  
  
prompt> set_fp_rail_voltage_area_constraints \  
-voltage_area INST -nets {VDD2+VDD2_Virtual VSS} \  
-voltage_supply 0.9 -power_budget 60 \  
-target_voltage_drop 90  
  
prompt> set_fp_rail_voltage_area_constraints \  
-voltage_area DEFAULT_VA -nets {VDD VSS} \  
-voltage_supply 1.5 -power_budget 200 \  
-target_voltage_drop 100
```

The following example shows the commands that apply the same voltage area constraints for a design in which the primary power and ground nets are defined in the UPF, so the **-nets** option is not required.

```
prompt> set_fp_rail_voltage_area_constraints \  
-voltage_area MULTIPLIER \  
-voltage_supply 1.1 -power_budget 120 \  
-target_voltage_drop 80  
  
prompt> set_fp_rail_voltage_area_constraints \  
-voltage_area INST \  
-voltage_supply 0.9 -power_budget 60 \  
-target_voltage_drop 90  
  
prompt> set_fp_rail_voltage_area_constraints \  
-voltage_area DEFAULT_VA \  
-voltage_supply 1.5 -power_budget 200 \  
-target_voltage_drop 100
```

The following example shows how to specify layer constraints for the MULTIPLIER voltage area. Power and ground straps will be generated on the METAL5 METAL6 layers in the voltage area, with a maximum width of 2.5 microns and a minimum width of 1.0 microns.

```
prompt> set_fp_rail_voltage_area_constraints \  
-voltage_area MULTIPLIER -layer {METAL5} \  
-max_width 2.5 -min_width 1.0  
  
prompt> set_fp_rail_voltage_area_constraints \  
-voltage_area MULTIPLIER -layer {METAL6} \  
-max_width 2.5 -min_width 1.0
```

The following example shows how to create a core ring and a voltage area ring for the VDD1 net and extend eight VDD1 straps from the left side of the MULTIPLIER voltage area to the core ring.

```
prompt> set_fp_rail_voltage_area_constraints \  

```

```
-voltage_area DEFAULT_VA -ring_nets {VDD VDD1 VSS} \  
-horizontal_ring_layer METAL7 \  
-vertical_ring_layer METAL8
```

```
prompt> set_fp_rail_voltage_area_constraints \  
-voltage_area MULTIPLIER -ring_nets {VDD1 VSS} \  
-horizontal_ring_layer METAL5 \  
-vertical_ring_layer METAL6 \  
-extend_strap core_ring \  
-num_extend_straps 8 \  
-extend_strap_direction {left}
```

SEE ALSO

```
remove_fp_rail_voltage_area_constraints(2)  
report_fp_rail_voltage_area_constraints(2)  
synthesize_fp_rail(2)
```

set_fp_relative_location

Specifies a constraint to place a macro relative to an anchor object.

SYNTAX

```
status set_fp_relative_location
-name constraint_name
-target_cell cell_name
[-target_orientation N | S | E | W | FN | FS | FE | FW]
[-target_corner bl | br | tl | tr]
[-anchor_object object_name]
[-anchor_corner bl | br | tl | tr]
[-x_offset distance]
[-y_offset distance]
```

Data Types

<i>constraint_name</i>	string
<i>cell_name</i>	string
<i>object_name</i>	string
<i>distance</i>	float

ARGUMENTS

-name *constraint_name*
Specifies name of the constraint.

-target_cell *cell_name*
This option specifies the name of the macro cell which the constraint will be applied.

-target_orientation N | S | E | W | FN | FS | FE | FW
Specifies the orientation to place the target cell. The value can be one of the 8 possible choices represented by the following strings.
N, S, E, W, FN, FS, FE, FW.

The default is N.

-target_corner bl | br | tl | tr
Specifies which corner of the target cell to apply the constraint on. The value can be one of the 4 possible choices represented by the following strings:
bl, br, tl, tr.

These 4 strings represent bottom-left, bottom-right, top-left, and top-right, respectively. The default is bl.

`-anchor_object object_name`
 Specifies the name of the anchor object. An anchor object can be a plan group, a fixed macro cells, a macro cell that has its own relative location constraint, or core area. This option is optional. Default is core area.

`-anchor_corner bl | br | tl | tr`
 Specifies the corner of the anchor object's bounding box that this constraint uses. The value can be one of the 4 possible choices represented by the following strings:
 bl, br, tl, tr.

These 4 strings represent bottom-left, bottom-right, top-left, and top-right, respectively. The default is bl.

`-x_offset distance`
 Specifies the distance from the target corner to the anchor corner in X dimension. The value can be positive, negative or zero. A positive value would indicate the target corner being on the right side of the the anchor corner, and a negative value would result in the target corner being on the left side of the anchor corner, while a value of zero would mean the target corner has the same X coordinate as the anchor corner. The default is zero.

`-y_offset distance`
 Specifies the distance from the target corner to the anchor corner in Y dimension. The value can be positive, negative or zero. A positive value would indicate the target corner being higher than the the anchor corner, and a negative value would result in the target corner being lower than the anchor corner, while a value of zero would mean the target corner has the same Y coordinate as the anchor corner. The default is zero.

DESCRIPTION

The **set_fp_relative_location** command specified a location constraint of a macro cell relative to an anchor object. The macro cell's location will be determined by the parameters given. These location constraints will be respected by `create_fp_placement` command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows how to specify a relative location constraint on cell

"A1"'s top-right corner with respect to the top-left corner of plan group "G1".

```
prompt>>set_fp_relative_location \  
        -name rp1 \  
        -target_cell "A1" \  
-target_orientation "S" \  
-target_corner "tr" \  
-anchor_object "G1" \  
-anchor_corner "tl" \  
        -x_offset 150 \  
        -y_offset 10
```

SEE ALSO

```
create_fp_placement(2)  
remove_fp_relative_location(2)  
extract_fp_relative_location(2)
```


set_fp_shaping_strategy

Sets parameters for controlling the command **shape_fp_blocks**.

SYNTAX

```
status set_fp_shaping_strategy  
[-default]  
[-avoid_power_grid on | off]  
[-distance_to_power_grid float]  
[-keep_top_level_together on | off]  
[-min_channel_size float]  
[-util_slack float]  
[-add_channel_blockages none | partial | soft | hard]  
[-adjust_macro_locations on | off]  
[-preserve_abutment on | off]  
[-max_shape_complexity integer]
```

ARGUMENTS

-default

Set all the default parameter values.

-avoid_power_grid on | off

Controls whether the plan group edges avoid the power grid. The default is off.

Note that this applies to the **shape_fp_blocks** command without the **-incremental** option. In incremental shaping, it applies only to the edges that the **shape_fp_blocks** command decides to move, that is, the command will not decide to move an edge just to "legalize" its status with respect to the power grid, but will put all edges it does move in a "legal" location.

-distance_to_power_grid *float*

Specifies the minimum allowed distance between a plan group boundary and the power grid assuming the **-avoid_power_grid** option is on. The value is in microns. If the value is negative, the tool will use one row height as the minimum distance. The default is -1.

-keep_top_level_together on | off

Controls whether the top level is kept together (contiguous) during non-incremental shaping. The default is off.

-min_channel_size *float*

Specifies the minimum channel size between two plan groups if the **-channels** option is used with the **shape_fp_blocks** command. The value is in microns. The default is 0.

-util_slack *float*

Controls how much the **shape_fp_blocks -incremental** command can exceed the target utilization. The default is 0.1, that is, the target utilization can be exceeded by 10 percent.

`-add_channel_blockages none | partial | soft | hard`
Specifies what type of placement blockages will be added in narrow congested channels by the **shape_fp_blocks -incremental congestion_driven** command. The default is soft blockages.

`-adjust_macro_locations on | off`
Allows the slight movement of macros when the **shape_fp_blocks -incremental congestion_driven** is run to make macro placement legal (no overlaps, the plan group macros are inside their own plan groups, and the top level macros are outside all plan groups). If this cannot be accomplished using small moves, some overlaps may remain to be handled by subsequent placement. The default is off.

`-preserve_abutment on | off`
Controls whether the abutment of plan groups is preserved when the **shape_fp_blocks -incremental target_utilization_driven** command is run. The default is off.

`-max_shape_complexity integer`
Specifies the maximum number of rectangles that plan group shapes can consist of. For example, L-shapes consist of 2 rectangles and U-shapes consist of 3 rectangles. This option applies to the **shape_fp_blocks** command without the **-incremental** option. The default is 0 (unlimited).

DESCRIPTION

This command sets parameters for controlling the design planning **shape_fp_blocks** command. These settings are not applicable to other IC Compiler commands. These settings are not persistent in the database.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLE

The following examples sets the minimum channel size to 10.5 microns.

```
prompt> set_fp_shaping_strategy -min_channel_size 10.5
INFO: Setting min_channel_size = 10.500
```

SEE ALSO

```
report_fp_shaping_strategy(2)
shape_fp_blocks(2)
```

set_fp_trace_mode

Marks a design and then loads the design in trace mode.

SYNTAX

```
status set_fp_trace_mode
[-verbose]
[-verbose]
```

ARGUMENTS

-verbose
Turns on verbose mode for trace mode marking and loading.

EXAMPLE

The following example shows a basic output of the set_fp_trace_mode command.

```
prompt> get_fp_trace_mode
```

```
Design is not in trace mode
0
```

```
prompt> set_fp_trace_mode
```

```
Loading db file '/home/proj/LIB_CENTER/130nm/mp/DB/slow.db'
Loading db file '/home/proj/synopsys/libraries/syn/dw01.sldb'
Loading db file '/home/proj/synopsys/libraries/syn/dw02.sldb'
Loading db file '/home/proj/synopsys/libraries/syn/dw03.sldb'
Loading db file '/home/proj/synopsys/libraries/syn/dw04.sldb'
Loading db file '/home/proj/synopsys/libraries/syn/dw05.sldb'
Loading db file '/home/proj/synopsys/libraries/syn/dw06.sldb'
Loading db file '/home/proj/synopsys/libraries/syn/dw_foundation.sldb'
Loading db file '/home/proj/synopsys/libraries/syn/gtech.db'
Loading db file '/home/proj/synopsys/libraries/syn/standard.sldb'
Information: linking reference library : /home/proj/LIB_CENTER/130nm/mp/MW/
tsmc13fsg_8lm. (PSYN-878)
Building design in trace mode ...
Reading netlist information...
Marking netlist...
Processing plan group 'regs0'
Processing plan group 'ac0'
Processing plan group 'regs0'
Processing plan group 'ac0'
Trace Mode marking completed.
```

```
Linking design 'mp'
Using the following designs and libraries:
```

```
-----
* (686 designs)          trace.CEL, etc
slow (library)           /home/proj/LIB_CENTER/130nm/mp/DB/slow.db
```

```
dw01.sldb (library)      /home/proj/synopsys/libraries/syn/dw01.sldb
dw02.sldb (library)      /home/proj/synopsys/libraries/syn/dw02.sldb
dw03.sldb (library)      /home/proj/synopsys/libraries/syn/dw03.sldb
dw04.sldb (library)      /home/proj/synopsys/libraries/syn/dw04.sldb
dw05.sldb (library)      /home/proj/synopsys/libraries/syn/dw05.sldb
dw06.sldb (library)      /home/proj/synopsys/libraries/syn/dw06.sldb
dw_foundation.sldb (library) /home/proj/synopsys/libraries/syn/dw_foundation.sldb
```

Finished building design in trace mode.

1

```
prompt> get_fp_trace_mode
```

Design is in trace mode

1

SEE ALSO

```
end_fp_trace_mode(2)
```

```
get_fp_trace_mode(2)
```

set_fp_voltage_area_constraints

Sets voltage area feedthrough constraints that are used by the global routing and the **analyze_fp_routing** command to create logical pins on voltage areas.

SYNTAX

```
status set_fp_voltage_area_constraints  
[-allow_feedthroughs true | false]  
[-create_feedthrough_module true | false]  
[-exclude_feedthroughs nets]  
[voltage_areas]
```

Data Types

<i>nets</i>	collection
<i>voltage_areas</i>	collection

ARGUMENTS

-allow_feedthroughs true | false

Specifies whether feedthrough ports can be created on the specified voltage areas.

By default, this option is **false** (no feedthrough ports are created).

If set to **true**, global routing and **analyze_fp_routing** can create feedthrough ports in the logical child cell. Each top-level net for which a feedthrough port is created is split into a set of new top-level nets and child-level nets (if feedthrough nets were created for the child net). Directions are assigned for the newly created feedthrough ports.

If a voltage area boundary is coincident with plangroup boundary, and this feedthrough constraints setting are conflict with set_fp_pin_constraints, voltage area's feedthrough constraints will be ignored.

-create_feedthrough_module true | false

Specifies whether a new logical module should be created to hold the feedthroughs for a voltage area

By default, this option is **false** (no new feedthrough module is created).

If this option is set to **true**, the global router creates only feedthroughs that have no connection inside the voltage area. These internally unconnected feedthroughs, which are sometimes called "pure" feedthroughs, can be assigned ports on a new "feedthrough only module" without affecting the interface ports on the original voltage area modules. The **analyze_fp_routing** command creates a new hierarchy module and assigns the feedthrough ports to this new module, rather than to some existing logical module. The new hierarchy module is a child of the first common ancestor of all the logical modules in the voltage area, and it also is a member of the voltage area. The new hierarchy module becomes part of the logical netlist. It is initially empty, but buffers could be added later for the feedthroughs. The name of those new modules will be "VoltageAreaName_ft_module", where "VoltageAreaName" is the name of voltage area to which the new module belongs. When this option is selected, router will route nets such that only pure feedthrough can be created and analyze_fp_routing will create those new modules and put those pure feedthroughs on those modules. In the case where more than two voltage areas

are fully abutted and the net has connections to all those abutted voltage areas, router can not route net without creating regular feedthrough. In suchy case, `analyze_fp_routing` will skip such net and no feedthrough would be created.

This option is ignored unless it is used together with the **-allow_feedthroughs true** option.

-exclude_feedthroughs nets

Excludes feedthrough port creation on the specified nets.

The effect of this option is not cumulative. A subsequent call to **set_fp_voltage_area_constraints** that uses this option overrides the previous setting. For example, after the following successive calls to **set_fp_voltage_area_constraints**, feedthrough exclusion applies only to net B:

```
set_fp_voltage_area_constraints -allow_feedthroughs true \  
  -exclude_feedthroughs [get_nets A]  
set_fp_voltage_area_constraints -allow_feedthroughs true \  
  -exclude_feedthroughs [get_nets B]
```

To remove all nets previously specified with this option from feedthrough exclusion, use this option with an empty collection as argument, **-exclude_feedthroughs {}**.

This option is ignored unless it is used together with the **-allow_feedthroughs true** option.

voltage_areas

Specifies a collection of voltage areas to which the specified constraints apply.

By default, the specified constraints apply to all voltage areas.

DESCRIPTION

This command allows you to set voltage area feedthrough constraints to be honored during global routing and **analyze_fp_routing**. The voltage area constraints are saved in the Milkyway database.

When route the design using `zroute`, user can NOT set `"-enforce_voltage_area off"` option in `"set_route_zrt_common_options"` command. By setting this option off, `zroute` will ignore constraints that set by `set_fp_voltage_area_constraints`. User needs to set `"-enforce_voltage_area"` option as `"relax"` (by default) or `"strict"` in order to honor constraints that set by `set_fp_voltage_area_constraints`.

Use **set_fp_voltage_area_constraints -allow_feedthroughs false** to reset all constraint values to their defaults.

Multicorner-Multimode Support

This command is not supported in a multi-scenario design flow.

EXAMPLES

The following command allows feedthrough nets to be created on voltage areas A, B, and C.

```
set_fp_voltage_area_constraints  
2464
```

```
prompt> set_fp_voltage_area_constraints -allow_feedthroughs true \  
[get_voltage_areas A B C]
```

SEE ALSO

set_groute_options

Sets global router cell-persistent options.

SYNTAX

```
status set_groute_options  
[-name name]  
[-value value]  
[-default]
```

Data Types

<i>name</i>	string
<i>value</i>	integer real string

ARGUMENTS

-name *name*
Specifies the name of the option to be set. Use command **report_groute_options** to check for valid names.

-value *value*
Specifies the value of the option to be set. Use command **report_groute_options -name** to check for valid range, type, default value, current value and a brief description of a named option.

-default
Set all groute options to their default values. To set a specific groute option to its default value, use "-name" and "-value" arguments. Use command **report_groute_options** to check for default value.

DESCRIPTION

The **set_groute_options** command sets options to tune global router operation. These settings are persistent in the Milkyway design. This command saves cell-persistent options in cell data base, and marks cell as being modified. To permanently save these options, you have to save cell upon exit. This command is optional, if you want defaults to apply. All options are optional too. However, if you want to set a specific option value, "-name" and "-value" have to be specified. To reset all options to their default values, use "-default" argument. To print all available global router options, their current value, default value, range, type and a brief description, use command **report_groute_options**

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

```
prompt> set_groute_options -name accessPolyPin -value 0
```

```
set_groute_options  
2466
```



```
prompt> set_groute_options -default
```

SEE ALSO

```
report_groute_options(2)
```

set_hierarchy_color

Sets colors on all leaf cells descended from top hierarchical cells in the current design or hierarchical cells in the specified collection.

SYNTAX

```
int set_hierarchy_color  
-color color_Id | -cycle_color  
[collection]
```

Data Types

<i>color_Id</i>	integer
<i>collection</i>	collection

ARGUMENTS

-color *color_Id*
Specifies a Milkyway color Id, which is used to set color on all leaf cells descended from hierarchical cells.
The **-color** option is mutually exclusive with the **-cycle_color** option; use one of the options, but not both.

-cycle_color
Specifies to automatically generate different colors for each hierarchical cell and sets the color on all leaf cells descended from the hierarchical cell.
The **-cycle_color** option is mutually exclusive with the **-color** option; use one of the options, but not both.

collection
Specifies a collection of objects on which to set the colors. Sets the colors on all leaf cells descended from hierarchical cells in the specified collection.
This argument can be used with **-color** or **-cycle_color**.

DESCRIPTION

The **set_hierarchy_color** command sets the colors on leaf cells descended from top hierarchical cells in the current design, or hierarchical cells in the specified collection. The colors set on cells are used to paint the cells on some graphic windows, such as Layout View.

When **-color** and *collection* are specified, the command sets the color on all leaf cells descended from hierarchical cells in the collection. When **-color** is specified without *collection*, the command sets the color on all leaf cells in the current design.

When **-cycle_color** and *collection* are specified, the command generates a different color for each hierarchical cell in the collection, and sets the color on all leaf cells descended from that hierarchical cell.

When **-cycle_color** is specified without *collection*, the command generates a different color for each top-level hierarchical cell in the current design, and sets the color on all leaf cells descended from that hierarchical cell.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example generates different colors for each top hierarchical cell in the current design, and sets the color on all leaf cells in the hierarchical cell:

```
prompt> set_hierarchy_color -cycle_color
```

The following example sets the color with Milkyway color Id 8 on all leaf cells inside the RES_l113 hierarchical cell:

```
prompt> set_hierarchy_color -color 8 [get_cell RES_l113]
```

SEE ALSO

`unset_hierarchy_color(2)`

set_host_options

Controls the maximum number of cores used by commands in the parent (master) process. Also controls the options used by commands that can run distributed jobs.

SYNTAX for the Parent (Master) Process

status **set_host_options**

[-max_cores number_of_cores]

SYNTAX for the Child (Slave) Processes

status **set_host_options**

[-name options_name]
[-num_processes number_of_jobs_or_partitions]
[-max_cores number_of_cores]
[-timeout timeout]
[-submit_command batch_submission_command_path]
[-submit_options batch_submission_command_options]
[-32bit]
[-pool lsf | grd]
[list_of_hosts]

Data Types for the Child (Slave) Processes

<i>options_name</i>	string
<i>number_of_jobs_or_partitions</i>	int
<i>timeout</i>	int
<i>batch_submission_command_path</i>	string
<i>batch_submission_command_options</i>	string
<i>list_of_hosts</i>	list

ARGUMENTS

-max_cores *number_of_cores*

If this option is specified by itself, it controls the maximum number of cores that can be used by multicore commands in the current (main) job.

If it is specified as part of a host or pool specification, it controls the number of cores each distributed slave job on the specified hosts or pool are allowed to use.

In both cases, the value is a maximum, so some commands might use fewer cores than the specified value, but in no case would they use more.

-name *options_name*

Specifies a name for this collection of options. The name can later be specified in the *remove_host_options* command to remove only these particular options.

If you do not specify this option, the tool generates a name. If a set of options with the specified name already exists, the specified options replace the existing options.

`-num_processes` *number_of_jobs_or_partitions*

Controls the level of parallelism used for job submission. If you are defining an LSF, GRD, or custom pool, this is the maximum total number of jobs that can be submitted to it. If you are defining a collection of named hosts, this is the maximum number of jobs per host that can be started on those hosts. You can restrict the number of active jobs to a specified number based on hardware or license resource restrictions.
The default is no limit.

`-timeout` *timeout*

Specifies the job submission timeout value in seconds.
If the submitted jobs do not become running jobs within the specified time, they are killed. This is to prevent hanging jobs in the queuing system.
If you do not specify this option, the system waits as long as needed for jobs to start.

`-submit_command` *batch_submission_command_path*

Specifies the full path name of the LSF, GRD, or custom job submission program.
If you do are defining a simple pool of machines and do not specify this option, **rsh** is used as the submission program.
If you specify the **-pool** option and do not specify this option, the system might be able to locate and use the standard LSF or GRD job submission programs.

`-submit_options` *batch_submission_command_options*

Specifies any options to be passed to the LSF, GRD, rsh, or custom job submission program.

`-32bit`

Indicates that the program used by the distributed commands for executing their slave jobs is run in 32-bit mode.
If you do not specify this option, 64-bit mode is used if possible; otherwise, 32-bit mode is used.

`-pool` *lsf | grd*

Used by some commands to optimize their behavior for the specified type of queuing system.
If you are defining a queuing system, you must specify either this option or the **-submit_command** option.
If you are defining a simple pool of machines, this option is meaningless and is not allowed.

`list_of_hosts`

Specifies a list of hosts that are used to run distributed jobs.
If you specify this argument, the **-pool** and **-timeout** options are meaningless and are not allowed.

DESCRIPTION

IC Compiler supports several commands that can be run in distributed mode, in which a task is broken up into many pieces and each subproblem is run independently. The **set_host_options** command defines the parameters for running a command in distributed mode.

It allows for simple setup for the new user and also for arbitrarily complex setup for the user with more sophisticated needs.

In its simplest form, the command is used to control the number of cores used by multicore commands. A multicore command can use fewer cores than this maximum, but the command can never exceed the stated maximum.

The command is versatile enough to interface to LSF, GRD, or a custom job-submission system, and has a mode in which a listed set of machines are used with the rsh command to execute remote jobs without having a queuing system at all.

If you are using a queuing system such as LSF, GRD, or a custom queuing system, you use the **set_host_options** command to specify a path to the batch submission system and a queuing-system-specific set of command-line options that are passed along to the queuing system. These are typically used to ask for specific resources or otherwise control the queuing system. You can also control the submit timeout to make sure that jobs don't get stuck forever in a queuing system.

If you are not using a queuing system, you can still use distributed processing by providing a list of host machines, which the tool then uses to launch remote jobs with the rsh command.

If you run the **set_host_options** command multiple times, the settings are additive. This allows you to define different numbers of jobs to different machines. Because you can specify more than one host, all hosts with the same settings can be configured with a single call, minimizing the number of calls that must be made to build up the host options set.

You can name the collection of options that the **set_host_options** command creates by using the **-name** option. If you do not specify a name, IC Compiler automatically generates a name. You can use the name with the **remove_host_options** command to remove only the options of that name.

You can use the **report_host_options** command to report information about the defined host option sets.

There is a great deal of power and flexibility in the **set_host_options** command. You should start by using a single set of options until you fully understand how host option sets are made and how they are used by distributed commands. The EXAMPLES section below provides examples growing in complexity from basic to complex.

EXAMPLES

The following example sets up multicore operation within the parent process.

```
prompt> set_host_options -max_cores 4
...
## The following command uses max 4 threads.
prompt> some_threaded_command ...
```

The following distributed job example shows the use of host mode. It sets the timeout as well, so that if a submitted job doesn't start within 1200 seconds, the job is killed.

```
prompt> set_host_options -timeout 1200 -num_processes 2 sleepy doc
prompt> set_host_options -timeout 1200 -num_processes 4 grumpy
...
## This runs 8 jobs at once.
prompt> some_distributed_command ...
```

Note that in the previous example there are a total of 8 job slots available (2 each from sleepy and doc, and 4 from grumpy).

The following example shows the use of LSF mode. It sets the maximum number of jobs that can be submitted to 12, chooses the amd64 queue, and names this set of options lsfOptions. This name can be used later to identify or delete this set of options.

```
prompt> set_host_options -pool lsf -num_processes 12 \
-name "lsfOptions" \
-submit_command /lsf/bin/bsub -submit_options "-q amd64"
...
```

The following example shows the use of a non-standard pool. It sets the maximum number of jobs that can be submitted to 8, defines the custom script for job submission, and names this set of options customOptions. This name can be used later to identify or delete this set of options.

```
prompt> set_host_options -num_processes 8 -name "customOptions"\
-submit_command /usr/local/bin/my_submit_command
...
```

SEE ALSO

```
get_dominant_scenarios(2)
remove_host_options(2)
report_host_options(2)
run_parallel_jobs(2)
```

set_ideal_latency

Specifies ideal network latency.

SYNTAX

```
string set_ideal_latency  
[-rise | -fall]  
[-min | -max]  
delay  
object_list
```

Data Types

<i>delay</i>	float
<i>object_list</i>	list

ARGUMENTS

-rise | -fall
Specifies whether the latency is for data rise or data fall transition. If you do not specify **-rise** or **-fall**, both values are set. The **-rise** and **-fall** options are mutually exclusive.

-min | -max
Specifies whether the latency is to be used for minimum delay analysis or maximum delay analysis. By default, the delay is used for both maximum and minimum delay analysis. The **-min** and **-max** options are mutually exclusive.

delay
Specifies the ideal latency value on pins in the ideal network. The delay must be expressed in units consistent with the technology library used during optimization. For example, if the technology library specifies delay values in nanoseconds, *delay* must be expressed in nanoseconds.

object_list
Specifies a list of leaf cell pins or top-level ports that are the startpoints of the timing arcs for which ideal latency is to be set.

DESCRIPTION

Sets the ideal latency on top-level ports and leaf cell pins of the ideal network.

Design Compiler assumes ideal timing for ideal networks and ideal nets, which means pins have a specified ideal latency (from the **set_ideal_latency** command) or zero ideal latency by default. The ideal network is normally used for pre-layout to reduce runtime by avoiding unnecessary DRC optimization and retiming. Ideal latency provides an estimate of the ideal network or ideal nets for pre-layout.

The specified latency value overrides the internally-estimated cell delay value and net delay value. If the specified pins do not belong to the ideal network, ideal nets, or auto disable drc nets, the **report_ideal_network** command generates an error

message and the *delay* value is not used for those pins.

The **set_ideal_latency** command affects all pins in the transitive fanout of the pins or ports. The total ideal latency at an ideal boundary pin is the sum of latency on the ideal network source and all ideal latencies on the path.

You can use **set_ideal_latency** for pins at lower levels of the design hierarchy. Pins are specified in the form of "INSTANCE1/INSTANCE2/PIN_NAME."

To list ideal latency values, use the **report_ideal_network** command.

To remove ideal latency values from a design, use the **remove_ideal_latency** or **reset_design** command.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example specifies a rise latency of 1.2 and a fall latency of 0.9 for ports named *A*, *B*, and *C*.

```
prompt> set_ideal_latency 1.2 -rise {A B C}
prompt> set_ideal_latency 0.9 -fall {A B C}
```

SEE ALSO

```
current_design(2)
remove_ideal_latency(2)
remove_ideal_network(2)
remove_ideal_transition(2)
report_ideal_network(2)
report_timing(2)
set_ideal_net(2)
set_ideal_network(2)
set_ideal_transition(2)
```

set_ideal_net

This command is replaced by `set_ideal_network -no_propagate` under the hood. It is recommended to use `set_ideal_network` instead of `set_ideal_net`.

SYNTAX

```
status set_ideal_net  
net_list
```

Data Types

```
net_list      list
```

ARGUMENTS

```
net_list
```

Specifies a list of names of nets on which the **ideal_net** attribute is to be set. These nets must be visible from the current design.

DESCRIPTION

Starting from the 2004.12 release, the **set_ideal_net** and **set_ideal_network** commands are combined. The **set_ideal_net** command is replaced by **set_ideal_network -no_propagate** command under the hood.

Global driver pins of the specified nets are marked as sources of ideal network. These nets must be visible in the current design. Design Compiler treats a net as an ideal net if all global driver pins of the net are ideal. By default, Design Compiler treats all clock nets as ideal nets. However, this does not include the logic gates in the networks, and the nets driven by those logic gates. The ideal properties of the net's global driver pins do not propagate through logic gates. However, they propagate through hierarchies. Once set on the net's global driver pins, the ideal properties are enabled on all the nets that are electrically connected to those pins.

To prevent clock nets from being treated as ideal nets, use **set_auto_disable_drc_nets -clock false** or **set_propagated_clock -all_clocks()**.

After **set_propagated_clock -all_clocks()** has executed, the **ideal_net** attribute is not reported by **report_net**.

Ideal nets are networks of nets that are free from `max_capacitance`, `max_fanout`, and `max_transition` design rule constraints. Ideal nets are useful for reducing DRC violations caused by clock trees, because these networks usually have high `max_capacitance` and `max_fanout` violations. Ideal nets use ideal latency specified by the **set_ideal_latency** command and ideal timing specified by the **set_ideal_transition** for timing calculation. By default, zero ideal latency and zero transition are used. To automatically spread ideal attributes through a network, use the **set_ideal_network** command without the **-no_propagate** option.

Design Compiler automatically sets **size_only** on the cells of the ideal network

sources. And ideal net implies that this net is dont_touch. Note that Design Compiler may still optimize away combinational cells in the fanout of the ideal net. However, Design Compiler guarantees that the ideal network sources are never lost. To automatically set clock, constant or scan nets in the design as ideal, use the **set_auto_disable_drc_nets** command.

Use **remove_ideal_net** command to revert the result from **set_ideal_net** command in the current design. **set_auto_disable_drc_nets -none** removes the **auto_disable_drc_net** attribute. **reset_design** removes all attributes from the design, including both the ideal attributes and **auto_disable_drc_net** attributes; however, by default Design Compiler still treats clock networks as ideal nets.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets the **ideal_net** attribute on nets in design CLOCK_GEN:

```
prompt> current_design CLOCK_GEN
prompt> set_ideal_net [get_nets]
```

SEE ALSO

```
remove_ideal_net(2)
reset_design(2)
set_auto_disable_drc_nets(2)
set_ideal_latency(2)
set_ideal_network(2)
set_ideal_transition(2)
set_dont_touch(2)
set_dont_touch_network(2)
```

set_ideal_network

Marks a set of ports or pins in the current design as sources of an ideal network. This disables timing update and optimization of cells and nets in the transitive fanout of the specified objects.

SYNTAX

```
integer set_ideal_network  
object_list  
[-dont_care_placement]  
[-no_propagate]
```

Data Types

object_list list

ARGUMENTS

object_list
Specifies a list of objects (ports, pins, or nets) to mark as the sources of an ideal network. If more than one object name is specified, each must be enclosed in quotation marks or braces {}. The source objects are input ports on the design or any internal pin, except for pins at hierarchical boundaries. If nets are provided, all of the nets' global driver pins are marked as ideal network sources, and the corresponding ideal network attributes are actually set on all of the global driver pins of the specified nets. These objects must be visible from the current design.
The **set_ideal_network** command accepts nets only when you specify the **-no_propagate** option.

-dont_care_placement
Indicates that the ideal network is not considered in placement. The nets in the ideal network are treated as disconnected. Random locations are assigned to ideal network cells. By default, the ideal network is placed at the lowest priority.

-no_propagate
Indicates that the ideal network is not propagated through logic gates, but it still propagates through hierarchies. Ideal properties are enabled on all nets that are electrically connected to ideal network sources. By default, this option is off.

DESCRIPTION

This command marks a set of ports or pins in the current design as sources of an ideal network. Ideal networks are an extension of ideal nets that incorporate automatic propagation of the **ideal** attribute. You specify only the source of the network; the **compile** command treats all nets, cells, and pins on the transitive fanout of these objects as ideal. The ideal property is automatically spread by the tool and respread as necessary during **compile** optimizations. The criteria for propagating the ideal property, starting at the source pins and ports, are as

follows:

- A pin is marked as ideal if you specify it by using the **set_ideal_network** command if it is either a driver pin and its cell is ideal or it is a load pin attached to a net that is ideal.
- A net is marked as ideal if all of its driving pins are ideal.
- A combinational cell is marked as ideal if all of its input pins are either ideal or attached to a constant net (and other input pins are ideal). Objects with the **case analysis** attribute set are not treated as constant.

Propagation traverses through combinational cells but stops at sequential cells. If an ideal network overlaps a clock network, the clock timing overrides the ideal timing for the clock part of the network.

In addition to disabling timing updates and timing optimizations, all cells and nets in the ideal network have the **dont_touch** attribute set.

The **size_only** attribute is set on all cells of ideal network sources. If nets are specified, **size_only** is set on all cells that are cells of the specified nets' global driver pins. This guarantees that ideal network sources are not optimized away by compile.

NOTE: The implied **size_only** attribute set by this command overrides the four **FALSE** attribute values set by the **set_size_only** command. Use the **report_cell** command to determine if a cell is **size_only** or not.

DRC checking is turned off so the nets in an ideal network are free of **max_capacitance**, **max_fanout**, and **max_transition** design rule constraints. Disabling all of these features improves runtime and timing optimization; for example, by not resetting and scanning networks that you might want synthesized separately when using the **compile** command.

The latency and transition times of an ideal network are 0 by default, but you can override them by using the **set_ideal_latency** and **set_ideal_transition** commands.

If the **ideal_network** attribute is set on floating objects in a design, these objects are still optimized away during compile.

To reverse the effect of the **set_ideal_network** command, use the **remove_ideal_network** command and specify the source pins or ports for the network. Timing, source pins, and boundary pins of an ideal network are displayed by using the **report_ideal_network** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates an ideal network on objects in a design named **CLOCK_GEN**.

```
prompt> current_design CLOCK_GEN  
prompt> set_ideal_network {port1 port2}
```

SEE ALSO

```
current_design(2)  
remove_ideal_network(2)  
report_cell(2)  
report_ideal_network(2)  
reset_design(2)  
set_auto_disable_drc_nets(2)  
set_dont_touch(2)  
set_dont_touch_network(2)  
set_ideal_latency(2)  
set_ideal_net(2)  
set_ideal_transition(2)  
set_size_only(2)
```

set_ideal_transition

Specifies ideal transition for the ideal network & ideal nets.

SYNTAX

```
string set_ideal_transition  
[-rise | -fall]  
[-min | -max]  
transition_time  
object_list
```

Data Types

<i>transition_time</i>	float
<i>object_list</i>	list

ARGUMENTS

-rise | -fall

Indicates whether the specified transition time is applicable for rising or falling transition. If you do not specify **-rise** or **-fall**, both values are set. The **-rise** and **-fall** options are mutually exclusive.

-min | -max

Specifies whether the transition is to be used for minimum delay analysis or maximum delay analysis. By default, the delay is used for minimum and maximum delay analysis. The **-min** and **-max** options are mutually exclusive.

transition_time

Specifies the ideal transition value on the pins in an ideal network. The transition time must be expressed in units consistent with the technology library used during optimization. For example, if the technology library specifies transition values in nanoseconds, *transition_time* must be expressed in nanoseconds.

object_list

Specifies a list of leaf cell pins or top-level ports that are the startpoints of the timing arcs for which ideal transition is set.

DESCRIPTION

Sets the ideal transition on top-level ports and leaf cell pins of an ideal network.

Design Compiler assumes ideal timing for the ideal network, which means pins have a specified ideal transition (from the **set_ideal_transition** command) or zero ideal transition by default. The ideal network is normally used for pre-layout to reduce runtime by avoiding unnecessary DRC optimization and retiming. Ideal transition provides an estimate of the ideal network for pre-layout.

The specified transition value overrides the internally-estimated cell and net transition value. If the specified pins do not belong to the ideal network, an error

message is generated by the **report_ideal_network** command and *transition_time* is not used for those pins.

The ideal transition at an ideal boundary pin is the ideal transition of the closest pin with specified ideal transition.

You can use **set_ideal_transition** for pins at lower levels of the design hierarchy. Pins are specified in the form of "INSTANCE1/INSTANCE2/PIN_NAME."

To list ideal transition values, use the **report_ideal_network** command.

To remove the ideal transition values from a design, use the **remove_ideal_transition** or **reset_design** command.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example specifies a rise transition of 1.2 and a fall transition of 0.9 for ports named A, B, and C.

```
prompt> set_ideal_transition 1.2 -rise {A B C}
prompt> set_ideal_transition 0.9 -fall {A B C}
```

SEE ALSO

```
current_design(2)
remove_ideal_latency(2)
remove_ideal_network(2)
remove_ideal_transition(2)
report_ideal_network(2)
report_timing(2)
set_ideal_latency(2)
set_ideal_network(2)
```


set_ignore_cell_timing

Skips the analysis of part of the design.

SYNTAX

```
int set_ignore_cell_timing  
cell_list
```

Data Types

cell_list list

ARGUMENTS

cell_list
Specifies a collection of cells on which the **ignore_cell_timing** attribute is to be set. These cells must be visible from the current design.

DESCRIPTION

The `set_ignore_cell_timing` command is used in the exploration phase of the design process to skip the analysis of part of the design. If there is one or more hierarchical blocks that are either incomplete, or require large runtimes they can be set as ignored so that optimization and analysis can concentrate on other parts of the design. Leaf cells may also be set as ignored.

The effect of `set_ignore_cell_timing` is to suppress all timing paths that are within, through, into, or out of that cell. It also suppresses any changes to the netlist due to timing or DRC reasons.

Multicorner-Multimode Support

This command uses information from active scenarios only.

EXAMPLES

The following example sets the **ignore_cell_timing** attribute on cells in design `CLOCK_GEN`.

```
prompt> current_design CLOCK_GEN  
prompt> set_ignore_cell_timing get_cells(cell)
```

SEE ALSO

```
remove_ignore_cell_timing(2)  
set_ideal_network(2)
```

set_ignored_layers

Sets ignored routing layers for RC estimation and congestion analysis. This command can also set the minimum and maximum routing layers for the design.

SYNTAX

```
status set_ignored_layers
[-rc_congestion_ignored_layers layer_names]
[-min_routing_layer min_name]
[-max_routing_layer max_name]
```

Data Types

<i>layer_names</i>	list
<i>min_name</i>	string
<i>max_name</i>	string

ARGUMENTS

-rc_congestion_ignored_layers *layer_names*
Lists the routing layers to be ignored in congestion analysis and RC estimation. This option removes the specified layers from both RC estimation and congestion analysis, but does not affect the behavior of the router.

-min_routing_layer *min_name*
Specifies the minimum routing layer for the design. The router respects this setting. To ensure that RC estimation and congestion analysis stay synchronized with the router, when you set this option, the tool marks all routing layers below the specified layer as ignored layers for RC and congestion estimation.

-max_routing_layer *max_name*
Specifies the maximum routing layer for the design. The router respects this setting. To ensure that RC estimation and congestion analysis stay synchronized with the router, when you set this option, the tool marks all routing layers above the specified layer as ignored layers for RC and congestion estimation.

DESCRIPTION

This command sets the routing layers that are ignored during congestion analysis and RC estimation. If a routing layer is ignored, congestion analysis and interconnect RC estimation do not use this layer. You can use this command when you do not want to route nets on certain routing layers.

If you set the minimum and maximum routing layers for a design by using this command, the router respects these settings. If you set the minimum routing layer for the design to a certain layer, this command automatically sets all the routing layers below this layer as ignored layers. The same behavior also applies to the maximum routing layer setting.

By default, the minimum and maximum routing layer settings are hard constraints. You can adjust the hardness of these constraints by setting detail route options. If you are using the classic router, use the **set_droute_options** command to set the **hardMinLayerConx**, **hardMaxLayerConx**, and **viaOverMaxLayer** detail route options. If you are using Zroute, use the **set_route_zrt_common_options** command to set the **min_layer_mode**, **max_layer_mode**, **number_of_vias_under_min_layer**, and **number_of_vias_over_max_layer** common routing options. For more information about these options, see the man page for the option setting command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example assigns the m5 and m6 layers to be ignored during congestion analysis and RC estimation:

```
prompt> set_ignored_layers -rc_congestion_ignored_layers {m5 m6}
```

The following example sets m2 as the minimum routing layer and m5 as the maximum routing layer. All layers below m2 and above m5 are set as ignored layers.

```
prompt> set_ignored_layers -min_routing_layer m2 -max_routing_layer m5
```

SEE ALSO

```
remove_ignored_layers(2)  
report_ignored_layers(2)  
report_lib(2)  
set_droute_options(2)  
set_route_zrt_detail_options(2)
```

set_input_delay

Sets input delay on pins or input ports relative to a clock signal.

SYNTAX

```
status set_input_delay  
delay_value  
[-clock clock_name]  
[-clock_fall]  
[-level_sensitive]  
[-network_latency_included]  
[-source_latency_included]  
[-rise]  
[-fall]  
[-max]  
[-min]  
[-add_delay]  
port_pin_list
```

Data Types

<i>delay_value</i>	float
<i>clock_name</i>	string
<i>port_pin_list</i>	list

ARGUMENTS

delay_value
Specifies the path delay. The *delay_value* must be in units consistent with the technology library used during optimization. The *delay_value* represents the amount of time the signal is available after a clock edge. This represents a combinational path delay from the clock pin of a register.

-clock *clock_name*
Specifies the clock to which the specified delay is related. If **-clock_fall** is used, **-clock clock_name** must be specified. If **-clock** is not specified, the delay is relative to time zero for combinational designs. For sequential designs, the delay is considered relative to a new clock with the period determined by considering the sequential cells in the transitive fanout of each port.
The *clock_name* can be either a string or collection of one object.

-clock_fall
Specifies that the delay is relative to the falling edge of the clock. The default is the rising edge.

-level_sensitive
Specifies that the source of the delay is a level-sensitive latch. This allows the tool to derive the setup and hold relationship for paths from this port as if the port is a level-sensitive latch. If **-level_sensitive** is not used, the input delay is treated as if it is a path from a flip-flop.

-network_latency_included

Specifies that the clock network latency is not added to the input delay value. If this option is not specified, the clock network latency of the related clock is added to the input delay value. It has no effect if the clock is propagated or the input delay is not specified with respect to any clock.

-source_latency_included

Specifies that the clock source latency is not added to the input delay value. If this option is not specified, the clock source latency of the related clock will be added to the input delay value. It has no effect if the input delay is not specified with respect to any clock.

-rise

Specifies that *delay_value* refers to a rising transition on specified ports of the current design. If neither **-rise** nor **-fall** is specified, rising and falling delays are assumed to be equal.

-fall

Specifies that *delay_value* refers to a falling transition on specified ports of the current design. If neither **-rise** nor **-fall** is specified, rising and falling delays are assumed equal.

-max

Specifies that *delay_value* refers to the longest path. If neither **-max** nor **-min** is specified, maximum and minimum input delays are assumed equal.

-min

Specifies that *delay_value* refers to the shortest path. If neither **-max** nor **-min** is specified, maximum and minimum input delays are assumed equal.

-add_delay

Specifies whether to add delay information to the existing input delay or to overwrite. The **-add_delay** option enables you to capture information about multiple paths leading to an input port that are relative to different clocks or clock edges.

For example, the following command removes all other maximum rise input delay from A, since **-add_delay** is not specified. Other input delay with a different clock or with **-clock_fall** is removed.

```
set_input_delay 5.0 -max -rise -clock phil {A}
```

In the following example, **-add_delay** is specified. If there is an input maximum rise delay for A relative to clock *phil* rising edge, the larger value is used. The smaller value will not result in critical timing for maximum delay. For minimum delay, the smaller value is used. If there is maximum rise input delay relative to a different clock or different edge of the same clock, it remains with the new delay.

```
prompt> set_input_delay 5.0 -max -rise -clock phil -add_delay {A}
```

port_pin_list

Specifies a list of input port or internal pin names in the current design to which *delay_value* is assigned. If more than one object is specified, the objects are enclosed in quotes (") or in braces ({}). If input delay is specified on a pin, the cell of the pin is set to size only to leave room for compile applying sizing on it.

DESCRIPTION

The **set_input_delay** command sets input path delay values for the current design. Used with **set_load** and **set_driving_cell**, the input and output delays characterize the operating environment of the current design.

A path starts from a primary input or clock of sequential element and ends at a sequential element or primary output. The **delay_value** to be specified is the delay between the startpoint and the object on which the **set_input_delay** is being set, relative to the clock edge.

The **set_input_delay** command sets input path delays on input ports relative to a clock edge. Input ports are assumed to have zero input delay, unless specified. For inout (bidirectional) ports, you can specify the path delays for both input and output modes.

To describe a path delay from a level-sensitive latch, use the **-level_sensitive** option. If the latch is positive-enabled, set the input delay relative to the rising clock edge; if it is negative-enabled, set the input delay relative to the falling clock edge. If time is being borrowed at that latch, add that time borrowed to the path delay from the latch when determining input delay.

The **characterize** command automatically sets input and output delay, drive, and load values based on the environment of a cell instance.

The timer adds input delay to path delay for paths starting at primary inputs and output delay for paths ending at primary outputs.

Use the **report_port** command to list input delays associated with ports.

To list input delays of internal pins, use **report_design**.

Use **remove_input_delay** or **reset_design** to remove input delay values.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example sets an input delay of 2.3 for ports IN1 and IN2 on a combinational design. Because the design is combinational, no clock is needed.

```
prompt> set_input_delay 2.3 {IN1 IN2}
```

The following example uses a clock collection and sets an input delay of 1.2 relative to the rising edge of CLK1 for all input ports in the design:

```
prompt> set_input_delay 1.2 -clock [get_clocks CLK1] [all_inputs]
```

The following example sets the input and output delays for the bidirectional port INOUT1. The input signal arrives at INOUT1 2.5 units after the falling edge of CLK1. The output signal is required at INOUT1 at 1.4 units before the rising edge of CLK2.

```
prompt> set_input_delay 2.5 -clock CLK1 -clock_fall {INOUT1}
```

```
prompt> set_output_delay 1.4 -clock CLK2 {INOUT1}
```

The following example has three paths to the IN1 input port. One of the paths is relative to the rising edge of CLK1. Another path is relative to the falling edge of CLK1. The third path is relative to the falling edge of CLK2. The **-add_delay** option is used to indicate that new input delay information will not cause old information to be removed.

```
prompt> set_input_delay 2.2 -max -clock CLK1 -add_delay {IN1}
```

```
prompt> set_input_delay 1.7 -max -clock CLK1 -clock_fall \  
-add_delay {IN1}
```

```
prompt> set_input_delay 4.3 -max -clock CLK2 -clock_fall \  
-add_delay {IN1}
```

In this example, two different maximum delays and two minimum delays for port A are specified using **-add_delay**. Because the information is relative to the same clock and clock edge, only the largest of the maximum values and the smallest of the minimum values are maintained, in this 5.0 and 1.1. If **-add_delay** is not used, the new information overwrites the old information.

```
prompt> set_input_delay 3.4 -max -clock CLK1 -add_delay {A}
```

```
prompt> set_input_delay 5.0 -max -clock CLK1 -add_delay {A}
```

```
prompt> set_input_delay 1.1 -min -clock CLK1 -add_delay {A}
```

```
prompt> set_input_delay 1.3 -min -clock CLK1 -add_delay {A}
```

SEE ALSO

```
all_inputs(2)  
characterize(2)  
create_clock(2)  
current_design(2)  
remove_input_delay(2)  
report_design(2)  
report_port(2)  
reset_design(2)  
set_driving_cell(2)  
set_load(2)  
set_output_delay(2)
```

set_input_transition

Sets the **max_transition_rise**, **max_transition_fall**, **min_transition_rise**, or **min_transition_fall** attributes to the specified transition values on the specified input and inout ports.

SYNTAX

```
int set_input_transition  
transition  
[-rise] [-fall] [-min] [-max]  
port_list
```

Data Types

<i>transition</i>	float
<i>port_list</i>	list

ARGUMENTS

transition

Specifies a nonnegative transition value to which the **max_transition_rise**, **max_transition_fall**, **min_transition_rise** or **min_transition_fall** attributes are to be set on the ports in *port_list*. The *transition* is the transition time of a slope that drives the port, such that a higher transition value means longer delays. Thus, a *transition* of 0 is infinite transition, or no delay between the ports and all that is connected to them. *transition* must be in units consistent with the technology library used during optimization.

-rise -fall

Indicates that the **rise** or **fall** attributes of *port_list* are to be set to *transition*. If neither is specified, both are set to *transition*.

-min -max

Indicates that the transition value is to be applied for minimum or maximum delay analysis. If no value is specified for minimum analysis, the maximum value is used.

port_list

Specifies a list of names of input or inout ports in the current design, on which the transition value is to be set. If you specify more than one port, you must enclose the ports in either quotes or braces ({}).

DESCRIPTION

Sets the **max_transition_rise**, **max_transition_fall**, **min_transition_rise** or **min_transition_fall** attributes to *transition* on specified input and inout ports in the current design.

Note: The **set_input_transition** command removes any corresponding rise or fall driving cell attributes and rise or fall drive attributes on the specified ports. If you want a fixed transition value, use **set_input_transition**, otherwise, use

set_driving_cell instead of **set_drive**, if possible.

To view transition information on ports, use **report_port -verbose**. To remove transition attributes from ports, use **remove_attribute**. The **reset_design** command removes all attributes from a design, including the transition attributes.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example sets the maximum rise and fall transition values of ports "A", "B", and "C" to 7.0.

```
prompt> set_input_transition 7.0 {A B C}
```

The following example sets the maximum rise and fall transition values of all input ports to 7 and sets the maximum rise transition value on port "B" to 11.

```
prompt> set_input_transition 7 [all_inputs]  
prompt> set_input_transition -rise 11 B
```

The following example sets the minimum fall transition value 13 to port C

```
prompt> set_input_transition -fall -min 13.0 {C}
```

SEE ALSO

```
all_inputs(2)  
current_design(2)  
remove_attribute(2)  
report_port(2)  
reset_design(2)  
set_drive(2)  
set_driving_cell(2)  
set_load(2)
```

set_inter_clock_delay_options

Sets options for interclock delay balancing.

SYNTAX

```
status set_inter_clock_delay_options
[-balance_group source_objects]
[-balance_group_name string]
[-delay_offset float]
[-offset_to source_objects]
[-offset_from offset_from_obj]
[-offset_from_group string]
[-target_delay_clock target_clock_obj]
[-target_delay_value float]
[-honor_sdc true | false]
```

ARGUMENTS

- balance_group *source_objects*
Set a list of clocks as a group, inter clock delay balance will be performed among clocks in the group.
- balance_group_name *string*
Specifies a group name for the clocks defined under -balance_group. This option is optional.
- delay_offset *float*
Specifies delay offset value, positive value means clock under offset_to is lagging clock under offset_from, negative value means clock under offset_to is ahead clock under offset_from.
- offset_to *source_objects*
Specifies the clock(s) get set an offset from another clock under offset_from.
- offset_from *offset_from_obj*
Specifies the base clock for offset set to clock(s) under offset_to.
- offset_from_group *string*
Specifies a balance group for offset set to clock(s) under offset_to. The clock with the longest path delays in the balance group will be used as the offset from clock.
- target_delay_clock *target_clock_obj*
Specifies a clock of a target delay under -target_delay_value.
- target_delay_value *float*
Specifies target delay value of clock(s) under -target_delay_clock.
- honor_sdc true | false
Honors the latency defined in the Synopsys Design Constraints and performs interclock delay balance based on the command-line options, when this option

is set to **true**. When the option is either set to **false** (the default value) or not specified, the tool ignores the latency defined in the Synopsys Design Constraints, even if available.

DESCRIPTION

The **set_inter_clock_delay_options** command sets the options for inter clock delay balance. By using **-delay_offset**, **-offset_to** and **-offset_from** combined, the insertion delays under **-offset_to** are balanced to a clock under **-offset_from** with an offset value, a positive offset value means clocks under **-offset_to** are lagging and a negative value means **-offset_to** are ahead of the base clock under **-offset_from** or longest delay clock under balance group given through **-offset_from_group**. With **-target_delay_clock** and **-target_delay_value**, clocks under **-target_delay_clock** are balanced to the delay value specified by **-target_delay_value**. For different clocks with different target delay values, use separate **set_inter_clock_delay_options** command line. By using **-balance_group** and **-balance_group_name**, the inter clock balance will be performed among clocks in its own group.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command performs inter clock delay balance using set maximum target delay options:

```
prompt> balance_inter_clock_delay -max_target_delay 0.5
```

The following command performs inter clock delay balance for the *Clk1* clock tree to 0.5ns and */fIClk2* to 0.8ns, by using **set_inter_clock_delay_options** command and **balance_inter_clock_delay** command:

```
prompt> set_inter_clock_delay_options -target_delay_clock CLK1 -  
target_delay_value 0.5
```

```
prompt> set_inter_clock_delay_options -target_delay_clock CLK2 -  
target_delay_value 0.8
```

```
prompt> balance_inter_clock_delay
```

The following command performs inter clock delay balance for the *Clk1* clock tree lags *Clk2* by 0.5ns, by using **set_inter_clock_delay_options** command and **balance_inter_clock_delay** command:

```
prompt> set_inter_clock_delay_options -delay_offset 0.5 -offset_to Clk1 -  
offset_from Clk2
```

```
prompt> balance_inter_clock_delay
```

The following command performs inter clock delay balance by groups. Clk1 and Clk2 are in group1, while clk3, clk4 and clk5 are in group2 by using set_inter_clock_delay_options command and balance_inter_clock_delay command:

```
prompt> set_inter_clock_delay_options -balance_group "Clk1 Clk2" -  
balance_group_name group1
```

```
prompt> set_inter_clock_delay_options -balance_group "Clk3 Clk4 Clk5" -  
balance_group_name group2
```

```
prompt> balance_inter_clock_delay
```

SEE ALSO

balance_inter_clock_delay(2)

set_isolate_ports

Specifies the ports that are to be isolated from internal fanouts of their driver nets.

SYNTAX

```
int set_isolate_ports
[-type inverter | buffer]
[-driver cell_name]
[-force]
port_list
```

Data Types

```
cell_name      string
port_list      list
```

ARGUMENTS

-type inverter | buffer
Specifies that inverter pairs or buffers should be used to isolate the ports.

-driver *cell_name*
Indicates that the specified cell should be used to isolate the ports. The cell must be a buffer or an inverter in the target library. Note that Design Compiler may size the inserted cell, which could cause the actual driver cell to be different from the requested isolation cell. If this behavior is not desired, use the **-force** option in conjunction with the **-driver** option, to force the driver to be the requested isolation cell.

-force
Specifies that isolation must be performed on the ports, even if they do not need isolation because there are no internal loads on the nets driving the ports. Note: Isolation may not be performed if there is already a local buffer, inverter pair or cell corresponding to the specified isolation logic.

port_list
Specifies the ports in the current design that are to be isolated.

DESCRIPTION

The **set_isolate_ports** command specifies the input or output ports in the current design that are to be isolated. For an output port the **compile** command attempts to ensure that the net driving the port does not have any other internal fanout by inserting an isolation cell. For an input port an isolation cell is inserted if the port is driving multiple cells or if the port is driving a pin of a cell that contains more than one input pin. By default, no action is taken if the above conditions are not met. If the **-force** option is specified, isolation is performed on the port even if these conditions are not met. If the type of isolation cell to be inserted is not specified, a buffer is inserted.

Note that bidirectional ports cannot be isolated using this command. In addition, no isolation is performed on a port by **compile** if the net connected to the port has a **dont_touch** attribute or if the port has been defined as a clock source.

Use **remove_isolate_ports** or **reset_design** to remove a port from the list of ports to be isolated. Use **report_isolate_ports** to list the isolation status of these ports.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example requests isolation on output port `qout` with a buffer.

```
prompt> set_isolate_ports -type buffer qout
```

The following command specifies isolation using a particular cell from the target library on an output port.

```
prompt> set_isolate_ports -driver IVDAP qout
```

SEE ALSO

```
remove_isolate_ports(2)  
report_isolate_ports(2)  
reset_design(2)
```

set_isolation

Defines the UPF isolation strategy for the power domains in the design. This command is supported only in UPF mode.

SYNTAX

```
status set_isolation  
isolation_strategy  
-domain power_domain  
-isolation_power_net isolation_power_net  
-isolation_ground_net isolation_ground_net  
[-clamp_value 0 | 1 | z | latch]  
[-applies_to inputs | outputs | both]  
[-elements objects]  
[-no_isolation]
```

Data Types

<i>isolation_strategy</i>	string
<i>power_domain</i>	string
<i>isolation_power_net</i>	string
<i>isolation_ground_net</i>	string
<i>objects</i>	list

ARGUMENTS

isolation_strategy
Specifies the UPF isolation strategy name. The isolation strategy name should be unique within the specified power domain.

-domain *power_domain*
Specifies the power domain name that this UPF isolation strategy will be applied to.

-isolation_power_net *isolation_power_net*
Specifies the isolation power net for the isolation cells that will be created based on this UPF isolation strategy.

-isolation_ground_net *isolation_ground_net*
Specifies the isolation ground net for the isolation cells that will be created based on this UPF isolation strategy. At least one of the -**isolation_power_net** or the -**isolation_ground_net** options should be specified if -**no_isolation** is not specified.

-clamp_value 0 | 1 | z | latch
Specifies the clamp value of the isolation cells that should be created based on this strategy. The default value for this option is 0.

-applies_to inputs | outputs | both
Specifies whether the given **set_isolation** command applies to all inputs or outputs or both kind of ports of the power domain. The default value for this option is **outputs**. This option cannot be used with the -**elements** option.

-elements *objects*

Specifies the objects that this UPF isolation strategy will be applied to. The objects can be pins of the root cells of the power domain or ports of the top level design for top level power domains.

-no_isolation

Specifies that elements under the influence of this **set_isolation** command should not be isolated.

DESCRIPTION

This command defines the UPF isolation strategy for the ports of the specified power domain.

If the **-elements** and **-applies_to** options are not specified, then the isolation strategy will be applied to all output ports of the power domain.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows how to define a UPF isolation strategy for all output ports of the specified power domain PD1. Power domain PD1 is defined on instance shutdown_inst.

```
prompt> set_isolation isolation_1 -domain PD1 \  
       -isolation_power_net PN1 -isolation_ground_net GN1
```

The following example shows how to define a UPF isolation strategy for specific ports of the specified power domain:

```
prompt> set_isolation isolation_2 -domain PD1 \  
       -isolation_power_net PN1 -isolation_ground_net GN1 \  
       -elements shutdown_inst/special_port
```

SEE ALSO

set_isolation_control(2)

set_isolation_cell

On-The-Fly setting of library isolation cell.

SYNTAX

```
Boolean set_isolation_cell  
cell_name  
[-data_pin {data_pin_name}]  
[-enable_pin {enable_pin_name}]  
[-reset]
```

Data Types

<i>cell_name</i>	string
<i>data_pin_name</i>	string
<i>enable_pin_name</i>	string

ARGUMENTS

cell_name
Specify the library cell as isolation cell.

-data_pin {*data_pin_name*}
Specify the isolation cell data pin name.

-enable_pin {*enable_pin_name*}
Specify the isolation cell enable pin name.

-reset
reset the isolation cell attributes.

DESCRIPTION

The **set_isolation_cell** command set all the cells with the same name specified as isolation cell. Data and enable pin option can be specified for the isolation cell.

If -reset is issued, the isolation cell attribute will be cleaned up.

If -reset is combined with other options, reset happens first, other options will be set on the cell after reset.

EXAMPLES

```
prompt> set_isolation_cell ISO* -enable_pin {EN}
```

SEE ALSO

`report_lib(2)`
`check_library(2)`

set_isolation_control

Provides additional options needed for creating isolation cells. This command is needed with most **set_isolation** commands. This command is supported only in UPF mode.

SYNTAX

```
status set_isolation_control  
isolation_strategy  
-domain power_domain  
-isolation_signal isolation_signal  
[-isolation_sense low | high]  
[-location self | parent]
```

Data Types

<i>isolation_strategy</i>	string
<i>power_domain</i>	string
<i>isolation_signal</i>	string

ARGUMENTS

isolation_strategy
Specifies the UPF isolation strategy name. The isolation strategy should already be defined in the specified power domain.
This argument is required.

-domain *power_domain*
Specifies the power domain name to which this UPF isolation strategy will be applied.
This option is required.

-isolation_signal *isolation_signal*
Specifies the isolation signal to use as the control signal of the isolation cells that should be created as a result of this isolation strategy. The *isolation_signal* can be a pin, port, or net.
This option is required.

-isolation_sense low | high
Specifies the isolation sense of the isolation cells that should be created as a result of this isolation strategy. The default value is high.

-location self | parent
Specifies the hierarchical location of the isolation cells that should be created as a result of this isolation strategy.
A value of self specifies that the isolation cell will be put in the hierarchy of the port being isolated. A value of parent specifies that isolation cells will be added in the parent hierarchy of the isolating port's hierarchy. The default value is self.

DESCRIPTION

This command applies to an existing isolation strategy specified by the **set_isolation** command. It provides extra parameters needed for creating isolation cells.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows how to define a UPF isolation strategy for all output ports of the specified power domain PD1. Power domain PD1 is defined on instance shutdown_inst, and isolation strategy isolation_1 is already defined.

```
prompt> set_isolation_control isolation_1 -domain PD1 \  
-location parent -isolation_signal en \  
-isolation_sense
```

SEE ALSO

set_isolation(2)

set_keepout_margin

Creates a keepout margin of the specified type for the specified cell or library cell.

SYNTAX

```
status set_keepout_margin  
[-type hard | soft]  
[-outer {lx by rx ty}]  
[-tracks_per_macro_pin value]  
[-min_padding_per_macro value]  
[-max_padding_per_macro value]  
[-all_macros]  
[-macro_masters]  
[-macro_instances]  
[-north]  
[object_list]
```

ARGUMENTS

-type hard | soft

Specifies the type of the keepout margin to be created. The value can be either soft or hard. If not specified hard will be taken as default value.

-outer {lx by rx ty}

Specifies the margin parameters for a keepout. lx, by, rx and ty represents the left, bottom, right, and top margins respectively. The margin parameters must be positive and all the four quantities are required. The unit for specification is micron. If the margins have been set on a standard cell instance or a standard cell master, if the standard cell instance is "fixed", then all four margins will be honored. However, if the standard cell instance is "movable", then for horizontal designs, only "lx" and "rx" will be honored; for vertical designs, only "by" and "ty" will be honored.

Setting "top/bottom" keepout_margin on "movable standard cells" is not recommended. If a standard cell has been set with "top/bottom" keepout_margin, usually the "top/bottom" keepout_margin will be ignored. In the situation that this "movable" standard cell becomes a "fixed cell" from a "movable cell", then some times users may find that there may be some "cell overlapping" report from command check_legality for this standard cell which has "top/bottom" keepout_margin. For example, within some CTS commands, such as compile_clock_tree and optimize_clock_tree, some "movable standard cells" may become "fixed standard cells". as a result, some "additional cell overlapping" report may be printed out by command check_legality. This is one limitation. In order to avoid this problem, users should not set "top/bottom" keepout_margin on the "movable standard cells". Only when the standard cells are fixed, users may set "top/bottom" keepout_margin on these standard cells.

-tracks_per_macro_pin value

Specifies the number of tracks per macro pin. Default value is 0.5 tracks/pin. This will be used to calculate the pin count based derived keepout margin around macro cells. Pin count based derived keepout margin is always hard and -type will be ignored. -all_macros, -macro_masters, and -macro_instances are

not allowed when we specify this option.

-min_padding_per_macro value

Specifies the min padding per macro size. Default value is min grid size which is 0. This will be used during the calculation of pin count based derived keepout margin and -type option will be ignored. -all_macros, -macro_masters and -macro_instances are not allowed when we specify this option. NOTE: During the derived keepout margin calculation of left and right side of the macro *height* of the cell will be taken into consideration and *width* of the cell for top and bottom.

-max_padding_per_macro value

Specifies the max padding per macro size. Default value is -1. This will be used during the calculation of pin count based derived keepout margin and -type option will be ignored. -all_macros, -macro_masters and -macro_instances are not allowed when we specify this option. NOTE: During the derived keepout margin calculation of left and right side of the macro *height* of the cell will be taken into consideration and *width* of the cell for top and bottom.

-all_macros

Keepout margin of specified type will be created on all the macro cells of the design. Internally tool will find out the macro cells in the design.

-macro_masters

Keepout margin of the specified type will be created on all the macro masters or the macro masters of the macro instances specified in the object_list. NOTE: If we specify this option, keepout margin will be created on only macro masters and tool will ignore all the standard cells and standard lib cells specified in the object_list.

-macro_instances

Keepout margin of the specified type will be created on all the macro instances or all the macro instances of the macro masters specified in the object_list. If we specify macro master in the object list tool will find out all the macro instances and creates the keepout margin on macro instances of the macro master. NOTE: If we specify this option, keepout margin will be created on only macro instances and tool will ignore all the standard cells and standard lib cells specified in the object_list.

-north

Specifies that given margin values are with respect to the north orientation of the cell.

object_list

Specifies the list of cells or library cells for which keepout margins are to be created.

DESCRIPTION

The **set_keepout_margin** command creates keepout margins for the specified cells and library cells. The tool will derive the keepout margins for the cells in the following order. 1. keepout margin specified on the cell itself 2. keepout margin specified on the lib cells of the cell 3. Pin count based derived keepout margin values in case of macro cells. NOTE: Please note that tool will derive pin count

based keepout margin values only for the macro cells. If the tool doesnot find the keepout margin on the cell then it looks for the keepout margin specified on the lib cells. In case of macro cells if there is no user specified keepout margin on cells and lib cells then the tool looks for the pin count based derived keepout margin values. A keepout will be created for a cell or library cells whether or not the cell has a FIXED_PLACEMENT attribute. Keepout margins can be set on ILM cells and references also.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example runs the **set_keepout_margin** command.

```
prompt> set_keepout_margin -type hard \  
        -outer {10 10 10 10} MY_CELL  
prompt> set_keepout_margin -type hard \  
        -outer {10 10 10 10} MY_LIB_CELL
```

This sets a keepout margin on the ILM reference blk.

```
prompt> set_keepout_margin -type hard \  
        -outer {10 10 10 10} [get_ilms -reference blk]
```

This sets keepout margin on all the macro cells in the design.

```
prompt> set_keepout_margin -type hard -all_macros \  
        -outer {10 10 10 10}
```

This sets a keepout margin on the basis of pin count of the macro.

```
prompt> set_keepout_margin -tracks_per_macro .6  
        -min_padding_per_macro .1 -max_padding_per_macro 0.2
```

This sets a keepout margin on all the ILM instances in the design.

```
prompt> set_keepout_margin -type hard \  
        -outer {10 10 10 10} [get_ilms]
```

SEE ALSO

get_attribute(2)
remove_keepout_margin(2)
report_keepout_margin(2)

set_latency_adjustment_options

Performs setting options for io latency adjustment.

SYNTAX

```
int set_latency_adjustment_options  
[-from_clock clock_name]  
[-to_clock collection_or_string_list]  
[-exclude_clock collection_or_string_list]  
[-latency float]
```

ARGUMENTS

- from_clock *clock_name***
Specify the clock from which the insertion delay (median arrival time among the sinks) will be applied to the real and/or virtual clocks in the **-to_clock** option.
- to_clock *collection_or_string_list***
Specify the list of real and /or virtual clocks whose latencies are to be updated based on either the latency from the **-from_clock** or the value from **-latency**.
- exclude_clock *collection_or_string_list***
Specify the list of clocks to be excluded from being updated.
- latency *float***
Specify the value to be used when clock latencies are being updated.

DESCRIPTION

The **set_latency_adjustment_options** command is used to set the stage for io latency adjustment after clock tree synthesis.

There are two basic modes of usage :

1. **set_latency_adjustment_options -from_clock clkA -to_clock {clkX clkY clkZ**
2. **set_latency_adjustment_options -latency x.y -to_clock {clkM clkN clkJ**

set_left_right_filler_rule

Sets left-right filler cell insertion rules.

SYNTAX

```
status set_left_right_filler_rule  
-left left_filler  
-right right_filler  
-lib_cell cell_list  
[-follow_stdcell_orientation
```

Data Types

<i>left_filler</i>	string
<i>right_filler</i>	string
<i>cell_list</i>	collection

ARGUMENTS

```
-left left_filler  
    Specify the filler cell that is to be inserted to the immediate left side of  
    the standard cell in the cell_list.  
  
-right right_filler  
    Specify the filler cell that is to be inserted to the immediate right side  
    of the standard cell in the cell_list.  
  
-lib_cell cell_list  
    cell_list must refers to a collection of cell references. The left-right  
    filler cell insertion rule will be applied to the cells specified in this  
    list.
```

DESCRIPTION

Use this command to set the left-right filler rules of the current design. A left (right) filler rule inserts a left (right) filler cell to the immediate left (right) side of a standard cell. A left or right filler cell is often used to avoid certain design rule spacing problems. The insertion of the left (right) filler cell is performed by the `insert_stdcell_filler` command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example specifies a filler rule that inserts "lf" on the left and "rf" on the right of library cells "cellA" and "cellB". The rule will be used in the `insert_stdcell_filler` command.

```
prompt> set_left_right_filler_rule -lib_cell {cellA, cellB} -left "lf" -right "rf"
```

SEE ALSO

```
insert_stdcell_filler(2)  
set_cell_vt_type(2)  
remove_left_right_filler_rule(2)  
report_left_right_filler_rule(2)
```

set_level_shifter

Sets a strategy for level shifting during implementation. This command is supported only in UPF mode.

SYNTAX

```
status set_level_shifter  
level_shifter_name  
-domain domain_name  
[-elements list]  
[-applies_to inputs | outputs | both]  
[-threshold value]  
[-rule low_to_high | high_to_low | both]  
[-location self | parent | fanout | automatic]  
[-no_shift]
```

Data Types

<i>level_shifter_name</i>	string
<i>domain_name</i>	string
<i>list</i>	list
<i>value</i>	float

ARGUMENTS

level_shifter_name
Specifies the level shifter strategy name, which is used only for reporting. This argument is required.

-domain *domain_name*
Specifies the domain for which the strategy is applied. This argument is required.

-elements *list*
Specifies a list of design elements, pins, or ports to which this strategy is applied.

-applies_to inputs | outputs | both
Specifies whether the domain's input ports, output ports, or both are level shifted. The default is both.

-threshold *value*
Specifies the voltage threshold (in volts) for determining when level shifters are required. The default is 0.

-rule low_to_high | high_to_low | both
Specifies which type of level shifters are required. The default is both.

-location self | parent | fanout | automatic
Specifies where the level shifter is placed in the logic hierarchy. The default is automatic.

`-no_shift`

Prevents the insertion of level shifters on the specified ports, pins, and nets when used with the **-elements** option.

DESCRIPTION

The **set_level_shifter** command is used to set a strategy for level shifting during implementation. Level shifters are placed on the signals that have sources and sinks operating at different voltages, as their associated design elements are connected to different supply nets.

If a level shifter strategy is not specified on a power domain, the default level shifter strategy consists of all elements in the power domain, and uses the default strategy settings. Power state table (PST) and operating conditions are considered to determine if level shifters are needed for design elements on the boundaries of power domains.

If **-elements** *list* is specified, the elements must be in the domain specified by **-domain** *domain_name*.

The **-threshold** option defines how large the voltage difference between the driver and the sink needs to be before level shifters are inserted. Normally, this threshold value is determined from the cell libraries. Use the **-threshold** option to override the library values.

The **-rule** value can be `low_to_high`, `high_to_low`, or `both`.

- If `low_to_high` is specified, signals going from a lower voltage to a higher voltage get a level shifter when the voltage difference exceeds that specified by **-threshold**.
- If `high_to_low` is specified, signals going from a higher voltage to a lower voltage get a level shifter when the voltage difference exceeds that specified by **-threshold**.
- If `both` is specified, it is equivalent to having specified both rules in the strategy.

The **-location** option defines where the level shifter cells are placed in the logic hierarchy. All necessary supplies need to be available in the specified location. The option values are as follows:

- `self` specifies that the level shifter cell is placed inside the model or cell being shifted.
- `parent` specifies that the level shifter cell is placed in the parent of the cell or model being shifted.
- `fanout` specifies that the level shifter occurs at all fanout locations (sinks) of the port being shifted.
- `automatic` specifies that the implementation tool chooses the appropriate locations.

Note that this command does not apply to inout ports. Also, it is an error if the specified location is not within the logic design starting at the design root.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets the level shifter strategy on PowerDomainZ:

```
prompt> set_level_shifter shift_up -domain PowerDomainZ -applies_to outputs \  
        -threshold 0.02 -rule both
```

SEE ALSO

```
add_port_state(2)  
add_pst_state(2)  
create_pst(2)  
report_pst(2)
```

set_level_shifter_cell

On-The-Fly setting of library level shifter cell.

SYNTAX

```
Boolean set_level_shifter_cell  
cell_name  
[-cell_type cell_type]  
[-cell_input_voltage_range {lower_range upper_range}]  
[-cell_output_voltage_range {lower_range upper_range}]  
[-std_cell_main_rail_pg_pin pg_pin_name]  
[-data_pin data_pin_name]  
[-input_voltage_range {lower_range upper_range}]  
[-output_voltage_range {lower_range upper_range}]  
[-input_signal_level signal_level]  
[-enable_pin enable_pin_name]  
[-enable_signal_level signal_level]  
[-output_signal_level signal_level]  
[-reset]
```

Data Types

<i>cell_name</i>	string
<i>lower_range</i>	float
<i>upper_range</i>	float
<i>pg_pin_name</i>	string
<i>data_pin_name</i>	string
<i>signal_level</i>	string
<i>enable_pin_name</i>	string

ARGUMENTS

cell_name
Specify the library cell as level_shifter cell.

-reset
reset the level shifter cell attributes.

DESCRIPTION

The **set_level_shifter_cell** command set all the cells with the same name specified as level shifter cell.

If -reset is issued, the level_shifter cell attributes will be cleaned up.

If -reset is combined with other options, reset happens first, other options will be set on the cell after reset.

EXAMPLES

```
prompt> set_level_shifter_cell LVLLH* -cell_type LH -std_cell_main_rail_pg_pin VDD -  
enable_pin_name EN -input_signal_level VDDL -enable_signal_level VDDH -  
output_signal_level VDDH -cell_input_voltage_range {0.7 1.4} -  
cell_output_voltage_range {0.7 1.4}
```

SEE ALSO

```
report_lib(2)  
check_library(2)
```

set_level_shifter_strategy

Sets the type of strategy to use for adjusting the voltage levels in the design.

SYNTAX

```
int set_level_shifter_strategy  
-rule all | low_to_high | high_to_low  
[-location inside | outside | source | sink]
```

ARGUMENTS

-rule all | low_to_high | high_to_low
Specifies to adjust the voltage levels when the operating voltage levels between a source and sink are different.
Using **-rule low_to_high** adjusts the voltage levels when a source at a higher voltage drives a sink at a lower voltage.
Using **-rule high_to_low** adjusts the voltage levels when a source at a lower voltage drives a sink at a higher voltage.
The **all** value is the default strategy.

-location inside | outside | source | sink
This option is only effective in power state table based level shifter insertion. It specifies the location for a level shifter when it is inserted on a boundary of two domains.
Using **-location inside** requires to place the level shifter in the inside domain of the two domains on the boundary.
Using **-location outside** requires to place the level shifter in the outside domain of the two domains on the boundary.
Using **-location source** requires to place the level shifter in the source domain of the two domains on the boundary.
Using **-location sink** requires to place the level shifter in the sink domain of the two domains on the boundary.
By default, the locations of inserted level shifters are automatically determined to avoid main power violation.

DESCRIPTION

This command specifies the strategy for level shifter insertion. The strategy defines the the level shifters insertion is to take place in the design.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets the level shifter strategy to the **high_to_low** option:

```
prompt> set_level_shifter_strategy -rule high_to_low
```


SEE ALSO

`set_level_shifter_threshold(2)`

set_level_shifter_threshold

Sets the minimum threshold beyond which the voltage adjustment is required.

SYNTAX

```
int set_level_shifter_threshold  
-voltage volt  
-percent diff
```

Data Types

```
volt      float  
diff      float
```

ARGUMENTS

```
-voltage volt  
    The absolute difference between the source and sink voltages.  
  
-percent diff  
    The percentage by which the source and sink voltages must differ. The  
    percentage is determined as follows:  
  
     $\text{abs}(\text{driver}(v) - \text{load}(v)) / \text{driver}(v) * 100$ 
```

DESCRIPTION

This command specifies the minimum threshold value for the voltage difference between a source and a sink. Voltage values above the minimum threshold are adjusted. The threshold can be specified as the absolute difference in voltages or a percentage difference or both. If the **-voltage** and the **-percent** options are both specified, then either of the two thresholds be satisfied.

The default threshold voltage and percentage is zero.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets the level shifter threshold value to the absolute difference of 0.5:

```
prompt> set_level_shifter_threshold -voltage 0.5
```

The following example sets the threshold value 5 percent:

```
set_level_shifter_threshold  
2516
```

```
prompt> set_level_shifter_threshold -percent 5
```

The following example sets the threshold value to 0.5 difference and 5 percent, so if either value is reached, the adjustment occurs:

```
prompt> set_level_shifter_threshold -voltage 0.5 -percent 5
```

SEE ALSO

`set_level_shifter_strategy(2)`

set_lib_attribute

Sets the value of an attribute on a library object.

SYNTAX

```
list set_lib_attribute  
object_list  
attribute_name  
attribute_value
```

Data Types

<i>object_list</i>	list
<i>attribute_name</i>	string
<i>attribute_value</i>	datatype of attribute_name

ARGUMENTS

object_list
A list of the library objects on which the attribute is to be set.

attribute_name
The name of the attribute to set.

attribute_value
The value of the attribute. The datatype must be the same as that of the attribute.

DESCRIPTION

Sets the value of an attribute on a library object. Only the attribute values of a small set of library objects can be set using this command. For the complete list of such library objects and their related "updatable" attributes, please refer to the user manual.

This command returns the list of objects that have the specified attribute value set. A returned empty list means that the command was not successful.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

This command is used to find the cell "INV" and set its area value to 1.0:

```
prompt> set_lib_attribute sample/INV area 1.0  
Performing set_lib_attribute on library object 'sample/INV'.
```

```
{sample/INV}
```

In this example, an error is issued because the cell "INV1" is not found:

```
prompt> set_attribute sample/INV1 area 1.0  
Error: The 'INV1' object does not exist in the 'sample' technology library  
. (UIL-54)  
{}
```

In the following command, the "lib_udg_attr" value for the user-defined group "lib_udg1" in the library "sample" is set to "test":

```
prompt> set_lib_attribute sample/lib_udg1 lib_udg_attr test  
Performing set_lib_attribute on library object 'sample/lib_udg1'.  
{sample/lib_udg1}
```

In the following command, the area values for gates "G1" and "G2" in the library "tech_lib" are both set to 2.0:

```
prompt> get_lib_attribute {sample/G1, sample/G2} area  
Performing set_lib_attribute on library object 'sample/G1'.  
Performing set_lib_attribute on library object 'sample/G2'.  
{"sample/G1", "sample/G2"}
```

SEE ALSO

`get_attribute(2)`

set_lib_cell_spacing_label

Sets an intercell-spacing constraint label on a reference cell.

SYNTAX

```
status set_lib_cell_spacing_label  
-names {list_of_label_names}  
[-left_lib_cells {lib_cell_collection}]  
[-right_lib_cells {lib_cell_collection}]
```

Data Types

<i>list_of_label_names</i>	list of strings
<i>lib_cell_collection</i>	collection

ARGUMENTS

```
-names {list_of_label_names}  
    Lists intercell constraint labels to assign to one or more reference cell.  
  
-left_lib_cells {lib_cell_collection}  
    Specifies collection of reference cells that have the given labels assigned  
    to the left side of the cell in the "North" orientation.  
  
-right_lib_cells {lib_cell_collection}  
    Specifies collection of reference cells that have the given labels assigned  
    to the right side of the cell in the "North" orientation.
```

DESCRIPTION

This command assigns labels to a collection of reference (library) cells in order to enable the legalization tool to obey intercell spacing constraints. A label is a string assigned by the user to the left and/or right of a standard cell. This command works in concert with the command `set_spacing_label_rule` that defines the range of spacing between two adjacent labels that is illegal. All spacing rules specified by `set_spacing_label_rule` commands need to be satisfied to produce a legal placement. With intercell spacing rule specified, it is no longer assumed that any two standard cells can be placed next to each other with a certain spacing.

The labels and spacing rules can be used to express spacing requirements that originate from mask rules and the layout of standard cells. The labels and spacing rules are stored in the library and will be applied to all designs of the library once specified.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

In the following example, reference cells AND* are assigned label "X" on the left of the cell in the "North" orientation, and reference cells NAND* are assigned label "X" on the right.

```
prompt> set_lib_cell_spacing_label -names {X}\  
-left_lib_cells {AND*}\  
-right_lib_cells {NAND*}
```

In the following example, reference cell OR2X4 is assigned labels "Y" and "Z" on the right side.

```
prompt> set_lib_cell_spacing_label\  
-names {Y Z} -right_lib_cells {OR2X4}
```

SEE ALSO

```
remove_all_spacing_rules(2)  
report_spacing_rules(2)  
set_spacing_label_rule(2)
```

set_load

Sets the **load** attribute to a specified value on specified ports and nets.

SYNTAX

```
status set_load  
value  
objects  
[-subtract_pin_load]  
[-min]  
[-max]  
[[-pin_load] [-wire_load]]
```

Data Types

```
value      float  
objects    list
```

ARGUMENTS

value

Specifies the value with which to set the **load** attribute on the ports and nets contained in *objects*. The *value* must be expressed in units consistent with the technology library used during optimization. For example, if the technology library specifies load values in picofarads, *value* must also be expressed in picofarads.

objects

Specifies a list of ports and nets in the current design whose loads are to be set.

-subtract_pin_load

Indicates that the current pin capacitances of the net are to be subtracted from *value* before the net load value is set. If the resulting net load value is negative, it is set to 0. This option sets the **subtract_pin_load** attribute on *objects*. With this attribute set, the total load computed on these nets during **update_timing** does not include pin loads. Use this option if *value* includes pins and ports capacitances. The **subtract_pin_load** attribute is not placed on ports.

-min

Indicates that the load value is to be used for minimum delay analysis. You cannot use the legacy option **-fanout_number** with **-min**, because different fanout numbers for min and max are not supported. If no minimum load value is specified, the maximum value is used.

-max

Indicates that the load value is to be used for maximum delay analysis. If no value is specified for maximum analysis on a net, and a value has been specified for minimum analysis, the minimum value is ignored. (You cannot annotate only a minimum value on a net.)

`-pin_load -wire_load`

Indicates whether the specified *value* on the port is to be treated as a pin load, as a wire load, or as both. These options can be used only with ports; an error message is displayed if the *objects* contains any nets. If you do not specify either **-pin_load** or **-wire_load**, **-pin_load** is used as the default. You can use both arguments together, in which case the load value is set as both a pin load and as a wire load on the specified ports.

DESCRIPTION

This command sets the **load** attribute on ports and nets in the current design. If the current design is hierarchical, it must have been linked with the **link** command. The total load on a net is the sum of all of pin loads, port loads, and wire loads associated with that net. The specified *value* overrides the internally-estimated net load value.

You also can use the **set_load** command for nets at lower levels of the design hierarchy. These nets are specified as BLOCK1/BLOCK2/NET_NAME.

If you use the **-wire_load** option, *value* is set as a wire load on the specified port. However, the value is actually counted as part of the total wire load and not as part of the pin or port load.

To view load values on ports, use the **report_port** command. To view load values on nets, use the **report_net** or **report_internal_loads** command.

To reset a load value, use the **remove_attribute** command. To reset all annotated load values in a design, use the **reset_design** command.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example sets a load of 2 units to the port named `the_answer`:

```
prompt> set_load 2 the_answer
```

The following example sets a load of 2.5 units (the estimated wire load) plus the load of 3 inverter input pins from the `tech_lib` library on all output ports. The user-defined `dc_shell` variable **port_load** is used to store this load value.

```
prompt> link -all
```

```
prompt> port_load = 2.5 + 3 * load_of(tech_lib/IV/A)
```

```
prompt> set_load port_load all_outputs()
```

The following example sets a load of pin Z of IV from the `tech_lib` library to the

input_1 and input_2 ports, using the **load_of** command:

```
prompt> set_load load_of(tech_lib/IV/Z) {input_1 input_2}
```

In the following example, a load of 3 is set on the U1/U2/NET3 net. The wire capacitance is set to 3. (Total net capacitance is 3 plus the sum of the pin and port capacitances.)

```
prompt> set_load 3 U1/U2/NET3
```

In the following example, a total net capacitance (wire capacitance plus pin capacitances) of 3 is set on the U1/U2/NET3 net. If the pin and port capacitances equal 2, the wire capacitance is annotated with 1. If the pin and port capacitances are 3 or more, the wire capacitance is annotated with 0.

```
prompt> set_load -subtract_pin_load 3 U1/U2/NET3
```

The following example sets a wire load of 5 units on the port named the_answer:

```
prompt> set_load -wire_load 5 the_answer
```

The following sequence of commands describes the external fanout of an output port:

```
prompt> set_load -fanout_number 5 [get_ports O1]
```

```
prompt> set_wire_load_model -port_list [get_ports O1] "default_wl"
```

The following commands remove the back-annotated load on a port and on a net:

```
prompt> remove_attribute [get_ports the_answer] load
```

```
prompt> remove_attribute [get_ports the_answer] "wire_capacitance"
```

```
prompt> remove_attribute [get_ports O1] "fanout_number"
```

```
prompt> remove_attribute [get_nets U1/U2/NET3] load
```

SEE ALSO

all_outputs(2)
current_design(2)
load_of(2)
remove_attribute(2)
report_internal_loads(2)
report_net(2)
report_port(2)
reset_design(2)
set_drive(2)
target_library(3)

set_local_link_library

Sets the **local_link_library** attribute to specified files and libraries on the current design.

SYNTAX

```
int set_local_link_library  
local_link_library
```

Data Types

```
local_link_library      string
```

ARGUMENTS

```
local_link_library  
    Specifies a list of files with which the local_link_library attribute is to  
    be set on the current design. The local_link_library files are not loaded or  
    checked until the link_library is used.
```

DESCRIPTION

Sets the **local_link_library** attribute to *local_link_library* on the current design. The *local_link_library* is a list of design files and libraries that is added to the beginning of the **link_library** whenever a link operation is performed. Files in the *local_link_library* are searched first.

The **compile**, **insert_dft**, and **translate** commands set this attribute to the same value as the **target_library** variable.

Use **report_design** or **get_attribute** to identify the **local_link_library** for a design.

To remove this attribute, use the **remove_attribute** or **reset_design** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command sets the **local_link_library** "tech_lib.db" on the current design "my_design":

```
prompt> set_local_link_library tech_lib.db
```

```
Setting local link library 'tech_lib.db' on design 'my_design'
```

SEE ALSO

`current_design(2)`
`link(2)`
`remove_attribute(2)`
`reset_design(2)`
`link_library(3)`
`target_library(3)`

set_logic_dc

Specifies one or more input ports in the current design that are to be driven by don't care. The **set_logic_one** and **set_logic_zero** commands are used the same way as this command.

SYNTAX

```
int set_logic_dc  
port_list
```

Data Types

port_list list

ARGUMENTS

port_list

A list of input port names in the current design that are to be driven by don't care. You can use only one of **set_logic_one**, **set_logic_zero**, or **set_logic_dc** on any given port. If you specify more than one port, you must include them in braces ({}).

DESCRIPTION

Assigns a **driven_by_dont-care** attribute to the ports in *port_list*; a given port can have only one of the **driven_by_logic_one**, **driven_by_logic_zero**, and **driven_by_dont-care** attributes. This information is used by **compile** to create smaller designs by eliminating logic that is tied to a specific value and therefore might not need to be maintained during optimization. After optimization, a port connected to logic one, logic zero, or don't care usually does not drive anything inside the optimized design.

Note: This command cannot be used on output ports. To specify output ports as unconnected, use the **set_unconnected** command.

When a **set_logic_dc** command is used on an input port, **compile** is given the freedom to assign any signal to that input (including, but not limited to, zero and one). The meaning of this command is that the outputs of the design are significant only when the non-don't care inputs completely determine all outputs, independent of the don't care inputs. This information is used to optimize the design.

For example, suppose the design is a 2:1 multiplexer with inputs S, A, B, and output Z. The function computed is as follows:

$$Z = S*A + S'*B$$

If you issue the following command,

```
prompt> set_logic_dc B
```

the implication is that the value of Z is significant only when S=1, and is a don't

care when S=0. The resulting simplification done by **compile** represents the reduced logic as a wire, and the function is as follows:

```
Z = A
```

Use the **remove_attribute** command to remove these attributes.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following commands are issued to drive input ports "A" and "B" by logic 0, input port "C" by logic 1, and input port "D" by don't care:

```
prompt> set_logic_zero {A B}
prompt> set_logic_one C
prompt> set_logic_dc D
```

SEE ALSO

```
get_attribute(2)
remove_attribute(2)
reset_design(2)
set_logic_one(2)
set_logic_zero(2)
set_unconnected(2)
```

set_logic_one

Specifies one or more input ports in the current design that are to be driven by logic one. The **set_logic_zero** and **set_logic_dc** commands are used the same way as this command.

SYNTAX

```
int set_logic_one  
port_list
```

Data Types

port_list list

ARGUMENTS

port_list

A list of input port names in the current design that are to be driven by logic one. You can use only one of **set_logic_one**, **set_logic_zero**, or **set_logic_dc** on any given port. If you specify more than one port, you must include them in braces ({}).

DESCRIPTION

Assigns a **driven_by_logic_one** attribute to the ports in *port_list*; a given port can have only one of the **driven_by_logic_one**, **driven_by_logic_zero**, and **driven_by_dont-care** attributes. This information is used by **compile** to create smaller designs by eliminating logic that is tied to a specific value and therefore might not need to be maintained during optimization. After optimization, a port connected to logic one, logic zero, or don't care usually does not drive anything inside the optimized design.

Note: This command cannot be used on output ports. To specify output ports as unconnected, use the **set_unconnected** command.

Use the **remove_attribute** command to remove these attributes.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following commands are issued to drive input ports "A" and "B" by logic 0, input port "C" by logic 1, and input port "D" by don't care:

```
prompt> set_logic_zero {A B}  
prompt> set_logic_one C
```

```
prompt> set_logic_dc D
```

SEE ALSO

```
get_attribute(2)  
remove_attribute(2)  
reset_design(2)  
set_logic_dc(2)  
set_logic_zero(2)  
set_unconnected(2)
```


set_logic_zero

Specifies one or more input ports in the current design that are to be driven by logic zero. The **set_logic_one** and **set_logic_dc** commands are used the same way as this command.

SYNTAX

```
int set_logic_zero  
port_list
```

Data Types

```
port_list      list
```

ARGUMENTS

port_list

A list of input port names in the current design that are to be driven by logic zero. You can use only one of **set_logic_one**, **set_logic_zero**, or **set_logic_dc** on any given port. If you specify more than one port, you must include them in braces ({}).

DESCRIPTION

Assigns a **driven_by_logic_zero** attribute to the ports in *port_list*; a given port can have only one of the **driven_by_logic_one**, **driven_by_logic_zero**, and **driven_by_dont-care** attributes. This information is used by **compile** to create smaller designs by eliminating logic that is tied to a specific value and therefore might not need to be maintained during optimization. After optimization, a port connected to logic one, logic zero, or don't care usually does not drive anything inside the optimized design.

Note: This command cannot be used on output ports. To specify output ports as unconnected, use the **set_unconnected** command.

Use the **remove_attribute** command to remove these attributes.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following commands are issued to drive input ports "A" and "B" by logic 0, input port "C" by logic 1, and input port "D" by don't care:

```
prompt> set_logic_zero {A B}  
prompt> set_logic_one C
```

```
prompt> set_logic_dc D
```

SEE ALSO

```
get_attribute(2)  
remove_attribute(2)  
reset_design(2)  
set_logic_dc(2)  
set_logic_one(2)  
set_unconnected(2)
```

set_macro_cell_bound_spot

Updates the macro cell move bound or group bound by specifying the exact location (spot) for the bound inside the macro cell.

SYNTAX

```
status set_macro_cell_bound_spot  
macro_cell_object  
-coordinates {spot_x spot_y}
```

Data Types

<i>macro_cell_object</i>	string
<i>spot_x</i>	float
<i>spot_y</i>	float

ARGUMENTS

macro_cell_object
Specifies the macro cell that is related to a move bound or group bound.

-coordinates {*spot_x spot_y*}
Uses {*spot_x spot_y*} to specify the x and y coordinates of the location for the bound inside the macro cell. Note that {*spot_x spot_y*} are related to the physical library cell of the macro cell (that is, they are based on the library cell origin without cell rotation). The move bound or group bound related to this macro cell tries to bound the other standard cells to this exact location in the macro cell.

DESCRIPTION

This command specifies a location (or spot) inside the macro cell so that tools bound the standard cell to this exact point in the macro cell. Without this capability, the placer places the standard cell at any location in the macro cell. This can cause poor quality of results (QoR) if the standard cell is placed on the wrong side of the macro cell.

You can also use the **create_bounds** or **update_bounds** command to directly specify the macro cell pin so that the first pin geometry in the macro cell is assigned as the location in it to which standard cells are bound.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows how to bound the standard cell Top/U1 to the macro cell Top/RAM16x16 at the exact location {10.5 20.2}, related to its physical library cell

origin.

```
prompt> create_bounds\  
-coord {0 0 100 100} {Top/U1 Top/RAM16x16  
prompt> set_macro_cell_bound_spot\  
-coord {10.5 20.2} [get_cell "Top/RAM16x16"]fp
```

The following example shows the use of the command given the description below. If the CLK pin in the RAM16x16 macro cell is the one that must bound with the standard cell U1, the following command directly bounds the CLK pin inside the Top/RAM16x16 cell to the Top/U1 standard cell. However, the placer picks the first CLK port geometry in the physical library cell of the macro and assigns it as the location (spot) in the macro cell. It might or might not be {10.5 20.2}, depending on how many geometries for the CLK port are in the physical library cell.

```
prompt> create_bounds\  
-coord {0 0 100 100} {Top/U1 Top/RAM16x16/CLK
```

SEE ALSO

```
create_bounds(2)  
get_bounds(2)  
update_bounds(2)  
report_bounds(2)
```

set_max_area

Sets the **max_area** attribute to a specified value on the current design.

SYNTAX

```
int set_max_area  
[-ignore_tns]  
area_value
```

Data Types

area_value float

ARGUMENTS

-ignore_tns
Specifies that the area is prioritized above total negative slack (TNS).

area_value
Specifies the value to which the **max_area** attribute is to be set. The units of area must be consistent with the units in the technology library used during optimization.

DESCRIPTION

This command sets the **max_area** attribute to *area* on the current design. The **max_area** attribute represents the target area of the design and is used by the **compile** command to calculate area cost of the design.

By default, **compile** prioritizes total negative slack (TNS) above area. This means that **compile** does not create new delay violations or worsen existing delay violations on a path that has negative delay slack in order to improve area.

When the **-ignore_tns** option is used, the **ignore_tns** attribute is set on the design. When this attribute is set, **compile** prioritizes area above TNS. This means that **compile** may increase delay violations at an endpoint in order to improve area, as long as the new delay violation is smaller than the delay violation on the endpoint of the most critical path in the same path group.

If you specify **max_area** more than once on a design, the most recent *area* is used and the earlier values are removed.

If the **set_max_area** command is used without the **-ignore_tns** option, then the **ignore_tns** attribute (if any) on the design is deleted.

To undo **set_max_area**, use the **remove_attribute** or **reset_design** command.

Use the **report_area** command to get a detailed summary of area of the design. Use the **report_constraint** command to show area cost of the design.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets the target area for the current design to 0.0 (zero). This causes **compile** to create a design that has been optimized for the smallest possible size.

```
prompt> set_max_area 0.0
```

SEE ALSO

```
current_design(2)  
remove_attribute(2)  
report_area(2)  
report_constraint(2)  
reset_design(2)
```

set_max_capacitance

Sets the **max_capacitance** attribute to a specified value on the specified input ports and designs.

SYNTAX

```
int set_max_capacitance  
  capacitance_value  
  object_list
```

ARGUMENTS

capacitance_value

Specifies a value to which the **max_capacitance** attribute is to be set. *capacitance_value* must be expressed in units consistent with the technology library used during optimization. For example, if the library specifies capacitance values in picofarads, then *capacitance_value* must also be expressed in picofarads.

object_list

Specifies a list of names of input ports and/or designs on which the **max_capacitance** attribute is to be set.

DESCRIPTION

Sets the **max_capacitance** attribute to *capacitance_value* on the specified input ports or designs. **compile** attempts to ensure that the capacitance value for a net is less than *capacitance_value*. The maximum capacitance value for a net is defined as the least of the maximum capacitance values of the cell pins and design ports on that net.

In cases where both a global maximum capacitance value is set on a design and a local value is set on a port, **compile** attempts to meet the smaller (more restrictive) value.

If **max_capacitance** attributes are already specified in a technology library (implicit constraints), **compile** automatically tries to meet them.

By default, a port or design has no **max_capacitance** constraint.

Please note that by this command **max_capacitance** attribute can not be set on output or bidirectional port.

max_capacitance, along with **max_fanout** and **max_transition**, is a *design rule constraint*; **max_delay** and **max_area** are optimization constraints. Design rule constraints reflect those technology-specific restrictions that *must* be met for a design to function correctly, while optimization constraints reflect goals and restrictions that are desirable, but not crucial for the operation of a design. Design Compiler attempts to meet all constraints placed on a design, but gives priority to design rule constraints in the optimization process. Therefore, **compile** gives preference to **max_capacitance** and other design rule constraints, even if they

adversely affect optimization constraints on a design.

To get information about optimization and design rule constraints, use **report_constraint**; to get information on the current port settings, use **report_port**. To remove the **max_capacitance** attribute from a port or a design, use **remove_attribute**.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example sets a maximum capacitance value of 2.0 units on the port named "late_riser":

```
prompt> set_max_capacitance 2.0 late_riser
```

The following example specifies a maximum capacitance value of 2.0 units for the design "TEST":

```
prompt> set_max_capacitance 2.0 TEST
```

SEE ALSO

- all_inputs(2)
- all_outputs(2)
- characterize(2)
- current_design(2)
- remove_attribute(2)
- report_constraint(2)
- report_port(2)
- reset_design(2)
- set_min_capacitance(2)

set_max_delay

Specifies a maximum delay target for paths in the current design.

SYNTAX

```
int set_max_delay
delay_value
[-rise | -fall]
[-from from_list
  | -rise_from rise_from_list
  | -fall_from fall_from_list]
[-through through_list]
[-rise_through rise_through_list]
[-fall_through fall_through_list]
[-to to_list
  | -rise_to rise_to_list
  | -fall_to fall_to_list]
[-group_path group_name]
[-reset_path]
```

Data Types

<i>delay_value</i>	float
<i>from_list</i>	list
<i>rise_from_list</i>	list
<i>fall_from_list</i>	list
<i>through_list</i>	list
<i>rise_through_list</i>	list
<i>fall_through_list</i>	list
<i>to_list</i>	list
<i>rise_to_list</i>	list
<i>fall_to_list</i>	list
<i>group_name</i>	string

ARGUMENTS

delay_value
Specifies the value of the desired maximum delay for paths between start and end points. You must express *delay_value* in the same units as the technology library used during optimization. If a path startpoint is on a sequential device, clock skew is included in the computed delay. If a path startpoint has an input delay specified, that delay value is added to the path delay. If a path endpoint is on a sequential device, clock skew and library setup time are included in the computed delay. If the endpoint has an output delay specified, that delay is added into the path delay.

-rise | -fall
Specifies whether endpoint rising or falling delays are constrained. If you don't specify either, both rising and falling delays are constrained.

-from *from_list*
Specifies a list of path startpoints (port, pin, clock, or cell names) of the

current design. If you specify a clock, all path startpoints related to that clock are affected. If you specify a cell name, one path startpoint on that cell is affected. All paths from these startpoints to the endpoints in the *to_list* are constrained to *delay_value*. If you don't specify *to_list*, all paths from *from_list* are affected. This list cannot include output ports. If you include more than one object, you must enclose the objects in quotation marks (") or braces ({}).

-rise_from *rise_from_list*

Same as the **-from** option, except that the path must rise from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by rising edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of **-from**, **-rise_from**, and **-fall_from**.

-fall_from *fall_from_list*

Same as the **-from** option, except that the path must fall from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of **-from**, **-rise_from**, and **-fall_from**.

-through *through_list*

Determines a list of path throughpoints (port, pin, or leaf cell names) of the current design. The max delay value applies only to paths that pass through one of the points in the *through_list*. If you include more than one object, you must enclose the objects in quotation marks (") or braces ({}). If you specify the **-through** option multiple times, the max delay values apply to paths that pass through a member of each *through_list* in the order in which the lists were given. In other words, the path must first pass through a member of the first *through_list*, then through a member of the second list, and so on, for every through list specified. If you use the **-through** option in combination with the **-from** or **-to** option, max delay applies only if the **-from** or **-to** conditions are satisfied and the **-through** conditions are satisfied.

-rise_through *rise_through_list*

Same as the **-through** option, but applies only to paths with a rising transition at the specified objects. You can specify **-rise_through** more than once in a single command invocation as with the **-through** option.

-fall_through *fall_through_list*

Same as the **-through** option, but applies only to paths with a falling transition at the specified objects. You can specify **-fall_through** more than once in a single command invocation as with the **-through** option.

-to *to_list*

Specifies a list of path endpoints (port, clock, cell, or pin names) of the current design. All paths to the endpoints in the *to_list* are constrained to *delay_value*. If you don't specify a *from_list*, all paths to *to_list* are affected. This list cannot include input ports. If you include more than one object, you must enclose the objects in quotation marks (") or braces ({}). If you specify a cell, one path endpoint on that cell is affected. If you specify a clock, all path endpoints related to that clock are affected.

-rise_to *rise_to_list*

Same as the **-to** option, but applies only to paths rising at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths captured by rising edge of the clock at clock source, taking into account any logical inversions along the clock path. You can use only one of **-to**, **-rise_to**, and **-fall_to**.

-fall_to *fall_to_list*

Same as the **-to** option, but applies only to paths falling at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of **-to**, **-rise_to**, and **-fall_to**.

-group_path *group_name*

Specifies the name of the group to which the paths are to be added. If the group does not already exist, it is created. This is equivalent to separately specifying the **group_path** command with the **-name** *group_name* **-from** *from_list* **-to** *to_list* options in addition to specifying the **set_max_delay** command. If you do not specify this option, the existing path grouping is not changed.

-reset_path

Removes existing point-to-point exception information on the specified paths. Only information of the same rise/fall setup/hold type is reset. This is equivalent to using the **reset_path** command with similar options before executing the **set_max_delay** command.

DESCRIPTION

Specifies the desired maximum delay for paths in the current design. This command specifies that the maximum path length for any startpoint in *from_list* to any endpoint in *to_list* must be less than *delay_value*.

Individual maximum delay targets are automatically derived from clock waveforms and port input or output delays. For more information, see the **create_clock**, **set_input_delay**, and **set_output_delay** man pages.

The optimization cost of the design depends on how path groups have been specified. For more information, see the **group_path** man page.

The **set_max_delay** command is a point-to-point timing exception command; that is, it overrides the default single-cycle timing relationship for one or more timing paths. Other point-to-point timing exception commands include the **set_multicycle_path**, **set_min_delay**, and **set_false_path** commands.

If a path satisfies multiple timing exceptions, the following rules are applied to determine which exceptions take effect. Rules referring to **-from** apply equally to **-rise_from** and **-fall_from**, and similarly for the rise and fall options of **-through** and **-to**.

1. Two **group_path** commands can conflict with each other. But a **group_path** exception by itself does not conflict with another type of exception. Thus, the remaining rules apply for two **group_path** exceptions or two non-**group_path** exceptions.

2. If both exceptions are **set_false_paths**, no conflict occurs.
3. If one exception is a **set_max_delay** and the other is a **set_min_delay**, no conflict occurs.
4. If one exception is a **set_multicycle_path -hold** and the other is **set_multicycle_path -setup**, no conflict occurs.
5. If one exception is a **set_false_path** and the other is not, the **set_false_path** takes precedence.
6. If one exception is a **set_max_delay** and the other is not, the **set_max_delay** takes precedence.
7. If one exception is a **set_min_delay** and the other is not, the **set_min_delay** takes precedence.
8. If one exception has a -from pin or -from cell and the other does not, the former takes precedence.
9. If one exception has a -to pin or -to cell and the other does not, the former takes precedence.
10. If one exception has any -through points and the other does not, the former takes precedence.
11. If one exception has a -from clock and the other does not, the former takes precedence.
12. If one exception has a -to clock and the other does not, the former takes precedence.
13. The exception with the more restrictive constraint then takes precedence. For **set_max_delay** and **set_multicycle_path -setup**, this is the constraint with the lower value. For **set_min_delay** and **set_multicycle_path -hold**, it is the constraint with the higher value.

The value of a **max_rise_delay** attribute cannot be less than that of a **min_rise_delay** attribute on the same path (and similarly for fall attributes). If this occurs, the old attribute is removed.

Note: Specifying a **min_delay** or **max_delay** to a pin that is not a path endpoint places an implicit **dont_touch** attribute on that cell.

The **-group_path** option modifies the path grouping. Path grouping affects the maximum delay cost function. The worst violator within each group adds to the cost. For optimization, grouping some paths separately can improve their delay cost, but it might also increase design area, and compile time.

Use the **report_timing_requirements** command to list the **max_delay**, **min_delay**, **multicycle_path**, and **false_path** information for the design. Use **report_path_group** to list the path groups which are defined.

To undo **set_max_delay**, use **reset_path**.

To modify paths grouped with the **-group_path** option, use the **group_path** command to place the paths in another group or the default group.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example shows how to optimize the design so that any delay path to a port named Y is less than 10 units.

```
prompt> set_max_delay 10.0 -to {Y}
```

This example shows how to specify that all paths from ff1a or ff1b that pass through cell u1 and end at ff2e must be less than 15.0 units.

```
prompt> set_max_delay 15.0 -from {ff1a ff1b} -through {u1} -to {ff2e}
```

example shows how to specify that all paths to endpoints clocked by PHI2 must be less than 8.5 units.

```
prompt> set_max_delay 8.5 -to [get_clocks PHI2]
```

This example shows how to set a requirement that all paths leading to ports named busA[*] have a delay of less than 5.0 and are included in the path group named busA.

```
prompt> set_max_delay 5.0 -to "busA[*]" -group_path "busA"
```

This example shows how to set a maximum delay requirement of 3.0 for all paths that first pass through either u1/Z or u2/Z then pass through u5/Z or u6/Z.

```
prompt> set_max_delay 3.0 -through {u1/Z u2/Z} -through {u5/Z u6/Z}
```

This example specifies that all timing paths from ff1/CP to ff2/D that rise through one or more of {U1/Z U2/Z} and fall through one or more of {U3/Z U4/C} must have delays less than 8.0 units.

```
prompt> set_max_delay 8.0 -from {ff1/CP} -rise_through {U1/Z U2/Z} -  
fall_through {U3/Z U4/C} -to {ff2/D}
```

SEE ALSO

```
create_clock(2)  
current_design(2)  
group_path(2)  
report_constraint(2)  
report_path_group(2)  
reset_design(2)  
reset_path(2)  
set_false_path(2)  
set_input_delay(2)
```

```
set_min_delay(2)
set_multicycle_path(2)
set_output_delay(2)
```

set_max_dynamic_power

Sets the target dynamic power for the current design by setting the **max_dynamic_power** attribute to a specified value.

SYNTAX

```
int set_max_dynamic_power  
dynamic_power  
[GW | MW | KW | W | mW | uW | nW | pW | fW | aW]
```

Data Types

dynamic_power float

ARGUMENTS

dynamic_power
Specifies the maximum target dynamic power of the current design; that is, the value to which the **max_dynamic_power** attribute is to be set.

GW | MW | KW | W | mW | uW | nW | pW | fW | aW
Specifies the power dimension. If not specified, then the units of *dynamic_power* are assumed to be the same as those in the technology library used during optimization.

DESCRIPTION

Sets the target (desired) dynamic power for the current design by setting the **max_dynamic_power** attribute. **max_dynamic_power** is set to a value no greater than *dynamic_power*. If **max_dynamic_power** is set more than once for a design, the last value is used.

To remove the **max_dynamic_power** attribute, use **remove_attribute** or **reset_design**.

To get information about the dynamic power usage of the current design, use **report_power**. To list the power cost of the design, use **report_constraint**.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example sets the target dynamic power to zero for the current design.

```
prompt> set_max_dynamic_power 0.0
```

SEE ALSO

`current_design(2)`
`remove_attribute(2)`
`report_constraint(2)`
`report_power(2)`
`reset_design(2)`

set_max_fanout

Sets the **max_fanout** attribute to a specified value on specified input ports and/or designs.

SYNTAX

```
int set_max_fanout  
fanout_value  
object_list
```

ARGUMENTS

fanout_value
Specifies the value to which the **max_fanout** attribute is to be set; that is, the maximum fanout value.

object_list
Specifies a list of input ports and/or designs on which the **max_fanout** attribute is to be set.

DESCRIPTION

Sets the **max_fanout** attribute on the specified input ports and/or designs. **compile** attempts to ensure that the sum of the **fanout_load** attributes for input pins on nets driven by the specified ports or all nets in the specified design is less than the given value. There is no maximum value placed by default, but if there is a **max_fanout** attribute already specified in a technology library (an implicit constraint), **compile** attempts to meet it.

In cases where both a global fanout limit is set on a design, and a local value is set on a port, **compile** attempts to meet the smaller (more restrictive) value.

In general, design rule constraints reflect technology-specific constraints that *must* be met for a design to function correctly. Optimization constraints are design goals and restrictions that are desirable but not crucial for the operation of a design. **compile** attempts to meet all constraints placed on a design, but it gives priority to design rule constraints in the optimization process.

The **max_fanout** attribute specifies Maximum Fanout, a *design rule constraint*. Thus, **compile** gives precedence to fixing a **max_fanout** violation, even if it adversely affects *optimization constraints* (for example, Maximum Delay and Maximum Area).

Use **report_constraint** to get information about optimization and design rule constraints.

To remove a maximum fanout from a port or design, use **remove_attribute**. **reset_design** removes all attributes, including **max_fanout**, from the current design.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example sets a maximum fanout of 20.0 units on the port "going_places":

```
prompt> set_max_fanout 20.0 going_places
```

In the following example, the **max_fanout** attribute is set to 18.5 on the design "TEST":

```
prompt> set_max_fanout 18.5 TEST
```

SEE ALSO

```
all_inputs(2)  
current_design(2)  
remove_attribute(2)  
report_constraint(2)  
report_port(2)  
reset_design(2)  
set_fanout_load(2)
```

set_max_leakage_power

Sets the target leakage power for the current design by setting the **max_leakage_power** attribute to a specified value.

SYNTAX

```
int set_max_leakage_power  
leakage_power  
[GW | MW | KW | W | mW | uW | nW | pW | fW | aW]
```

Data Types

leakage_power float

ARGUMENTS

leakage_power
Specifies the target leakage power of the current design; that is, the value to which the **max_leakage_power** attribute is to be set.

[GW | MW | KW | W | mW | uW | nW | pW | fW | aW]
Specifies the power dimension. If not specified, then the units of *leakage_power* are assumed to be the same as those in the technology library used during optimization.

DESCRIPTION

Sets the target (desired) leakage power of the current design by setting the **max_leakage_power** attribute. **max_leakage_power** is set to a value no greater than *leakage_power*. If **max_leakage_power** is set more than once for a design, the latest value is used.

To remove the **max_leakage_power** attribute, use **remove_attribute** or **reset_design**.

To get information about the leakage power usage of the current design, use **report_power**. To list the power cost of the design, use **report_constraint**.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example sets the target power to zero for the current design.

```
prompt> set_max_leakage_power 0.0
```

SEE ALSO

`current_design(2)`
`remove_attribute(2)`
`report_constraint(2)`
`report_power(2)`
`reset_design(2)`

set_max_lvth_percentage

Sets the the maximum percentage of the total cell area that can be of a low threshold voltage group.

SYNTAX

```
status set_max_lvth_percentage  
[max_lvth]  
[-lvth_groups groups]  
[-reset]
```

Data Types

<i>max_lvth</i>	float
<i>groups</i>	string list

ARGUMENTS

- max_lvth**
Specifies the max %lvth area constraint value to be set on the current design. The constraint value is a floating point number representing a prcentage value between 0 and 100.
- lvth_groups *groups***
Specifies the list of vth groups to be considered as low vth. The value of the *groups* argument is a list of vth group identifiers. Each identifier is a non-empty string that may consist of alphanumeric characters and the underscore character ("_").
- reset**
Removes the max %lvth constraint from the current design.

DESCRIPTION

The **set_max_lvth_percentage** command sets a max %lvth area constraint on the current design and specifies which vth groups are considered as low vth. The max %lvth area constraint specifies the maximum percentage of the total cell area that can be of a low voltage threshold (low vth) group.

The vth group of a cell is defined as the vth group of its library cell; the vth group of a library cell is defined by the value of the library cell level `threshold_voltage_group` attribute or the library level `default_threshold_voltage_group` attribute. If a design contains cells belonging to the vth groups: lvt, xlvt, svth and hvth; and if lvt and xlvt are considered as the low vth groups, then the %lvth area of the design is the ratio (represented as a percentage value) of the cell area of the cells belonging to the lvt or xlvt groups with respect to the total cell area of the design.

The max %lvth area constraint is an upperbound to the %lvth area of the design. The constraint value is specified as the floating point argument of the **set_max_lvth_percentage** command. The list of vth groups that are to be considered as

low vth is specified using the **-lvth_groups** argument. The **-reset** can be used to remove the current max %lvth constraint. The command can be issued without any arguments to display the current %lvth constraint set on the design.

In Design Compiler, the %lvth constraint is only honored by **compile_ultra** during leakage power optimization, that is, when the max_leakage_power constraint is also set. In IC Compiler, the constraint will be triggered with -power switch of mega commands (place_opt/clock_opt/route_opt) and atomic (psynopt) command.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example shows how to set the max %lvth area constraint on the current design.

```
prompt> set_max_lvth_percentage -lvth_groups {xlvt lvt} 20.0
```

In this case the %lvth constraint information specifies that the total area of the cells with vth groups xlvt and lvt can be 20% of the total design area. For example, if 10% of the cell area is of vth group xlvt and 5% is of vth group lvt, then the constraint is satisfied since 15% of the cell area is considered to be low vth. On the other hand, if 15% of the cell area is xlvt and 10% is lvt then the constraint is not met (25% of the cell area is low vth).

The following example removes the current max %lvth constraint.

```
prompt> set_max_lvth_percentage -reset
```

SEE ALSO

`set_max_leakage_power(2)`

set_max_net_length

Sets the **max_net_length** attribute to a specified value on specified input ports and/or designs.

SYNTAX

```
int set_max_net_length  
net_length_value  
object_list
```

ARGUMENTS

net_length_value

Specifies the value to which the **max_net_length** attribute is to be set; that is, the maximum net length value.

object_list

Specifies a list of input ports and/or designs on which the **max_net_length** attribute is to be set.

DESCRIPTION

Sets the **max_net_length** attribute on the specified input ports and/or designs.

physopt attempts to ensure that the real net length in on_route mode or the net length estimation in pre_route mode for the nets driven by the specified ports or all nets in the specified design is less than the given value. There is no maximum value placed by default.

In cases where both a global net_length limit is set on a design, and a local value is set on a port, **physopt** attempts to meet the smaller (more restrictive) value.

In general, design rule constraints reflect technology-specific constraints that *must* be met for a design to function correctly. Optimization constraints are design goals and restrictions that are desirable but not crucial for the operation of a design. **physopt** attempts to meet all constraints placed on a design, but it gives priority to design rule constraints in the optimization process.

The **max_net_length** attribute specifies Maximum Net Length, a *design rule constraint*. Thus, **physopt** gives precedence to fixing a **max_net_length** violation, even if it adversely affects *optimization constraints* (for example, Maximum Delay and Maximum Area).

Use **report_constraint** to get information about optimization and design rule constraints.

To remove a maximum net length from a port or design, use **remove_attribute**. **reset_design** removes all attributes, including **max_net_length**, from the current design.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example sets a maximum net length of 100.0 micron on the design "mydesign":

```
prompt> set_max_net_length 100.0 mydesign
```

SEE ALSO

```
all_inputs(2)  
current_design(2)  
remove_attribute(2)  
report_constraint(2)  
reset_design(2)
```


set_max_time_borrow

Sets the **max_time_borrow** attribute to a specified value on clocks, latch cells, data pins, or clock (enable) pins, to constrain the amount of time borrowing possible for level-sensitive latches.

SYNTAX

```
int set_max_time_borrow  
delay_value  
object_list
```

ARGUMENTS

delay_value

Specifies the value to which the **max_time_borrow** attribute is to be set; defines the desired limit of time borrowing on the latches specified in **object_list**. *delay_value* must be between zero and the default maximum derived from the waveform. By default, the maximum is derived from the ideal clock waveform driving each latch, and is equal to (closing_edge - open_edge). Library setup and data-to-Q propagation times are automatically taken into account. *delay_value* is assumed to be in the same units as those in the technology library used during optimization.

object_list

Specifies a list of objects in the current design for which time borrowing is to be limited to *delay_value*. The objects can be clocks, latch cells, data pins, or clock (enable) pins. If a cell is specified, all enable pins on that cell are affected. Note that if the **max_time_borrow** attribute is set on a cell and the cell is replaced during optimization, the attribute is moved from the cell to the enable pin.

DESCRIPTION

Sets the **max_time_borrow** attribute to **delay_value** to constrain the amount of time borrowing possible for level-sensitive latches. To meet delay targets, **set_max_time_borrow** prevents automatic use of all or part of the enabling clock pulse on a latch.

To get **max_time_borrow** attributes on the design, use **get_attribute**.

To undo **set_max_time_borrow**, use **remove_attribute**.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example restricts time borrowing on latches **latch1a** and **latch2f** to 4.0 units.

```
prompt> set_max_time_borrow 4.0 {latch1a latch2f
```

The following example specifies that no time borrowing will take place on **latch1c**.

```
prompt> set_max_time_borrow 0.0 {latch1c
```

SEE ALSO

```
current_design(2)  
remove_attribute(2)
```

set_max_total_power

Sets the target total power for the current design by setting the **max_total_power** attribute to a specified value.

The **set_max_total_power** constraint will be obsolete in a future release. Use **set_max_leakage_power** and **set_max_dynamic_power** constraints as a replacement.

SYNTAX

```
int set_max_total_power  
total_power  
[GW | MW | KW | W | mW | uW | nW | pW | fW | aW]
```

Data Types

total_power float

ARGUMENTS

total_power
Specifies the maximum target total power of the current design; that is, the value to which the **max_total_power** attribute is to be set.

GW | MW | KW | W | mW | uW | nW | pW | fW | aW
Specifies the power dimension. If not specified, then the units of *total_power* are assumed to be the same as the units of *dynamic_power* in the technology library used during optimization.

DESCRIPTION

The **set_max_total_power** constraint will be obsolete in a future release. Use **set_max_leakage_power** and **set_max_dynamic_power** constraints as a replacement.

Sets the target (desired) total power for the current design by setting the **max_total_power** attribute. **max_total_power** is set to a value no greater than *total_power*. If **max_total_power** is set more than once for a design, the last value is used. The total power cost is the sum of leakage and dynamic power costs.

Note that if the leakage power units in the library differ from those for dynamic power, then the design leakage power is scaled to the dynamic power units.

To remove the **max_total_power** design attribute, use **remove_attribute** or **reset_design**.

To get information about the dynamic and leakage power usage of the current design, use **report_power**. To list the total power cost and constraint of the design, use **report_constraint**.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example sets the target total power to zero for the current design. Since the constraint **0.0** can not realistically be met, `total_power` will be minimized.

```
prompt> set_max_total_power 0.0 mW
```

SEE ALSO

```
current_design(2)
remove_attribute(2)
report_constraint(2)
report_power(2)
reset_design(2)
set_max_dynamic_power(2)
set_max_leakage_power(2)
```

set_max_transition

Sets the **max_transition** attribute to a specified value on specified clocks group, ports or designs.

SYNTAX

```
int set_max_transition  
  transition_value  
  object_list
```

ARGUMENTS

transition_value

Specifies a maximum transition time for the specified clock groups, ports or designs. Sets the **max_transition** attribute to *transition_value* on the clock groups, ports or designs. *transition_value* must be expressed in units consistent with the those in the technology library used during optimization. For example, if the library specifies drive values in kilohms and load values in picofarads, then *transition_value* must be expressed in nanoseconds.

object_list

Specifies a list of names of the clock groups, ports or designs for whose associated nets *transition_value* is set.

DESCRIPTION

Sets the **max_transition** attribute on the specified clock groups, ports or designs. **compile** attempts to ensure that the transition time for a net is less than the specified value. The maximum transition time for a net is defined as the least of the maximum transition times of the cell pins and design ports on that net.

In cases where both a global maximum transition time is set on a design and a local value is set on a port or clock group, **compile** attempts to meet the smaller (more restrictive) value.

If **max_transition** attributes are already specified in a technology library (implicit constraints), **compile** automatically attempts to meet them.

By default, no restriction is placed on the transition time for an input or output port, so the maximum transition time for the driven net is determined by the maximum transition times (if any) of the driven cell inputs or the driving cell output. By assigning a lower maximum transition time to a port, the maximum transition time for the net is reduced; thus, the design constraints are made more restrictive.

The **max_transition** constraint is a design rule constraint; thus, **compile** gives precedence to this constraint, even if it adversely affects optimization constraints on a design. **max_delay** and **max_area** are optimization constraints, whereas **max_fanout** and **max_transition** are design rule constraints. Design rule constraints reflect those technology-specific restrictions that must be met for a design to function correctly. Optimization constraints reflect goals and restrictions that are desirable, but not crucial for the operation of a design. Design Compiler attempts

to meet all constraints placed on a design, but gives priority to design rule constraints in the optimization process.

Use **report_constraint** to get information about optimization and design rule constraints.

Use **remove_attribute** to remove a maximum transition time from a port or design.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example sets a maximum transition time of 2.0 units on the port named "late_riser".

```
prompt> set_max_transition 2.0 late_riser
```

The following example specifies a maximum transition time of 2.0 units for the design "TEST".

```
prompt> set_max_transition 2.0 TEST
```

SEE ALSO

all_inputs(2)
all_outputs(2)
characterize(2)
current_design(2)
remove_attribute(2)
report_constraint(2)
reset_design(2)

set_mcmm_job_options

Sets various job control options for Distributed MCMM.

SYNTAX

```
status set_mcmm_job_options  
[-work_dir working_directory_path]  
[-exec ICC_executable_path]  
[-setup_script setup_script_name]  
[-analysis_script analysis_script_name]
```

Data Types

<i>working_directory_path</i>	string
<i>ICC_executable_path</i>	string
<i>setup_script_name</i>	string
<i>analysis_script_name</i>	string

ARGUMENTS

-work_dir *working_directory_path*
Specifies the working directory where the Distributed MCMM jobs will be run.

-exec *ICC_executable_path*
Specifies the full path name of the ICC executable.

-setup_script *setup_script_name*
Specifies a TCL script that will be sourced by slave jobs.

-analysis_script *analysis_script_name*
Specifies a TCL script that will be sourced by slave jobs.

DESCRIPTION

This command allows you to set various options and parameters for running Distributed MCMM slave jobs. The most common parameters are set with the **set_host_options** command; **set_mcmm_job_options** is needed only for setting certain specialized parameters that are unique to distributed get_dominant_scenarios.

The **-work_dir** option specifies the full path to the working directory where all job files will be put, and where the jobs will actually be run. This directory should be accessible to all hosts that will run jobs. If it does not exist, the directory will be created. Otherwise, any existing files in it may be overwritten. If the **-work_dir** option is not given, Distributed MCMM will create a new directory under the current directory with a name of the form "mcmm_abc123".

The **-exec** option specifies an alternate ICC executable to be used by slave jobs. If this option is not given, the standard installed `icc_shell` will be used.

The **-setup_script** option specifies an optional user-written setup script to be sourced by each slave job. This script will be sourced once, at the beginning of the

slave job execution.

The `-analysis_script` option specifies an optional user-written analysis script to be sourced by each slave job. This script will be sourced immediately after each scenario is analyzed by the slave job. This can be useful for generating per-scenario reports.

The `-num_processes` option of the **set_host_options** command is used to specify how many slave jobs will be run. If a LSF or GRD compute pool is being used, the number of jobs will by default be the number of scenarios to be analyzed. If a host list is being used, the number of jobs will by default be the lesser of the number of scenarios to be analyzed, or the number of hosts specified. If the given value is greater than the number of scenarios to be analyzed, there will actually be only one job for each scenario. If the value is less than the number of scenarios, some of the jobs will analyze more than one scenario.

The `-submit_cmd` option of the **set_host_options** command is normally used to specify the full path name of the LSF or GRD job submission program. Distributed MCMM will use a default path name of this option is not given. For LSF, the default path name is `/lsf/bin/bsub`. For GRD, the default path name is `/usr/bin/qsub`. Note that the default GRD path name is unlikely to be correct for your installation; you will almost certainly need to specify the correct path name.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows how to override the executable and work dir.

```
prompt> set_mcmm_job_options -exec /usr/local/bin/my_icc_shell \  
-work_dir /remote/blah103/usr/joe/mcmm/my_work_dir  
1
```

SEE ALSO

set_host_options(2)
get_dominant_scenarios(2)

set_message_info

Set some information about diagnostic messages.

SYNTAX

```
string set_message_info  
-id message_id [-limit max_limit  
| -stop_on]
```

Data Types

<i>message_id</i>	string
<i>max_limit</i>	integer

ARGUMENTS

-id *message_id*
Information is to be set for the given *message_id*. The message must exist. Although different constraints allow different message types, no constraint allows severe or fatal messages.

-limit *max_limit*
Set the maximum number of occurrences for *message_id*. This is an integer greater than or equal to zero. If you set it to zero, that means the number of occurrences of the message is unlimited. Messages which occur after a limit is reached are automatically suppressed.

-stop_on
Force Tcl error if message is emitted.

DESCRIPTION

The **set_message_info** command sets constraints on diagnostic messages (typically error, warning, and informational messages).

Currently, you can set a upper limit for the number of occurrences of a message. You can set this to zero to indicate that there is no limit. You can retrieve the current limit for a message using the **get_message_info** command. ***When the limit is exceeded, all future occurrences of the message are automatically suppressed. A count of total occurrences (including those suppressed) can be retrieved using get_message_info.***

EXAMPLES

The following example uses **set_message_info** to set a limit on the number of APP-027 messages to 100. When the 101st APP-027 message is about to be issued, you will be warned that the limit has been exceeded, and that all future occurrences will be suppressed.

```
prompt> do_command
```

```
Warning: can't find node U27.1 (APP-027)
Warning: can't find node U27.2 (APP-027)
Warning: can't find node U27.3 (APP-027)
Warning: can't find node U27.100 (APP-027)
Note - message 'APP-027' limit (100) exceeded.  Remainder will be suppressed.
1
prompt>
```

SEE ALSO

```
get_message_info(2)
print_message_info(2)
suppress_message(2)
```

set_min_capacitance

Sets the **min_capacitance** attribute to a specified value on specified input ports in the current design.

SYNTAX

```
int set_min_capacitance  
capacitance_value  
object_list
```

Data Types

<i>capacitance_value</i>	float
<i>object_list</i>	list

ARGUMENTS

capacitance_value

Specifies a value to which the **min_capacitance** attribute is to be set. *capacitance_value* must be expressed in units consistent with the technology library used during optimization. For example, if the library specifies capacitance values in picofarads, then *capacitance_value* must also be expressed in picofarads.

object_list

Specifies a list of names of the input and/or bidirectional ports on which the **min_capacitance** attribute is to be set.

DESCRIPTION

Sets the **min_capacitance** attribute to *capacitance_value* on the specified input ports. **compile** attempts to ensure that the load driven by the input port is not reduced below *capacitance_value*. The minimum capacitance value for a net is defined as the greatest of the minimum capacitance values of the driving cell pins and ports on a net.

If **min_capacitance** attributes are already specified in a technology library (implicit constraints), **compile** automatically tries to meet them.

By default, a port has no **min_capacitance** constraint.

min_capacitance, along with **max_fanout** and **max_transition**, is a *design rule constraint*; **max_delay** and **max_area** are optimization constraints. Design rule constraints reflect those technology-specific restrictions that *must* be met for a design to function correctly, while optimization constraints reflect goals and restrictions that are desirable, but not crucial for the operation of a design. Design Compiler attempts to meet all constraints placed on a design, but gives priority to design rule constraints in the optimization process. Therefore, **compile** gives preference to **min_capacitance** and other design rule constraints, even if they adversely affect optimization constraints on a design.

To get information about optimization and design rule constraints, use **report_constraint**; to get information on the current port settings, use **report_port**. To remove the **min_capacitance** attribute from a port, use **remove_attribute**.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example sets a minimum capacitance value of 12.0 units on the port named "high_drive":

```
prompt> set_min_capacitance 12.0 high_drive
```

SEE ALSO

- all_inputs(2)
- characterize(2)
- current_design(2)
- remove_attribute(2)
- report_constraint(2)
- report_port(2)
- reset_design(2)
- set_max_capacitance(2)

set_min_delay

Specifies a minimum delay target for paths in the current design.

SYNTAX

```
int set_min_delay  
delay_value  
[-rise | -fall]  
[-from from_list  
    | -rise_from rise_from_list  
    | -fall_from fall_from_list]  
[-through through_list]  
[-rise_through rise_through_list]  
[-fall_through fall_through_list]  
[-to to_list  
    | -rise_to rise_to_list  
    | -fall_to fall_to_list]  
[-reset_path]
```

Data Types

<i>delay_value</i>	float
<i>from_list</i>	list
<i>rise_from_list</i>	list
<i>fall_from_list</i>	list
<i>through_list</i>	list
<i>rise_through_list</i>	list
<i>fall_through_list</i>	list
<i>to_list</i>	list
<i>rise_to_list</i>	list
<i>fall_to_list</i>	list

ARGUMENTS

- delay_value**
Specifies the value of the desired minimum delay for paths between startpoints and endpoints. Express *delay_value* in the same units as the technology library used during optimization. If a path startpoint is on a sequential device, clock skew is included in the computed delay. If a path startpoint has an input external delay specified, that delay value is added to the path delay. If a path endpoint is on a sequential device, clock skew and library setup time are included in the computed delay. If the endpoint has an output external delay specified, that delay is added into the path delay.
- rise | -fall**
Specifies whether endpoint rising or falling delays are constrained. If neither is specified, both rising and falling delays are constrained.
- from *from_list***
Lists the path startpoints (port, pin, clock, or cell names) of the current design. If a clock is specified, all path startpoints related to that clock

are affected. If you specify a cell name, one path startpoint on that cell is affected. All paths from these startpoints to the endpoints in the *to_list* are constrained to *delay_value*. If you do not specify *to_list*, all paths from *from_list* are affected. This list cannot include output ports. If more than one object is included, enclose the objects in double quotation marks (") or in braces ({}).

-rise_from *rise_from_list*

Same as the **-from** option, except that the path must rise from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by rising edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of **-from**, **-rise_from**, and **-fall_from**.

-fall_from *fall_from_list*

Same as the **-from** option, except that the path must fall from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of **-from**, **-rise_from**, and **-fall_from**.

-through *through_list*

Lists the path throughpoints (port, pin, or leaf cell names) of the current design. The minimum delay value applies only to paths that pass through one of the points in the *through_list*. If you include more than one object, enclose the objects in double quotation marks (") or in braces ({}). If you specify the **-through** option multiple times, the minimum delay values apply to paths that pass through a member of each *through_list* in the order the lists were given. The path must first pass through a member of the first *through_list*, then through a member of the second list, and so on for every through list specified. If you use the **-through** option in combination with the **-from** or **-to** options, the minimum delay applies only if the **-from** or **-to** conditions are satisfied and the **-through** conditions are satisfied.

-rise_through *rise_through_list*

Same as the **-through** option, but applies only to paths with a rising transition at the specified objects. You can specify **-rise_through** more than once in a single command invocation as with the **-through** option.

-fall_through *fall_through_list*

Same as the **-through** option, but applies only to paths with a falling transition at the specified objects. You can specify **-fall_through** more than once in a single command invocation as with the **-through** option.

-to *to_list*

Lists the path endpoints (port, clock, cell, or pin names) of the current design. All paths to the endpoints in the *to_list* are constrained to *delay_value*. If you do not specify *from_list*, all paths to *to_list* are affected. This list cannot include input ports. If more than one object is included, enclose the objects in either double quotation marks (") or in braces ({}). If you specify a cell, one path endpoint on that cell is affected. If you specify a clock, all path endpoints related to that clock are affected.

`-rise_to` *rise_to_list*

Same as the **-to** option, but applies only to paths rising at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths captured by rising edge of the clock at clock source, taking into account any logical inversions along the clock path. You can use only one of **-to**, **-rise_to**, and **-fall_to**.

`-fall_to` *fall_to_list*

Same as the **-to** option, but applies only to paths falling at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of **-to**, **-rise_to**, and **-fall_to**.

`-reset_path`

Removes existing point-to-point exception information on the specified paths. Only information of the same rise and fall setup or hold type is reset. This is equivalent to using the **reset_path** command with similar arguments before the **set_min_delay** command is issued.

DESCRIPTION

This command sets the minimum delay target for paths in the current design. Minimum delay is considered as an optimization constraint by the **compile** command. If a path violates the requirement given in a **set_min_delay** command, **compile** adds delay to fix the violation if maximum delay cost is not increased. You can prioritize minimum delay cost using the **set_cost_priority** command.

The value of a **min_rise_delay** attribute can never be greater than that of a **max_rise_delay** attribute for the same path (and similarly for fall attributes). If this occurs, the old attribute is removed.

Individual minimum delay targets are automatically derived from clock waveforms and port input or output delays. For more information, refer to the **create_clock**, **set_input_delay**, and **set_output_delay** command man pages.

The **set_min_delay** command is a point-to-point timing exception command. It overrides the default single-cycle timing relationship for one or more timing paths. Other point-to-point timing exception commands include **set_multicycle_path**, **set_max_delay**, and **set_false_path**.

If a path satisfies multiple timing exceptions, the following rules determine which exceptions take effect. Rules referring to **-from** apply equally to **-rise_from** and **-fall_from**, and similarly for the rise and fall options of **-through** and **-to**.

1. Two **group_path** commands can conflict with each other, but a **group_path** exception alone does not conflict with another type of exception. Therefore, the remaining rules apply for two **group_path** exceptions, or two non **group_path** exceptions.
2. If both exceptions are **set_false_path**, there is no conflict.
3. If one exception is a **set_max_delay** and the other is a **set_min_delay**, there is no conflict.
4. If one exception is a **set_multicycle_path -hold** and the other is a **set_multicycle_path -setup**, there is no conflict.
5. If one exception is a **set_false_path** and the other is not, the **set_false_path**

takes precedence.

6. If one exception is a **set_max_delay** and the other is not, the **set_max_delay** takes precedence.

7. If one exception is a **set_min_delay** and the other is not, the **set_min_delay** takes precedence.

8. If one exception has a **-from pin** or **-from cell** and the other does not, the one with the **-from pin** takes precedence.

9. If one exception has a **-to pin** or **-to cell** and the other does not, the one with the **-to pin** takes precedence.

10. If one exception has any **-through** points and the other does not, the one with **-through** points takes precedence.

11. If one exception has a **-from clock** and the other does not, the one with the **from clock** takes precedence.

12. If one exception has a **-to clock** and the other does not, the one with the **-to clock** takes precedence.

13. The exception with the more restrictive constraint then takes precedence. For **set_max_delay** and **set_multicycle_path -setup**, this is the constraint with the lower value. For **set_min_delay** and **set_multicycle_path -hold**, it is the constraint with the higher value.

To remove information set by **set_min_delay**, use the **reset_path** or **reset_design** command.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

In the following example, the **set_min_delay** command requires that any delay path that passes through the UI cell and ends at the Y port is greater than 12.5 time units:

```
prompt> set_min_delay 12.5 -through U1 -to Y
```

The following example specifies that all paths from A1 and A2 to Z5 must be greater than 4.0 units:

```
prompt> set_min_delay 4.0 -from {A1 A2} -to Z5
```

The following example command sets a minimum delay requirement of 3.0 for all paths that first pass through either u1/Z or u2/Z and then pass through u5/Z or u6/Z:

```
prompt> set_min_delay 3.0 -through {u1/Z u2/Z} -through {u5/Z u6/Z}
```

The following example specifies that all timing paths from ff1/CP to ff2/D that rise through one or more of {U1/Z U2/Z} and fall through one or more of {U3/Z U4/C} must have delays greater than 3.0 units.

```
prompt> set_min_delay 3.0 -from ff1/CP -rise_through {U1/Z U2/Z} -fall_through {U3/Z U4/C} -to ff2/D
```


SEE ALSO

`current_design(2)`
`report_constraint(2)`
`reset_design(2)`
`reset_path(2)`
`set_cost_priority(2)`
`set_fix_hold(2)`
`set_max_delay(2)`

set_min_library

Sets an alternate library to use for minimum delay analysis.

SYNTAX

```
int set_min_library  
max_library  
-min_version min_library | -none
```

Data Types

<i>max_library</i>	string
<i>min_library</i>	string

ARGUMENTS

max_library
Specifies the library file name to use as a *link_library* or *target_library*. This library is used for maximum delay analysis only. Do not include the path to the library file name.

-min_version *min_library*
Specifies the library file name that corresponds to the *max_library* and uses it for minimum delay analysis. The *min_library* should not be specified in the *link_library* or *target_library*. Do not include the path to the library file name.

-none
Unsets the setting of a minimum library.

DESCRIPTION

The **set_min_library** command creates a minimum/maximum relationship between two library files. The *max_library* is used for maximum delay analysis and the *min_library* is used for minimum delay analysis.

Whenever the tool computes a minimum delay value, it first consults the library cell from the max library. If a library cell exists with the same name, the same pins, and the same timing arcs in the min library, the timing information from the min library is used. If the tool cannot find a matching cell in the min library, the max library cell is used.

Do not include the path to the *max_library* nor *min_library* file name. Specify the path to those library files in *\$search_path*.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example is a typical usage of **set_min_library**:

```
prompt> set link_library "LIB_WC_COM.db"
prompt> set target_library $link_library
prompt> set_min_library LIB_WC_COM.db -min_version LIB_BC_COM.db
prompt> set_operating_conditions -max WC_COM -max_library LIB_WC_COM \
    -min BC_COM -min_library LIB_BC_COM
prompt> report_timing -delay max
prompt> report_timing -delay min
```

SEE ALSO

link(2)
set_operating_conditions(2)
link_library(3)
target_library(3)

set_mode

Selects the mode of a component.

SYNTAX

```
int set_mode  
[mode_list]  
[instance_list]
```

Data Types

```
mode_list          list  
instance_list      list
```

ARGUMENTS

```
mode_list  
    Specifies the active modes for timing analysis. All other modes of the given  
    mode group are implicitly inactive. The specified modes are active on the  
    given instance list.  
  
instance_list  
    Specifies the cells for which the specified modes are active.
```

DESCRIPTION

The **set_mode** command is used to select the active mode of a cell that has several functional modes. The other modes in the mode group that are not selected are disabled.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following command selects the READ mode for the RAM instance Uram1.

```
prompt> set_mode READ Uram1
```

SEE ALSO

```
report_mode(2)  
reset_mode(2)
```

set_mpc_macro_array

Specifies an array of macro cells.

SYNTAX

```
int set_mpc_macro_array
-name string
-elements list
[-align_edge t | b | l | r | c | top | bottom | left | right | center]
[-align_pins {ref_pin const_pin}]
[-x_offset float]
[-y_offset float]
[-use_keepout_margin]
[-vertical]
[-rectilinear]
[-align_2d lb | lc | lt | rb | rc | rt | cb | cc | cr | left-bottom | left-
center | left-top | right-bottom | right-center | right-top | center-
bottom | center-center | center-top]
[-verbose]
[-reset]
```

ARGUMENTS

-name *string*

Specifies name for the macro array. This is a mandatory option.

-elements *list*

This option specifies list-of-list of macro cells which will compose the array. Each sub list represents a row in the array. An array starts from top most row. A row starts from left. For example, to specify an array like following

A11 A12 A13

A21 A22 A23

A31 A32 A33

```
-elements [list [list A11 A12 A13] \
              [list A21 A22 A23] \
              [list A31 A32 A33]]
```

You can use the **get_cell** command to obtain the macro cell object. For example, [**get_cell** RAM1]. Note that [get_cell xx] won't work between double-quotes or curly braces. For example, you can't get a list of collection using "[get_cell A] [get_cell B]", instead you should use [list [get_cell A] [get_cell B]]. This argument is mandatory.

-align_edge t | b | l | r | c | top | bottom | left | right | center

Specifies an edge for the macro cells in one row to align with. This option is not mandatory. Default is to align to the centers.

-align_pins {*ref_pin* *const_pin*}

Align with pins. 'ref_pin' is the name of a pin on reference cell (first cell in the array). 'the_pin' is the name of a pin from second cell. This option takes effect only if specified array has two macro cells.

-x_offset float

Specifies the distance between two adjacent edges of macro cells on one ROW. Unit is micron. Default value is 0, which means the cells will abut to each other.

-y_offset float

Specifies the distance between two adjacent edges of macro cells on the same column of the next row. Unit is micron. Default value is 0, which means the cells will abut to each other.

-use_keepout_margin

Specifies the tool to use keepout margin to determine the spacing between two adjacent rows and/or columns of the array. This option is mutually exclusive with **-x_offset** and **-y_offset** options.

-vertical

If specified for one dimensional array, it causes the command to arrange one dimensional macros in a vertical column. By default these arrays are created as horizontal rows (unless you specify each cell in a separate list in the list). If specified for two dimensional array, it causes the command to arrange the array in column based structure, instead of row based structure. It is like rotate a two dimensional array 90 degrees.

-rectilinear

This option is for design planning only. This option is for 2D array. It specifies that the overall shape of the 2D array is rectilinear. For a rectilinear macro array, you can specify different number of cells in different rows. This option is not mandatory. By default, in design planning, a 2D macro array is placed as a single object that is rectangular in shape, and all the rows in the array have the same number of cells.

-align_2d lb | lc | lt | rb | rc | rt | cb | cc | cr | left-bottom | left-center | left-top | right-bottom | right-center | right-top | center-bottom | center-center | center-top

"left-bottom | left-center | left-top | right-bottom | right-center | right-top | center-bottom | center-center | center-top" This option is for design planning only. This option is for heterogeneous 2D array. It specified edges for the macro cells in a row or column to align with in a heterogeneous 2D macro array. This option is not mandatory. Default is to align the cells in the same column by their left edge, and align the cells in the same row by their bottom edge.

-verbose

Lets the tool print out information about what you have specified for each row and what object the tool is getting. This switch can be used to make sure the **-elements** option does create an array as you expected.

-reset

Removes the relative constraints between these macro cells and removes the macro array as an object.

DESCRIPTION

The **set_mpc_macro_array** command Defines an array of macro cells. The tool sets

relative location constraints between macro cells in the array according to the parameters specified for the array.

Multicorner-Multimode Support

This command is not supported in a multi-scenario design flow.

EXAMPLES

The following example shows how to specify a 2x2 array for cell A11 A12 A21 A22 with 10 microns space between rows and columns.

```
prompt>set_mpc_macro_array
        -name AA
        -elements [list [list A11 A12]
                        [list A21 A22]]
        -x_offset 10
        -y_offset 10
```

SEE ALSO

```
create_placement(2)
report_mpc_macro_options(2)
report_mpc_macro_array(2)
set_mpc_options(2)
set_mpc_port_options(2)
```

set_mpc_macro_options

Specifies constraints on the specific floorplan macro cells and macro arrays in MPC flows.

SYNTAX

```
int set_mpc_macro_options
[list_of_macro_objects]
-legal_orientations list
-anchor_bound {tl | t | tr | r | br | b | bl | l | tm | bm | lm | rm | c}
-x_offset float
-y_offset float
-align_pins list
-snap_to_edge side_and_float_list
-edge_channel float
-side_channel {left_right_top_bottom}
-group_name string
-group_spacing float
-reset
```

ARGUMENTS

list_of_macro_objects

This option specifies a list of macro cells the command applies to. Use the **get_cell** command to obtain the macro cell object. For example, [**get_cell Input***]. You can also specify macro array names to constraint them using the above switches. This argument is mandatory.

-legal_orientations *list*

Specify the legal orientation of the macro cells so that the placement tool will optimize the macro cell orientation within the defined values of the -legal_orientations option. The value can be one or more than one of the 8 possible orientations represented by following strings.

PDEF syntax: 0, 90, 180, 270, 0-mirror, 90-mirror, 180-mirror, 270-mirror

DEF syntax: N, W, S, E, FN, FW, FS, FE

This option will overwrite the legal orientations defined in library. If user does not specify this option, then the macro cell's library legal orientation will be used in the placement.

User can specify a single legal orientation to force the placer to place the cell in the fixed orientation.

User cannot specify orientation for macro arrays.

-anchor_bound {tl | t | tr | r | br | b | bl | l | tm | bm | lm | rm | c}

Specifies a macro cell's anchor bound (a kind of placement move bound).

The valid keywords and their definitions are shown below:

Keyword	Definition
tl	top-left corner
t	top
tr	top-right corner

r	right
br	bottom-right corner
b	bottom
bl	bottom-left corner
l	left
tm	top-middle
bm	bottom-middle
lm	left-middle
rm	right-middle
c	center

This anchor bound constrains the placement tool so that it places the macro cell in one of the boundary areas of the block. This forces the macro cell to stay at one of the boundaries of the block. The default move bound will be one-half or one-quarter of the core area associated with the specified anchor boundary. For example, if you specify `-anchor_bound` as **t**, the move bound will be the top half of the core area. If you specify `-anchor_bound` as **tr**, the move bound will be the top-right area of the core area. If the one-half or one-quarter core area is not large enough to hold the macro cell, with any set of dimensions, the move bound is set to cell width or height in that dimension.

The type of the `-anchor_bound` is SOFT, which means that placement can always move the macro cell out of the bound in case there is no space and the cost is too high.

You can specify `-anchor_bound` for macro arrays.

`-x_offset float`

`-y_offset float`

Specifies the distance between the constrained element and the core edges. If the `-anchor_bound` is t, b tm or bm, user should specify only `-y_offset`. If the `-anchor_bound` is l, r, lm or rm, user should specify only `-x_offset`. If the `-anchor_bound` is tl, tr, bl or br, user can specify `-x_offset` or `-y_offset` or both.

`-align_pins list`

Aligns a specific port to a pin of constrained macro cell. The argument passed with this switch is a list of two pins; reference pin followed by constrained pin. For example, `set_mpc_macro_options [get_cell A1] -anchor_bound t -align_pins [list DATA[0] [get_pins A1/DATA[0]]]`.

`-snap_to_edge side_and_float_list`

Specify this when you want to place the macro exactly an offset away from each side specified. Offset defaults to 0 if not provided. You can provide a list of side and offset values as input. The side could be either full name of single letter abreviation. (Left, LEFT, L)

`-edge_channel float`

Specify this when you want to place the macro atleast some distance between the core and the macro. MPC will place the macro the channel distance away from each side.

`-side_channel {left_right_top_bottom}`

Specify this when you want to place the macro at least some distances away from the core edge and the distance on each side are different.

-group_name *string*
 Provide a name for the group of macros which you try to specify 'group_spacing'. Without specifying it, the tool will generate a default name for the group.

-group_spacing *float*
 Specify this when you want to place the macros in the input list atleast some distance away from each other . This will create a group with a default name if none provided.

-reset
 Removes the attributes set with previous uses of the **set_mpc_macro_options** command.

DESCRIPTION

The **set_mpc_macro_options** command sets the constraints for the macro cell and macro array in the minimal physical constraints flow. The command constrains the **create_placement -mpc**, **physopt -mpc**, and **compile_physical -mpc** commands in placing the macro cells in the floorplan. You use this command before **create_placement -mpc** or **compile_physical -mpc** in the Tcl source file. Use this command after you load the physical library and the design.

This command is also available using the GUI, which provides a dialogue box for easy interface to create macro option constraints.

You can use this command to restrict macro cells and macro arrays in different ways:

- To a particular region, by specifying the **-anchor_bound** option.
- During the macro cell placement, by constraining with the **-legal_orientation** option, and the library legal orientations. Macro cell orientation must be one of the legal orientations.

Multicorner-Multimode Support

This command is not supported in a multi-scenario design flow.

EXAMPLES

The following example shows how to place the cell RAM1 only on the top-right corner of the core area and in south orientation.

```
prompt>set_mpc_macro_options [get_cell "RAM1"] \  
-legal_orientation S -anchor_bound tr  
prompt>set_mpc_macro_options ICRamMacroArray \  
-anchor_bound tr  
prompt>set_mpc_macro_options [all_macro_cells] -group_space 30  
prompt>set_mpc_macro_options \  
-snap_to_edge "left 10 right 20" ICRamMacroArray
```

```

prompt>set_mpc_macro_options \
    -snap_to_edge "l right 20" ICRamMacroArray
prompt>set_mpc_macro_options \
    -snap_to_edge "l 2 r t b 20" ICRamMacroArray
prompt>set_mpc_macro_options \
    -edge_channel 30 ICRamMacroArray
prompt>set_mpc_macro_options \
    -side_channel {20 30 30 0} ICRamMacroArray
prompt>set_mpc_macro_options \
    -group_spacing 30 [all_macro_cells]

```

SEE ALSO

```

create_placement(2)
report_mpc_macro_options(2)
report_mpc_options(2)
report_mpc_port_options(2)
set_mpc_options(2)
set_mpc_macro_array(2)
set_mpc_port_options(2)

```

set_mpc_options

Specifies the constraints used to generate the floorplan in the create_placement -mpc flow, the compile_physical -mpc flow, or the physopt -mpc flow.

SYNTAX

```
int set_mpc_options
-utilization util
-aspect_ratio y_x_ratio
-row_direction {h | v}
-origin {X Y}
-top_port_limit limit_value
-bottom_port_limit limit_value
-left_port_limit limit_value
-right_port_limit limit_value
-horz_port_layer_name layer
-vert_port_layer_name layer
-min_port_pitch integer
-routing_track_offset_x {h | f}
-routing_track_offset_y {h | f}
-io_margin_left margin_value
-io_margin_right margin_value
-io_margin_top margin_value
-io_margin_bottom margin_value
-corner_keepout distance
-core_width width
-core_height height
-first_row_orientation orient
-dont_snap_port
-dont_promote_layer
-shift_to_center
-rectilinear_outline
-reset
```

Data Types

<i>util</i>	float < 2.0
<i>y_x_ratio</i>	float
<i>X</i>	float
<i>Y</i>	float
<i>limit_value</i>	integer
<i>layer</i>	string
<i>integer</i>	limit_value
<i>margin_value</i>	float
<i>distance</i>	float
<i>width</i>	float
<i>height</i>	float
<i>orient</i>	string

ARGUMENTS

`-utilization util`

Specifies target placement utilization for core area. The utilization is defined as standard cell utilization:

$$\text{total\_standard\_cell\_area} / (\text{core area} - \text{blockage area} - \text{macro\_cell\_area})$$

Area	Definition
----	-----

Core area	Core area bounding box

Standard cell area	sum of fixed and non-fixed standard cell area

blockage area	sum of placement keepout area and filler cell (physical only cell) area

macro_cell area	sum of fixed and non-fixed macro cell area Note: for utilization calculation purpose, we define the macro cell as the cell in the netlist without lib site linked, or the cell width and height is larger than 5 minimum lib site height.

Value for this option is a float number less than 2. For example, 0.85 for 85% utilization. This option also related to following options: `-core_width`, `-core_height` and `-aspect_ratio`. See DESCRIPTION below for some details. The default value is 0.60.

`-aspect_ratio y_x_ratio`

Specifies aspect ratio for core area which is y:x ratio. For example, specifying 2 for this option will result a core area whose y dimension is 2 times of x dimension. This option also related to following options: `-core_width`, `-core_height` and `-utilization`. See DESCRIPTION below for details. The default value is 1.0.

`-row_direction {h | v}`

Specifies direction for core site array. The value can be one of horizontal (h) or vertical (v). The default direction is horizontal.

`-origin {X Y}`

Specifies the coordinates of lower-left corner of core area. The unit is micro in float format. The default value is {0.0 0.0}.

`-top_port_limit limit_value`

`-bottom_port_limit limit_value`

`-right_port_limit limit_value`

`-left_port_limit limit_value`

Specifies the limit of the number of pins on the each side of core area. Require an integer value for this constraint. The tool will not place more than the specified number of pins on each side. These limits have no effect if the `-dont_snap_port` option is set. It is integer type. The default value is none, e.g. unlimited.

`-horz_port_layer_name layer`

`-vert_port_layer_name layer`

Specifies the order of port layer names on each direction which the program uses to place ports. If user does not specify the port layer in `set_mpc_port_options` command, the program will use these layer names as the port layer where the ports will be placed.

The horizontal port layers apply to the ports located on left and right side of the block. The vertical port layers apply to the ports located on top and bottom side of the block.

If not specified, the tool will look at layers defined in physical library and picks the second encountered routing layer on the preferred direction as default layers for ports. This options takes comma or space separated metal names as input.

`-min_port_pitch integer`

Specifies the minimum metal pitch distance between two ports. Default is 1, which means ports can be placed at adjacent tracks.

`-routing_track_offset_x {h | f}`

Specifies x offset to origin of core area for vertical routing tracks. The value can be one of h (half metal pitch) or f (full metal pitch). The default offset is half metal pitch.

Note that the offset is relative to the origin of core area. Routing tracks will cover die area if different from core area.

`-routing_track_offset_y {h | f}`

Specifies y offset to origin of core area for horizontal routing tracks. The value can be one of h (half metal pitch) or f (full metal pitch). The default offset is half metal pitch.

Note that the offset is relative to the origin of core area. Routing tracks will cover die area if different from core area.

`-io_margin_top margin_value`

`-io_margin_bottom margin_value`

`-io_margin_left margin_value`

`-io_margin_right margin_value`

Specifies the I/O margin on each side. These distances are used to calculate DIEAREA defined in PDEF. It is float type. The unit is micron. The default value is 0.

`-corner_keepout distance`

Specifies a distance to keep ports away from corners. If the specified value is less than 1, it is used as a percentage to the length of core edges.

Otherwise, it is an absolute distance from the corner, unit is micron. Default value is 0.

`-core_width width`

`-core_height height`

Specifies width and height of the core area. These options are also related to following options: `-utilization` and `-aspect_ratio`. See DESCRIPTION below

`set_mpc_options`

for details. It is float type. Unit is micron. No default value for the options.

-first_row_orientation *orient*

Specifies the orientation of the sites on the bottom row (first row) of the placement area. In the placement area, the rows are placed in flip pattern, e.g. if the first row of sites are orient in 0 (N), the second row of sites will be orient in 180-Mirror (FS). You should check the physical library (LEF or .plib) to check the legal site orientations. An error is generated if you specify an illegal orientation. Default is 0 or NORTH.

PDEF syntax: 0, 90, 180, 270, 0-mirror, 90-mirror, 180-mirror, 270-mirror DEF syntax: N, W, S, E, FN, FW, FS, FE

-dont_snap_port

By default, the command snaps ports to block boundaries automatically. This option can switch off the port snapping if you want to see the ports at the locations from placement.

-dont_promote_layer

By default, the command tries to place the ports on all available track on all layers. The command tries to exhaust all the tracks on all the layers. If you strictly want the tool to use just the layers specified by using -vert_port_layer_name and -horz_port_layer_name, use this option.

-shift_to_center

By default, the origin (0 0) of core area is at lower-left corner. This option tells the tool to generate core area with origin (0 0) at the center.

-rectilinear_outline

This indicates to the mpc flow to treat the keepouts defined on the core boundary to be treated as rectilinear boundary for the core. This will be another way to specify the rectilinear outline. Also see set_mpc_rectilinear_outline command.

-reset

Removes all the attributes set by previous set_mpc_options commands. If there is default value, it will be set back to default value.

DESCRIPTION

This command defines design level physical constraints, so called MPC constraints. create_placement -mpc, compile_physical -mpc and physopt -mpc commands will take these constraints in generating coarse floorplan. User should use this command before the create_placement -mpc, physopt -mpc or compile_physical -mpc command in tcl source file. Also, user should use this command after the physical library and the design are loaded.

Following options are related and may causes conflicts. User should prevent the conflict when using them. Program will error out if there is any conflict which cannot be resolved. -utilization -aspect_ratio -core_width -core_height

There is no conflict when using -utilization together with -aspect_ratio; However, there may be conflict when using -core_width and -core_height together. Following table shows all the possible conflicts may occur. User should not use these options

together. From the table we can conclude that any 3 of them cannot use together. Every 2 of them can be used together.

util	ratio	width	height	CONFLICT?

use	use	use	use	YES
no	use	use	use	YES
use	no	use	use	YES
use	use	no	use	YES
use	use	use	no	YES
no	no	no	use	YES
no	no	use	no	YES
all other usages produce no conflicts				

Multicorner-Multimode Support

This command is not supported in a multi-scenario design flow.

EXAMPLES

The following example shows how to set floorplan options for current design. It sets the floorplan utilization to 85%, placement area aspect_ratio (y:x) to 2:1, horizontal row placement; and the first row has the site orientation of 0-mirror (e.g., FN, Flip-North), and the left-lower corner of the placement area is (10 10) micron. Also it sets that the placement tool can only place 10 ports on top side of the block.

```
prompt> set_mpc_options \ -utilization 0.85 \ -aspect_ratio 2.0 \ -row_orientation h
\ -first_row_orientation 180-Mirror \ -origin {10.0 10.0} \ -top_port_limit 10
```

```
prompt> set_mpc_options \ -horz_port_layer_name "M5,M7,M3" \ -vert_port_layer_name
"M4 M6 M2" \ -dont_promote_layer
```

SEE ALSO

```
create_placement(2)
set_mpc_port_options(2)
set_mpc_macro_options(2)
set_mpc_rectilinear_outline(2)
report_mpc_options(2)
report_mpc_port_options(2)
report_mpc_macro_options(2)
```


set_mpc_pnet_options

Specifies the constraints for pnet generation in the floorplan in the create_placement -mpc or physopt -mpc flows.

SYNTAX

```
int set_mpc_pnet_options  
-name pnet_name  
-reset  
-layer layer_name  
-type pwr | gnd  
-width float  
-pitch float  
-offset float  
-direction {h | v}  
-model_vias  
-mult_via_x float  
-mult_via_y float
```

Data Types

<i>pnet_name</i>	string
<i>layer_name</i>	string

ARGUMENTS

-name <i>pnet_name</i>	Unique name for the pnet. Used to create the name of the pnet.
-reset	Removes the attributes set by previous set_mpc_pnet_options commands.
-layer <i>layer_name</i>	Name of the layer the pnet trunk is to be drawn. Defined in the plib library.
-type pwr gnd	Type of the pnet must be one of pwr (power) or gnd (ground).
-width <i>float</i>	Width of the power trunk.
-pitch <i>float</i>	Center to center distance between two trunks.
-offset <i>float</i>	Distance from the left of die or core for vertical trunks and Distance from the bottom for horizontal trunks.
-direction {h v}	The direction must be one of h (horizontal) or v (vertical). If the direction is not specified, the tool uses the preferred direction for the layer, as defined in the library.

-model_vias
Should model vias for the layers in between.

-mult_via_x float
When modeling vias for pnets between trunk site or trunk to trunk pnets this value is to be used to generate a via model with width on x axis multiplied by this number.

-mult_via_y float
When modeling vias for pnets between trunk site or trunk to trunk pnets this value is to be used to generate a via model with width on y axis multiplied by this number.

DESCRIPTION

This command specifies the pnet options, which are used to create power trunks and model via in the `create_placement -mpc` command. You should issue these commands after the design and physical libraries are loaded.

During the flow the options would be read and the trunks would be drawn based on the options provided.

The generated pnets could be viewed after generation using the command **report_pnet -only_physical**.

The generated pnets can be removed using the **remove_net -only_physical [get_pnet name]** command. Please see the example.

Multicorner-Multimode Support

This command is not supported in a multi-scenario design flow.

EXAMPLES

The following example shows how to specify options for trunk type `metal6_pwr` on layer `metal6` using the widths and pitch. Followed by reporting and removing the generated pnet.

```
prompt> set_mpc_pnet_option -layer Metal6 \  
-name gnd_metal6 -width 3.00 -pitch 16.00 \  
-type gnd -offset 5.10 -direction v \  
-model_via -mult_via_x 3.3 -mult_via_y 6.6  
  
prompt> report_net -only_physical  
  
prompt> get_pnet  
  
prompt> remove_net -only_physical [get_pnet gnd_metal6]
```

SEE ALSO

`report_net(2)`
`remove_net(2)`
`report_mpc_pnet_options(2)`
`create_placement(2)`
`set_mpc_options(2)`
`set_mpc_macro_options(2)`
`report_mpc_options(2)`
`report_mpc_port_options(2)`
`report_mpc_macro_options(2)`

set_mpc_port_options

Specifies the constraints for ports in the floorplan in the `create_placement -mpc`, the `physopt -mpc`, or the `compile_physical -mpc` flows.

SYNTAX

```
int set_mpc_port_options
-side {l | r | t | b}
-x_bounds {min max}
-y_bounds {min max}
-layer layer_name
-group
-group_name group_name
-order {cw | ccw}
-pitch int
-start_location {x y}
-pin_order int
-offset float
-width float
-height float
-dont_snap
-reset
[port_list]
```

Data Types

<i>min</i>	float
<i>max</i>	float
<i>layer_name</i>	string
<i>group_name</i>	string
<i>x</i>	float
<i>y</i>	float
<i>float</i>	y
<i>port_list</i>	list or collection

ARGUMENTS

`-side {l | r | t | b}`
Specifies the side to which the ports should snap. The value is one of l (left), t (top), b (bottom), r (right). This option has no effect if `-dont_snap_pins` is used in the **set_mpc_options** command. By default there is no side constraint on the ports. The ports are snapped to the nearest side of the block.

`-x_bounds {min max}`

`-y_bounds {min max}`
Specifies the bound for the ports. Floating point numbers from 0.0 to 1.0 will be taken as percentage of the core x- or y-dimension. Values greater than 1.0 are of unit micron. The created bound is a hard bound. By default there are no bounds for the ports (the ports can be placed anywhere along the side of the placement area). Be careful when specifying the bound to avoid

conflicts with the `-side` specification. If conflicts occur the `-side` specification takes priority. For example, if you specify `-side top`, the `-y_bound` specification is ignored.

`-layer layer_name`

Specifies layer name of the port geometry.

If you do not specify the layer name, the program uses the default layer name as set by the **set_mpc_options** command.

`-group`

Indicates that the list of ports should be grouped together. No other ports should be introduced in between. You can use the `-group_name` option to specify a name for the port group. If you do not specify a name for the port group, the tool generates a default name for it. If you do not specify the `-order` option, the tool does not consider the port ordering.

`-group_name group_name`

Specifies a name for the port group. This argument is optional.

If you do not specify this option, the tool generates a default name.

`-order {cw | ccw}`

Specifies that the specified ports should be kept in clockwise or counterclockwise order. Valid values for this argument are `cw` (clockwise) or `ccw` (counterclockwise). If you do not specify `-group`, other ports might be placed in the space between ports.

`-pitch int`

Specifies the space between two ports in the group. The value of the integer is the number of pitches of the port layer. This argument is optional.

If you do not specify this option, the value specified by the `-min_port_pitch` option of the **set_mpc_options** command is used, or 1 pitch is used as the default.

`-start_location {x y}`

Specifies the location of the first port of the group. You can specify the start location using either absolute coordinates or relative coordinates (float numbers of $0 < x \leq 1$, $0 < y \leq 1$ are interpreted as a percentage of the x-dimension or y-dimension of the core area). This argument can only be used with `-group`. Use of this argument implies `-order`.

`-pin_order int`

Specifies the order for the pin (as per the TDF syntax). The order could be integer number from (0..32767). If a port has a `pin_order` value of 0, the placement of the pin to the other pins is not important. If a port has a `pin_order` value of 1, no pin can be placed between the pin and corner of the boundary. If a port has a `pin_order` value of 2, any number of pins can be placed between the pin and the corner of the boundary. If two pins have consecutive `pin_order` no pins can be placed between them. If a group of pins has the same `pin_order` value, no other pins can be placed between any two of the pins in the group. In this case, ordering inside the group is not important. However, the placement of the group relative to the other pins on the same side is determined by the `pin_order` value.

`-offset float`

Relative offset of the pin placement from the bottom edge or the left edge

of the cell boundary. Valid values: Any number (in distance unit) that is greater than the height of the tallest pin the layout. A negative number to indicate that you do not want to place an offset constraint on the pin. In case of contradiction between constraints If you specify a offset value that contradicts the specified pin order, offset value will be relaxed. If you specify two offset values that contradict each other, the offset constraint with the lowest pin_order value will be considered.

`-width float`

Specifies the dimension along the cell boundary. Must be used along with the `-height` switch.

`-height float`

Specifies the dimension of the port from the boundary towards the center of the cell. Must be used along with the `-width` switch.

`-dont_snap`

Specifies that the port should not be snapped to the core or the io margin boundary. You can specify this for each individual port.

`-reset`

Removes the attributes set by previous `set_mpc_port_options` commands.

`port_list`

Specifies the list of ports that this command applies to. This argument is mandatory.

DESCRIPTION

This command specifies the port constraints for MPC. It constrains the `create_placement -mpc` and `compile_physical -mpc` commands in placing the ports in the floorplan. Use this command after loading the physical library and design, but before running the `create_placement -mpc` or `compile_physical -mpc` command.

The ports are snapped to the specified side. If you do not specify a side, the ports are snapped to the side nearest to the port location assigned by the coarse placer. If you specify `-dont_snap_pins` in the **`set_mpc_options`** command, the ports are not snapped to the sides of the block. If you specify constraint `-dont_snap` on a pin, that pin is not snapped to the side of the block.

if `-height` and `-width` constraints are provided shape for the pin will be created if not already present, after the cell is fixed.

Multicorner-Multimode Support

This command is not supported in a multi-scenario design flow.

EXAMPLES

The following example shows how to constrain placement of the o1 port to the top side of the block.

```
set_mpc_port_options
2592
```

```
prompt> set_mpc_port_options [get_port "o1"] -side top prompt>  
set_mpc_port_options [get_port "o1"] -dont_snap
```

SEE ALSO

```
create_placement(2)  
set_mpc_options(2)  
set_mpc_macro_options(2)  
report_mpc_options(2)  
report_mpc_port_options(2)  
report_mpc_macro_options(2)
```

set_mpc_rectilinear_outline

Specifies the rectilinear outline constraint for the core generation in create_placement -mpc, physopt -mpc or compile_physical -mpc flows.

SYNTAX

```
int set_mpc_rectilinear_outline  
-coordinates list_of_floats  
-reset  
-verbose
```

ARGUMENTS

```
-coordinates list_of_floats  
    List of enclosing coordinates for the rectilinear outline shape.  
  
-reset  
    Removes the attributes set by previous set_mpc_rectilinear_outline commands.  
  
-verbose  
    Reports verbose messages.
```

DESCRIPTION

This is the command for specifying the rectilinear core outline in the create_placement -mpc, physopt -mpc and compile_physical -mpc flows. MPC will create a rectangular core from this outline. Places placement and wiring keepouts on the core to model a rectilinear core.

This rectilinear outline is used by MPC flow to create

Rectangular core area.

Placement and Wiring keepouts

Based on the io_margins provided the die area, if none provided the die and core will overlap.

The rectilinear site array.

Rectilinear power mesh and Power rings structure.

Outline aware port placement.

To report the constraints please use the **report_mpc_rectilinear_outline** command.

To remove the constraint please use the **-reset** option.

Multicorner-Multimode Support

This command is not supported in a multi-scenario design flow.

EXAMPLES

The following example shows how to specify the rectilinear outline.

```
prompt> set_mpc_rectilinear_outline \  
        -coordinates {500 500 800 500 800 \  
                      1000 1000 1000 1000 2000 800 2000 800 3000 \  
                      500 3000 500 2000 0 2000 0 1000 500 1000}  
  
prompt> report_mpc_rectilinear_outline
```

SEE ALSO

```
report_net(2)  
remove_net(2)  
report_mpc_pnet_options(2)  
create_placement(2)  
set_mpc_options(2)  
set_mpc_macro_options(2)  
report_mpc_options(2)  
report_mpc_port_options(2)  
report_mpc_macro_options(2)
```

set_mpc_ring_options

Specifies the constraints for ring generation around cores, macros, macro arrays, and designs in the floorplan in the flows of the **create_placement**, **physopt**, and **compile_physical** commands with the **-mpc** option.

SYNTAX

```
int set_mpc_ring_options  
[-name name]  
[-reset] | [-layer layer_name -type -width width]  
[-offset offset]  
[-sides {top | left | right | bottom}]  
[-layer ]
```

Data Types

<i>name</i>	string
<i>layer_name</i>	string
<i>width</i>	float
<i>offset</i>	float

ARGUMENTS

- name** *name*
Specifies a unique name for the ring. This option is used to create the name of the pnet. The default is for the tool to provide a name.
- reset**
Removes the attributes set by previous uses of the **set_mpc_pnet_options** command. If you do not use the **-reset** option, then the **-layer**, **-type**, and **-width** options are required.
- type** {pwr | gnd}
Defines the type of pnet as either **pwr** (for power) or **gnd** (for ground). This option is required if you do not use the **-reset** option. The default is for the tool to select a type.
- width** *width*
Specifies the width of the ring's strip. This option is required if you do not use the **-reset** option. The default is for the tool to select a width.
- offset** *offset*
Specifies the distance, from the starting side, from the edge of the object around which the ring is made and the center of the ring. The default is for the tool to select an offset.
- sides** {top | left | right | bottom}
Specifies the sides around the ring. The default is to use all sides.
- layer**
Specifies the name of the metal layer in which the macro ring is to be drawn. This name is defined in the Synopsys physical library (.plib). This option

is required if you do not use the **-reset** option. The default is for the tool to select a layer.

DESCRIPTION

This command specifies the ring options to be used to create power and ground rings around cores, macros, macro arrays, or designs designated by using the the **create_placement**, **physopt**, and **compile_physical** commands when they specify the **-mpc** option. You should issue these commands after the design and physical libraries are loaded.

During the flow, the options are read and the trunks are drawn based on the options provided. After the ring pnets are generated, you can view them by using the **report_net** command with the **-only_physical** option.

Multicorner-Multimode Support

This command is not supported in a multi-scenario design flow.

EXAMPLES

The following example shows how to specify options for a ring on the metal4 layer using the widths and pitch around all sides of the core.

```
prompt> set_mpc_ring_option -layer Metal4 -name ring \  
-width 3.00 -type gnd -offset 5.00 \  
-sides {TOP LEFT RIGHT BOTTOM} [current_design]
```

SEE ALSO

```
create_placement(2)  
remove_net(2)  
report_mpc_macro_options(2)  
report_mpc_options(2)  
report_mpc_pnet_options(2)  
report_mpc_port_options(2)  
report_mpc_ring_options(2)  
report_net(2)  
set_mpc_macro_options(2)  
set_mpc_options(2)
```

set_mtcmos_pna_strategy

Sets options for power network synthesis (PNS) and power network analysis (PNA), when exploring, replacing, and optimizing MTCMOS cell size. These settings are not persistent in the Milkyway database.

SYNTAX

```
status set_mtcmos_pna_strategy
[-reset]
[-power_budget budget]
[-voltage_supply voltage_supply]
[-lowest_voltage_drop]
[-target_voltage_drop target_voltage]
[-pad_lib_cell pad_names]
[-read_pad_cell_file cell_file]
[-read_pad_lib_cell_file lib_cell_file]
[-use_pins_as_pads]
[-use_strap_ends_as_pads]
[-create_virtual_rails layer_names]
[-synthesize_voltage_areas voltage_area_list]
[-disable_snap_to_row_and_tile]
[-relative_to_voltage_area]
[-pattern {normal | stagger}]
[-voltage_areas list]
```

Data Types

<i>budget</i>	float
<i>voltage_supply</i>	float
<i>target_voltage</i>	float
<i>pad_names</i>	string
<i>cell_file</i>	string
<i>lib_cell_file</i>	string
<i>layer_names</i>	string
<i>voltage_area_list</i>	collection or list

ARGUMENTS

-reset

Reset the specified strategy value. If no strategy name is specified, all the strategy values will be reset.

-power_budget *budget*

Specifies a total power budget to perform constraint-driven power network analysis and power net synthesis. The power budget is divided among the instances according to their size. The power unit is in milliwatts. For hard macros or standard cells, the power budget is computed based on the percentage of the total area. For a hierarchical block, the power budget is computed based on the sum of all the cells and blocks inside it. The default is 1000.

-voltage_supply *voltage_supply*

Specifies the voltage of the net you want analyzed or Synthesized the power

set_mtcmos_pna_strategy

pads. The voltage unit is volts. The default is 1.5.

-lowest_voltage_drop

Specifies the lowest IR drop option if you want to direct the synthesize engine to synthesize a power plan or pad that would give the lowest possible IR drop value based on user-defined constraints. Default is false.

-target_voltage_drop *target_voltage*

Specifies a target IR drop value. Default value is 10% of the power pad voltage.

-read_pad_cell_file *cell_file*

Specify the pad cell file name. Default is null.

The format is: *cell_name net_name* (The *net_name* is optional.)

For example, VDD1 VDD

Note:

If the net name is not specified, the command assumes that the specified pad cell is used for both power and ground nets.

-read_pad_lib_cell_file *lib_cell_file*

Specify the pad lib cell reference file name. Default is null.

The format is: *lib_cell net* (The *net* is optional.)

For example, VDD.FRAME VDD

Note:

If the net name is not specified, the command assumes that the specified pad lib cell is used for both power and ground nets.

-use_pins_as_pads

Treats pins as pads in block level design. Default is false.

-use_strap_ends_as_pads

Uses the ends of straps as pads for block level design that has no pins or pads. Default is false.

-create_virtual_rails *layer_names*

During the floorplanning stages, metal 1 straps for the standard cell pin connections are usually not available. Power network analysis uses this option to create virtual pseudo straps that represent the metal 1 straps for the standard cell pin connections. During power planning, it then analyzes the IR (voltage) drop and electromigration to predict what the effects might be when the metal 1 straps are available. Creates pseudo horizontal straps with specified layers for standard cell pin connections based on the row information in the database. Default is null.

-synthesize_voltage_areas *voltage_area_list*

Specify the voltage areas need be synthesized in PNS.

-disable_snap_to_row_and_tile

Disable snap MTCMOS cells to row when placement.

-relative_to_voltage_area

Coordinates are specified relative to lower left corner of voltage area's bbox.

`-pattern {normal | stagger}`
Pattern used for header/footer cell placement.

DESCRIPTION

Sets parameters of power network synthesis (PNS) and power network analysis (PNA), when exploring, replacing, optimizing MTCMOS cell size. These settings are not persistent in the Milkyway database. PNA and PNS operations are used in high-level commands such as `explore_header_footer` and `replace_header_footer`. This command is used to set the options needed to control the operations of PNA and PNS when running the high-level commands.

Options `-read_pad_cell_file`, `-lowest_voltage_drop`, `-target_voltage_drop`, `-use_strap_ends_as_pads`, `-synthesize_voltage_areas` only work with power network synthesis related operations. The rest options work for power network synthesis and analysis related operations.

Multicorner-Multimode Support

This command is not supported in a multi-scenario design flow.

EXAMPLES

The following example set use strap ends as pads, and specify 50 milliwatts power budget, 0.5 volts supply when PNA & PNS.

```
prompt> set_mtcmos_pna_strategy \  
      -power_budget 50 \  
      -voltage_supply 0.5 \  
      -use_strap_ends_as_pads
```

SEE ALSO

`analyze_fp_rail(2)`
`synthesize_fp_rail(2)`
`report_mtcmos_pna_strategy(2)`

set_multicycle_path

Modifies the single-cycle timing relationship of a constrained path.

SYNTAX

```
integer set_multicycle_path  
path_multiplier  
[-rise | -fall]  
[-setup | -hold]  
[-start | -end]  
[-from from_list  
  | -rise_from rise_from_list  
  | -fall_from fall_from_list]  
[-through through_list]  
[-rise_through rise_through_list]  
[-fall_through fall_through_list]  
[-to to_list  
  | -rise_to rise_to_list  
  | -fall_to fall_to_list]  
[-reset_path]
```

Data Types

<i>path_multiplier</i>	integer
<i>from_list</i>	list
<i>rise_from_list</i>	list
<i>fall_from_list</i>	list
<i>through_list</i>	list
<i>rise_through_list</i>	list
<i>fall_through_list</i>	list
<i>to_list</i>	list
<i>rise_to_list</i>	list
<i>fall_to_list</i>	list

ARGUMENTS

path_multiplier

Specifies the number of cycles that the data path must have for setup or hold relative to the startpoint or endpoint clock before data is required at the endpoint. When used with **-setup**, this value is applied to setup path calculations. When used with **-hold**, this value is applied to hold path calculations. If neither **-hold** nor **-setup** are specified, *path_multiplier* is used for setup, and 0 is used for hold. Changing the multiplier for setup also affects the hold check.

-rise

Specifies that rising path delays are affected by *path_multiplier*. The default is that both rising and falling delays are affected. Rise refers to a rising value at the path endpoint.

-fall

Specifies that falling path delays are affected by *path_multiplier*. The

default is that both rising and falling delays are affected. Fall refers to a falling value at the path endpoint.

-setup

Specifies that *path_multiplier* is used for setup calculations.

-hold

Specifies that *path_multiplier* is used for hold calculations.

-start | -end

Specifies whether the multicycle information is relative to the period of the start clock or the end clock. These options are only needed for multifrequency designs; otherwise start and end are equivalent. The start clock is the clock source related to the register or primary input at the path startpoint. The end clock is the clock source related to the register or primary output at the path endpoint. The default is to move the setup check relative to the end clock, and the hold check relative to the start clock. A setup multiplier of 2 with **-end** moves the relation forward one cycle of the end clock. A setup multiplier of 2 with **-start** moves the relation backward one cycle of the start clock. A hold multiplier of 1 with **-start** moves the relation forward one cycle of the start clock. A hold multiplier of 1 with **-end** moves the relation backward one cycle of the end clock.

-from from_list

Lists the names of clocks, ports, pins, or cells to use to find path startpoints. If a clock is specified, all registers and primary inputs related to that clock are used as path startpoints. If a cell is specified a cell, one path startpoint on that cell is affected.

-rise_from rise_from_list

Same as the **-from** option, except that the path must rise from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by rising edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of **-from**, **-rise_from**, and **-fall_from**.

-fall_from fall_from_list

Same as the **-from** option, except that the path must fall from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of **-from**, **-rise_from**, and **-fall_from**.

-through through_list

Lists the path throughpoints (port, pin, or leaf cell names) of the current design. The multicycle values apply only to paths that pass through one of the points in the *through_list*. If more than one object is included, the objects must be enclosed either in double quotation marks (") or in braces ({}). If you specify the **-through** option multiple times, the multicycle values apply to paths that pass through a member of each *through_list* in the order the lists were given. The path must first pass through a member of the first *through_list*, then through a member of the second list, and so on for every through list specified. If the **-through** option is used in combination with the **-from** or **-to** options, the multicycle values apply only if the **-from**

or **-to** conditions are satisfied and the **-through** conditions are satisfied.

-rise_through *rise_through_list*

Same as the **-through** option, but applies only to paths with a rising transition at the specified objects. You can specify **-rise_through** more than once in a single command invocation as with the **-through** option.

-fall_through *fall_through_list*

Same as the **-through** option, but applies only to paths with a falling transition at the specified objects. You can specify **-fall_through** more than once in a single command invocation as with the **-through** option.

-to *to_list*

Lists the names of clocks, ports, pins or cells to use to find path endpoints. If a clock is specified, all registers and primary outputs related to that clock are used as path endpoints. If a cell is specified, one path endpoint on that cell is affected.

-rise_to *rise_to_list*

Same as the **-to** option, but applies only to paths rising at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths captured by rising edge of the clock at clock source, taking into account any logical inversions along the clock path. You can use only one of **-to**, **-rise_to**, and **-fall_to**.

-fall_to *fall_to_list*

Same as the **-to** option, but applies only to paths falling at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of **-to**, **-rise_to**, and **-fall_to**.

-reset_path

Removes existing point-to-point exception information on the specified paths. When used only with **-to**, all paths leading to the specified endpoints are reset. When used only with **-from**, all paths leading from the specified startpoints are reset. When used with **-from** and **-to**, only paths between those points are reset. Only information of the same rise/fall, setup/hold type is reset. This is equivalent to using the **reset_path** command with similar arguments before issuing **set_multicycle_path**.

DESCRIPTION

The **set_multicycle_path** command specifies that designated timing paths in the current design have nondefault setup or hold relations.

The synthesis timing engine applies certain rules to determine single-cycle timing relationships for paths between clocked elements. The rules are based on active edges. For flip-flops, a single active edge both launches and captures data. For latches, the open edge is used to launch data, and the close edge is used to latch data.

The setup check ensures that the correct data signal is available on destination registers in time to be properly latched.

The rule for setup is that for multifrequency designs, there can be multiple setup relations between two clocks. For every latch edge of the destination clock, find the nearest launch edge which precedes each capture edge. The smallest difference of (setup_latch_edge - setup_launch_edge) determines the maximum delay requirement for this path.

This is known as single-cycle setup. You can override this default relationship using **set_multicycle_path** or **set_max_delay**. You can apply these commands to clocks, pins, ports, or cells. For example, setting the setup path multiplier to 2 with the **set_multicycle_path** command delays the latch edge one clock pulse. Changing the setup multiplier also affects the default hold check.

The hold check verifies the following items:

- Data from the source clock edge that follows the setup launch edge must not be latched by the setup latch edge.
- Data from the setup launch edge must not be latched by the destination clock edge that precedes the setup latch edge.

The hold check is determined relative to each valid setup relationship, after applying multicycle path multipliers. The most restrictive (largest) difference of (hold_latch_edge - hold_launch_edge) is used as a minimum delay requirement for the path.

The hold relation is conservative. In some cases you might need to override the default hold relation. For multifrequency designs, it is more straightforward to use the **set_min_delay** command to override the default instead of using **set_multicycle_path -hold**.

Setting *path_multiplier* for setup moves the setup check in time by an integer number of active edges. For example, specifying *path_multiplier* of 2 for setup implies a 2 cycle data path.

Most often, the setup check is moved relative to the end clock. This move changes the data latch time at the path endpoint. In multifrequency designs, use the following command to move the data launch time backward:

set_multicycle_path -setup -start

To move the hold relation relative to the end clock use the command

set_multicycle_path -hold -end

If you do not specify **-setup** or **-hold**, the setup relation is set to *path_multiplier* and the hold relation is set to 0.

If you specify **-setup**, the setup multiplier is set to *path_multiplier*. The hold multiplier is unaffected.

If you specify **-hold**, the hold multiplier is set to *path_multiplier*. The setup multiplier is unaffected.

There are separate setup and hold multipliers for rise and fall. In most cases, these are set to the same value. You can use the **-rise** or **-fall** options to apply different values.

The **set_multicycle_path** command is a point-to-point timing exception command. If a path satisfies multiple timing exceptions, the following rules are used in order to determine which exceptions take effect. Rules referring to **-from** apply equally to **-rise_from** and **-fall_from**, and similarly for the rise and fall options of **-through** and **-to**.

- Two **group_path** commands can conflict with each other, but a **group_path** exception alone does not conflict with another type of exception. So the remaining rules apply for two **group_path** exceptions, or two non-**group_path** exceptions.
- If both exceptions are **set_false_paths**, there is no conflict.
- If one exception is a **set_max_delay** and the other is **set_min_delay**, there is no conflict.
- If one exception is a **set_multicycle_path -hold** and the other is **set_multicycle_path -setup**, there is no conflict.
- If one exception is a **set_false_path** and the other is not, the **set_false_path** takes precedence.
- If one exception is a **set_max_delay** and the other is not, the **set_max_delay** takes precedence.
- If one exception is a **set_min_delay** and the other is not, the **set_min_delay** takes precedence.
- If one exception has a **-from** pin or **-from** cell and the other does not, the former takes precedence.
- If one exception has a **-to** pin or **-to** cell and the other does not, the former takes precedence.
- If one exception has any **-through** points and the other does not, the former takes precedence.
- If one exception has a **-from** clock and the other does not, the former takes precedence.
- If one exception has a **-to** clock and the other does not, the former takes precedence.
- The exception with the more restrictive constraint then takes precedence. For **set_max_delay** and **set_multicycle_path -setup**, this is the constraint with the lower value. For **set_min_delay** and **set_multicycle_path -hold**, it is the constraint with the higher value.

To undo a **set_multicycle_path** command, use **reset_path** or **reset_design**.

Use **set_false_path** to disable setup or hold calculations for paths.

Use **report_timing_requirements** to list the point-to-point exceptions on a design.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example sets all paths between latch1b and latch2d to 2 cycle paths for setup. Hold is measured at the previous edge of the clock at latch2d.

```
prompt> set_multicycle_path 2 -from {latch1b} -to {latch2d}
```

The following example moves the hold check to the preceding edge of the start clock:

```
prompt> set_multicycle_path -1 -from {latch1b} -to {latch2d}
```

The following example is a two-phase level-sensitive design, where the designer expects paths from phil to phil to be zero cycles:

```
prompt> set_multicycle_path 0 -from phil -to phil
```

The following example uses **-start** to specify a 3-cycle path relative to the clock at the path startpoint. Clock sources are specified, affecting all sequential elements clocked by that clock, or ports with input or output delay relative to that clock.

```
prompt> set_multicycle_path 3 -start -from {clk50mhz} -to {clk10mhz}
```

The following example affects all paths that pass first through the U1 or U2 cell and later pass through the U3 cell. The path resulting in a rise transition at an endpoint is constrained to 2 cycles, but falling paths are constrained to 1 cycle.

```
prompt> set_multicycle_path 2 -rise -through {U1, U2} -through {U3}
```

```
prompt> set_multicycle_path 1 -fall -through {U1, U2} -through {U3}
```

The following multifrequency example shows a 20ns clock to a 10ns clock, with a multicycle path of 2. Assume a path from ff1 (clocked by clk20) to ff2 (clocked by clk10).

```
prompt> create_clock -period 20 -waveform {0 10} clk20
```

```
prompt> create_clock -period 10 -waveform {0 5} clk10
```

```
prompt> set_multicycle_path 2 -setup -from ff1/CP -to ff2/D
```

The single-cycle setup relation is from the CLK1 edge at 0ns to the CLK2 edge at 10ns. With the multicycle path, the setup relation is now 20ns (from the CLK1 edge at 0ns to CLK2 edge at 20ns). The hold relations are determined according to that setup relation.

- Data from the source clock edge that follows the setup launch edge must not be latched by the setup latch edge. This implies a hold relation of 0ns (from CLK1 edge at 20ns to CLK2 edge at 20ns).

- Data from the setup launch edge must not be latched by the destination clock edge that precedes the setup latch edge. This implies a hold relation of 10ns (from CLK1 edge at 0ns to CLK2 edge at 10ns).

The most restrictive (largest) hold relation is used, so the minimum delay requirement for this path is 10ns. This is a conservative check that might not apply to certain designs. Often you know that the destination register is disabled for the clock edge at 10ns. You can modify this default hold relation with the following command to get a 0ns requirement:

```
prompt> set_min_delay 0 -from ff1/CP -to ff2/D
```

However, an easier approach is to use the following command:

```
prompt> set_multicycle_path 1 -hold -end -from ff1/CP -to ff2/D
```

SEE ALSO

```
current_design(2)  
report_timing_requirements(2)  
reset_design(2)  
reset_path(2)  
set_false_path(2)  
set_max_delay(2)  
set_min_delay(2)
```

set_mw_lib_reference

Sets the reference library for the Milkyway library.

SYNTAX

```
status_value set_mw_lib_reference  
[-mw_reference_library lib_list]  
[-reference_control_file file_name]  
libName
```

Data Types

<i>lib_list</i>	string
<i>file_name</i>	string
<i>libName</i>	string

ARGUMENTS

-mw_reference_library *lib_list*
Specifies a list of libraries to be set as reference libraries for the Milkyway library.
This argument and **-reference_control_file** are mutually exclusive.

-reference_control_file *file_name*
Specifies a reference control file containing information to set the reference libraries for the Milkyway library.
This argument and **-mw_reference_library** are mutually exclusive.

libName
Specifies the Milkyway library to be worked on.

DESCRIPTION

Sets or changes the reference libraries for the Milkyway library. The Milkyway library being updated must be closed for the command to work correctly. The **-reference_control_file** and **-mw_reference_library** options are mutually exclusive, and at least one of the two must be specified.

A status indicating success or failure is returned.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets the reference libraries with a list:

```
set_mw_lib_reference  
2608
```

```
prompt> set_mw_lib_reference -mw_reference_library {./lib/ref1 ./lib/ref2} design
```

SEE ALSO

```
create_mw_lib(2)  
set_mw_technology_file(2)
```

set_mw_technology_file

Sets the technology file of the Milkyway library.

SYNTAX

```
status_value set_mw_technology_file  
[-technology tech_file]  
[-plib plib_file]  
[-alf alf_file]  
libName
```

Data Types

<i>tech_file</i>	string
<i>plib_file</i>	string
<i>alf_file</i>	string
<i>libName</i>	string

ARGUMENTS

-technology *tech_file*
Specifies a new technology file to use with the Milkyway library. The old technology information is completely replaced. Ensure that the new technology information is compatible, or else you must rebuild or recreate the Milkyway library accordingly.

-plib *plib_file*
Specifies a new plib file to use with the Milkyway library. If the plib file is an incremental plib file , it will load the technology file in plib incrementally to update library technology. If it is a complete technology plib file, library technology info will be replaced completely. Ensure that the new technology information is compatible, or else you must rebuild or recreate the Milkyway library accordingly.

-alf *alf_file*
Specifies a new alf file to use with the Milkyway library. This option is for purpose of importing signal EM data into the library. If there is existing signal EM info in the library, it will be completely replaced by the data in the new alf file. Ensure that the new signal EM information is compatible with library technology info. Signal EM data in the alf file is updated to the library immediately, it does not have impact to opened design.

libName
Specifies the Milkyway library to be updated. The value of *mw_lib* must be a valid library name.

DESCRIPTION

This command sets the technology file for a Milkyway library.

The command returns a status indicating success or failure.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets the technology file of the library design to *new.tf*:

```
prompt> set_mw_technology_file -technology new.tf design  
1
```

SEE ALSO

```
close_mw_lib(2)  
copy_mw_lib(2)  
create_mw_lib(2)  
current_mw_lib(2)  
open_mw_lib(2)  
rename_mw_lib(2)  
set_mw_lib_reference(2)  
write_mw_lib_files(2)
```

set_name

Changes the name of the specified netlist object.

SYNTAX

```
status set_name  
-type net | port | cell  
object  
-name local_name  
[-top_design]
```

Data Types

<i>object</i>	collection
<i>local_name</i>	string

ARGUMENTS

-type net | port | cell
Specifies whether the specified object is a net, a hierarchical port, or a leaf cell instance.
This is a required option.

object
Specifies the single netlist object to which the **set_name** command is applied.
The *object* argument can be a single object collection returned by a query command, such as **get_nets**, or a netlist object name string.
This is a required argument.

-name *local_name*
Specifies the new local name, which is the name local to the parent hierarchical cell.
This is a required option.

-top_design
Allows changes to the top-level design name.

DESCRIPTION

Changes the name that is local to the parent hierarchical cell.

The **set_name** command changes the local names of nets, ports, and cells in a design. A netlist object name can consist of full hierarchical path to the object and the **set_name** command does not change this hierarchical path information.

EXAMPLES

The following example uses the **set_name** command to change the name of net M1/N1 to M1/N2 in the M1 hierarchical cell by changing its local name from N1 to N2.

```
prompt> set_name -type net M1/N1 -name N2
prompt> get_nets M1/N2
{M1/N2}
```

The following example uses the **set_name** command to change the name of the M1/C1 leaf cell to M1/C2 in the M1 hierarchical cell by changing its local name from C1 to C2.

```
prompt> set_name -type cell [get_cells M1/C1] -name C2
prompt> get_cells M1/*
{M1/C2}
```

The following example uses the **set_name** command to change the name of the top-level design from TOP1 to TOP2. You can see the new top-level design name by using the **write_verilog** command.

```
prompt> set_name -type cell TOP1 -name TOP2 -top_design
prompt> write_verilog my_netlist.v
```

SEE ALSO

```
change_names(2)
define_name_rules(2)
report_name_rules(2)
report_names(2)
default_name_rules(3)
```

set_net_aggressors

Sets the aggressor nets for a victim net.

SYNTAX

```
status set_net_aggressors  
-victim_net net  
-aggressor_nets collection_of_nets
```

Data Types

collection_of_nets list

ARGUMENTS

```
-victim_net net  
           Specified a victim net.  
  
-aggressor_nets collection_of_nets  
               Specified one or multiple aggressor nets
```

DESCRIPTION

Specified a victim net and its associated aggressor nets. This information will be used during the routing stage to minimize crosstalk effect.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example is to set all nets, which are initialized with "NET_INV" and "OUT", to be aggressor nets for a victim net, NET_ANTENNA.

```
prompt> set_net_aggressors -victim_net NET_ANTENNA -  
aggressor_nets {NET_INV* OUT*}
```

SEE ALSO

set_net_routing_layer_constraints

Assigns routing layer constraints to the specified nets.

SYNTAX

```
status set_net_routing_layer_constraints  
list_of_nets  
-min_layer_name minimum_routing_layer_name  
-max_layer_name maximum_routing_layer_name
```

Data Types

<i>list_of_nets</i>	collection
<i>minimum_routing_layer_name</i>	string
<i>maximum_routing_layer_name</i>	string

ARGUMENTS

list_of_nets
Specifies the nets to which the routing layer constraints are applied. This is a required argument.

-min_layer_name *minimum_routing_layer_name*
Specifies the minimum routing layer. You can specify only a single layer name. This is a required option.

-max_layer_name *maximum_routing_layer_name*
Specifies the maximum routing layer. You can specify only a single layer name. This is a required option.

DESCRIPTION

This command assigns routing layer constraints to the specified nets.

The routing layer names you specify must exist in the physical library as well as in the design. In addition, the maximum routing layer must be higher than the minimum routing layer.

If a net already has routing layer constraints set, the tool issues a warning message and overrides the existing routing layer constraints.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example assigns routing layer constraints to the net named netA.

```
prompt> set_net_routing_layer_constraints {netA}  
-min_layer_name metal1 -max_layer_name metal3
```

SEE ALSO

```
report_net_routing_layer_constraints(2)  
remove_net_routing_layer_constraints(2)  
route_opt(2)
```

set_net_routing_rule

Assigns a nondefault or default routing rule to specific nets.

SYNTAX

```
integer set_net_routing_rule  
-rule rule_name  
list_of_nets  
[-timing_driven_spacing]  
[-reroute normal | minorchange | freeze]  
[-top_layer_probe AnyPort | OutPort | AllPort]
```

Data Types

<i>rule_name</i>	string
<i>list_of_nets</i>	string

ARGUMENTS

-rule *rule_name*
Specifies the name of the routing rule to be assigned. The *rule_name* can be either a name of an existing nondefault rule or a keyword -default-, which means that you must assign the default routing.

list_of_nets
Lists the net names for which a routing rule is assigned.

DESCRIPTION

This command assigns a nondefault or default routing rule to specific nets. Nondefault rules are defined in the physical library or by using the **define_routing_rule** command and can have associated user-defined width and spacing rules and via types. These rules govern nondefault wiring such as the wide wires and vias that are typically used for clock signals and other sensitive signals.

Library-specific rules are defined in the physical library and stored in the

If nondefault routing rules are not defined in the physical library provided with the design database, you can define them in a separate .plib file that can be updated into the existing .pdb file by using the **update_lib** command. For more information about the **update_lib** command, see the man page.

You can define design-specific rules by using the **define_routing_rule** command and storing them in the design .db file.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example, assigns a nondefault rule named *WideMetal* to a net named *B_CLK*.

```
prompt> set_net_routing_rule {B_CLK} -rule WideMetal
```

SEE ALSO

```
update_lib(2)  
define_routing_rule(2)  
report_routing_rules(2)
```


set_object_boundary

Sets the boundary of a set of objects to a bounding box or a list of points.

SYNTAX

```
status set_object_boundary
{-bbox rect | -boundary boundary}
[-keep_placement]
[-keep_pad_to_core_distance]
[-ignore_fixed]
objects
```

Data Types

<i>rect</i>	x1 y1 x2 y2
<i>objects</i>	collection

ARGUMENTS

-bbox *rect*
Specifies the bounding box to use as the boundary.

-boundary *boundary*
Specifies the list of points to use as the boundary.
Note: There must be at least four points.

-keep_placement
This option is effective only when applied to the core or the die. It will be ignored if applied on any other object.
Cells in the core and IO pad cells are re-placed if necessary. The relative positions of the cells are not guaranteed to be preserved, although an attempt will be made to preserve them as much as possible. Furthermore, the output of the re-placement is not legalized. The user needs to explicitly legalize the placement using `legalize_placement`.
The default is **false**.

-keep_pad_to_core_distance
This option is effective only when applied to the core or the die. It will be ignored if applied on any other object.
This maintains the distance between the core and the pad cells and by implication the distance between the core and the die, thus forcing changes to the core and the die to be made in tandem with each other. If pad cells are absent then the core-to-die distance will still be maintained.
The default is **false**.

-ignore_fixed
Normally fixed objects can not have their boundary set. If this flag is supplied then fixed objects will also be changed.

objects
Specifies the objects to have their boundary set.

DESCRIPTION

This command sets the boundary of one or more objects to the specified bounding box or set rectilinear points.

Only objects which can be resized can have their bounding box set.

Only objects which can be rectilinear can have a boundary of more than four points.

See `get_edit_property(2)` man page for details on which objects can be resized and which objects can be rectilinear.

NOTES

Fixed objects will not have their boundary set unless `-ignore_fixed` is specified.

Snapping is done automatically using global snap settings.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following examples set the boundary of the selected object to (0 0) (100 100).

```
prompt> set_object_boundary -bbox {0 0 100 100} [get_selection]
```

The following examples set the boundary of the selected object to a L shape.

```
prompt> set_object_boundary \  
-boundary {{0 0} {100 0} {100 50} {50 50} {50 100} {0 100}} \  
[get_selection]
```

SEE ALSO

`set_object_shape(2)`

set_object_fixed_edit

Sets the fixed state for one or more objects.

SYNTAX

```
status set_object_fixed_edit  
objects  
fixed
```

ARGUMENTS

objects
List of objects to set fixed for

fixed
List of one or more Boolean fixed flags

DESCRIPTION

Set the fixed state for the supplied objects. Each object is set to the value of the flag at the corresponding position in the fixed list. If the fixed list is shorter than the object list the last value in the fixed list is propagated to achieve the required size. If the fixed list is longer than the object list the extra values are silently ignored.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example fixes all selected objects

```
> set_object_fixed_edit [get_selection] 1
```

SEE ALSO

`get_object_fixed_edit(2)`

set_object_shape

Sets the shape of a set of objects to one of the standard rectilinear shapes.

Only objects which can be resized can be set to a "rect" shape.

Only objects which can be rectilinear can be set to a "l", "t", "u" or "cross" shape.

See `get_edit_property(2)` man page for details on which objects can be resized and which objects can be rectilinear.

SYNTAX

```
status set_object_shape
-shape {rect | l | t | u | cross}
-lengths real_list
[{-utilization real
  | -area real
  | -keep_area}]
[-rotate {0 | 90 | 180 | 270}]
[-keep_placement]
[-keep_pad_to_core_distance]
[-ignore_fixed]
objects
```

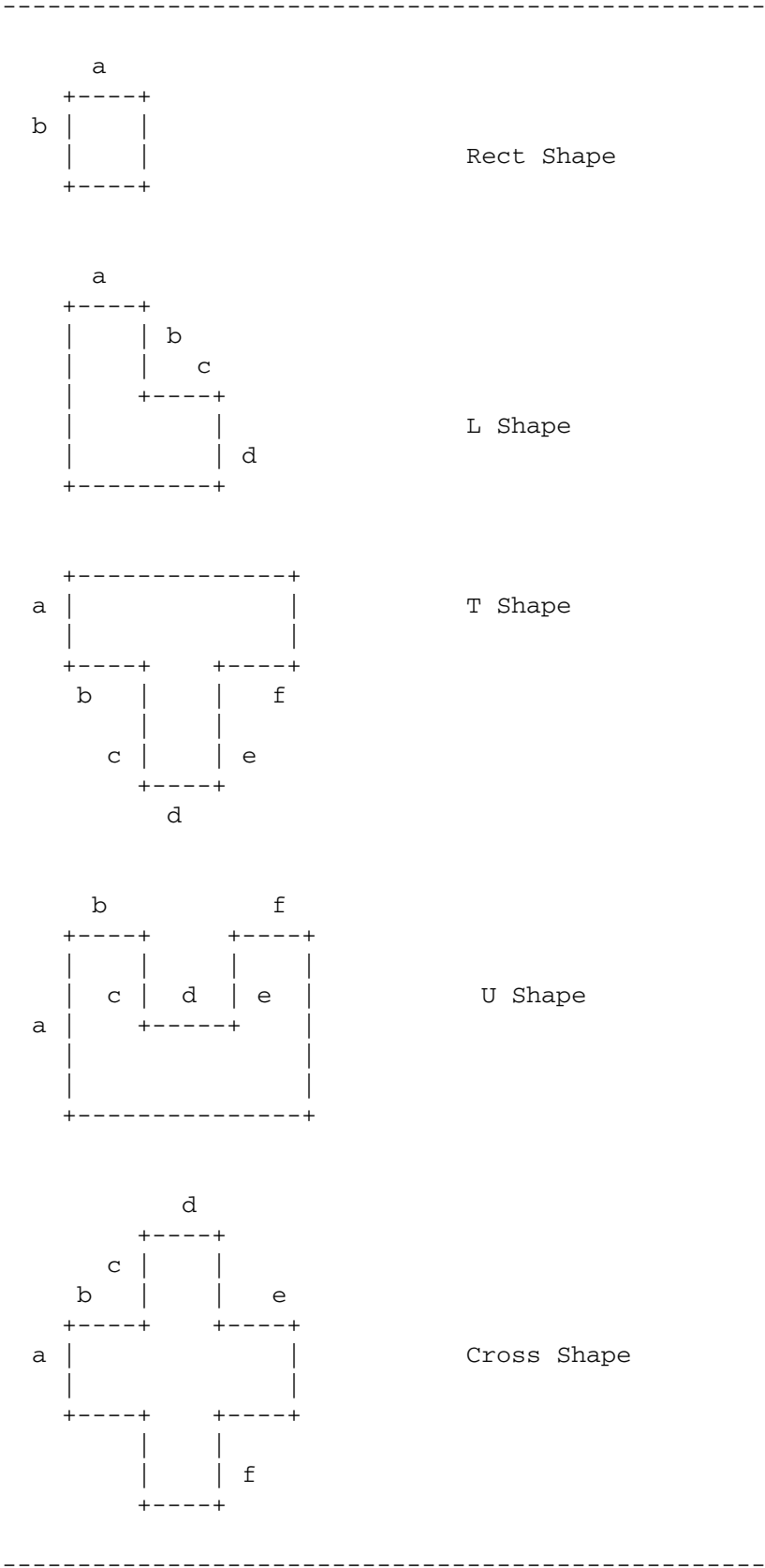
Data Types

objects collection

ARGUMENTS

```
-shape {rect | l | t | u | cross}
    Specifies the shape to use.
    This is either a Rectangular, L, T, U or Cross shape.

-lengths real_list
    Specifies the absolute lengths in microns or length ratios of the sides of
    the shape.
    Absolute lengths are assumed unless one of -utilization, -area or -keep_area
    is specified.
    If length ratios are specified then the resultant shape will have sides with
    the corresponding ratios e.g. if the ratios are all one then all lengths will
    be the same. If one ratio is two and the others are one then that side will
    be twice as long as the others.
    Each shape requires a different number of values as described below:
    &. rect      2
    &. l         4
    &. t         6
    &. u         6
    &. cross    6
    See diagrams below for the meaning of parameter values for each shape (a is
    first value, b is second value etc.):
```



-utilization real

Specifies that the new shape should have the specified utilization.
Utilization is defined as `physical_area` divided by placement area and is most commonly used for plan groups and utilization where the enclosed area can be larger than the associated child physical objects.
Utilization must be positive and non-zero.

Note: this implies that lengths are ratios and not absolute values.
The default is **0.7**.

-area real

Specifies that the new shape should have the specified area.

Note: this implies that lengths are ratios and not absolute values.
The default is **1.0**.

-keep_area

Specifies that the new shape should have the same area as the object current has.

Note: this implies that lengths are ratios and not absolute values.

-rotate {0 | 90 | 180 | 270}

Specifies that the base shape is geometrically rotated from the default orientation (See diagram above) before being applied to the object.

Note: this option cannot be used with `-shape rect`

Note: 0 is the default value so this option only needs to be specified for 90, 180 and 270 rotation.

-keep_placement

This option is effective only when applied to the core or the die. It will be ignored if applied on any other object.

Cells in the core and IO pad cells are re-placed if necessary. The relative positions of the cells are not guaranteed to be preserved, although an attempt will be made to preserve them as much as possible. Furthermore, the output of the re-placement is not legalized. The user needs to explicitly legalize the placement using `legalize_placement`.

The default is **false**.

-keep_pad_to_core_distance

This option is effective only when applied to the core or the die. It will be ignored if applied on any other object.

This maintains the distance between the core and the pad cells and by implication the distance between the core and the die, thus forcing changes to the core and the die to be made in tandem with each other. If pad cells are absent then the core-to-die distance will still be maintained.

The default is **false**.

-ignore_fixed

Normally fixed objects can not have their shape set.

If this flag is supplied then fixed objects will also be changed.

objects

list of objects for which the boundary shape will be set.

DESCRIPTION

This command sets the shape of the boundary of one or more objects.

Only objects that can be resized can be set to a "rect" shape.

Only objects that can be rectilinear can be set to a "l", "t", "u" or "cross" shape.

See the `get_edit_property(2)` man page for details on which object types can be resized and which object types can be rectilinear.

NOTES

Snapping is done automatically using global snap settings.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets the boundary of the selected objects to an L shape with all sides of equal length and a utilization of 0.7

```
> set_object_shape -shape l -params {1 1 1 1} -utilization 0.7 [get_selection]
```

SEE ALSO

`set_object_boundary(2)`

set_object_snap_type

Sets the object snap grid for a class of objects, or enables or disables snapping for all objects.

SYNTAX

```
old_value set_object_snap_type  
-class snap_class -snap snap_type  
    | -enabled true | false
```

Data Types

```
snap_class      string  
snap_type       string
```

ARGUMENTS

```
-class snap_class  
    Specifies the object class for which you are setting or changing the snap  
    grid. When using this option, you must also use the -snap option. You cannot  
    use -class with the -enabled option.  
    The snap_class must be one of the following object types:  
    soft_macro      Soft macro  
    plan_group      Plan group  
    movebound       Movebound  
    hard_macro      Hard macro  
    standard_cell    Standard cell  
    routing          Net shape, via, and via array  
    shape           Polygon  
    pin             Top-level pin, soft macro pin, and pin guide  
    voltage_area     Voltage area  
    placement_blockage Placement blockage  
    wiring_keepout   Wiring keepout  
    rp_group         Relative placement group  
  
-snap snap_type  
    Sets the snap grid for the specified class of objects. When using this option,  
    you must also use the -class option. You cannot use -snap with the -enabled  
    option.  
    The snap_type must be one of the following grid types:  
    litho           Design grid (also known as minimum grid or lithographic grid)  
    row             Row edge  
    row_tile        Row edge and unit tile  
    row_site        Row edge and placement site  
    mid_row         Mid-row (mostly for soft macro placement)  
    mid_row_tile    Mid-row and unit tile  
    mid_row_site    Mid-row and placement site  
    flip_chip       Flip-chip grid (for flip chips)  
    unit_tile       Unit tile  
    wiretrack       Wire track (for pins, wires, paths, and terminals)  
    halfwiretrack   Mid-wire track (for pins, wires, paths, and terminals)
```


user User grid

Note that setting the snap type to litho is equivalent to turning off snapping for the specified object class.

The default snap types for the different object classes are

soft_macro	mid_row
plan_group	mid_row
movebound	row_site
hard_macro	row
standard_cell	row_site
routing	wiretrack
shape	litho
pin	wiretrack
voltage_area	litho
placement_blockage	litho
wiring_keepout	litho
rp_group	litho

-enabled true | false

Enables snapping for all object classes when set to true, or disables snapping for all object classes when set to false. You cannot use -enabled with the -class and -snap options.

When you use this option, the command returns the previous state: 0 (disabled) or 1 (enabled). For example, if you use **-enabled true** when snapping is currently disabled, the command returns the value 0. This does not mean that the command has failed. If you use **-enabled true** when snapping is enabled, the command returns the value 1.

DESCRIPTION

This command either sets the object snap grid for a class of objects or controls whether object snapping is enabled or disabled for all object types.

Use the -class and -snap options to set the snap grid for a specific class of objects. All of the editing tools and editing commands use this snap grid to set the positions and geometries of the objects during editing operations. You can set a different snap type for each object class.

Use the -enabled option to enable or disable object snapping for all object types.

Note that the command requires either the -class option or the -enabled option, which are mutually exclusive, and that the -class and -snap options must be used together.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets the snap type for hard macro objects to the user grid:

```
prompt> set_object_snap_type -class hard_macro -snap user
```

litho

The following example disables snapping for all object types:

```
prompt> set_object_snap_type -enabled false  
1
```

SEE ALSO

```
move_objects(2)  
resize_objects(2)  
rotate_objects(2)  
align_objects(2)  
distribute_objects(2)  
expand_objects(2)  
flip_objects(2)  
get_object_snap_type(2)
```

set_operand_isolation_scope

Specifies whether a design or instance should be included or excluded for operand isolation processing.

SYNTAX

```
status set_operand_isolation_scope  
object_list  
[true | false]
```

Data Types

object_list list

ARGUMENTS

object_list
Specifies a list of designs or instances for which operand isolation processing is to be enabled or disabled.

true | false
Specifies whether to enable (**true**) or disable (**false**) operand isolation processing for the *object_list*. The default value is **true**, which enables operand isolation processing.

DESCRIPTION

Use of the **set_operand_isolation_scope** command is not recommended. This feature will be obsolete in a future release. The **set_operand_isolation_scope** command sets an attribute on a list of designs or instances. This attribute controls whether or not an instance or top level of the current design will be processed during the operand isolation optimization phase.

When an attribute is specified for an instance as well as the corresponding design of that instance, the attribute value of the instance has higher priority. When the attribute is not specified on either the design or the instance, the behavior is inherited from the parent instance. The default value for the top level of the current design is **true**. By default, operand isolation processing is enabled for the entire design.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example indicates that all instances of the design named *dsg* should be considered for processing during operand isolation:

```
prompt> set_operand_isolation_scope [get_designs dsg]
```

This example specifies that the *U3/U5* instance should be excluded from operand isolation processing:

```
prompt> set_operand_isolation_scope [get_cells U3/U5] false
```

SEE ALSO

```
do_operand_isolation(3)
```

set_operating_conditions

Defines the operating conditions for the current design.

SYNTAX

```
int set_operating_conditions  
[-analysis_type bc_wc | on_chip_variation]  
[-min min_condition]  
[-max max_condition]  
[-min_library min_lib]  
[-max_library max_lib]  
[-min_phys min_proc]  
[-max_phys max_proc]  
[-library lib]  
[-object_list objects]  
[condition]
```

Data Types

<i>min_condition</i>	list
<i>max_condition</i>	list
<i>objects</i>	list
<i>condition</i>	list

ARGUMENTS

- analysis_type bc_wc | on_chip_variation**
Specifies how to use the operating conditions. The **bc_wc** and **on_chip_variation** options are mutually exclusive; use only one per command. Specifying either **bc_wc** or **on_chip_variation** switches the design to min_max mode. The **bc_wc** value specifies that the min and max operating conditions are two extreme operating conditions. In the bc_wc analysis, setup violations are checked only for the maximum operating condition, and the hold violations are checked only for the minimum operating condition. The **on_chip_variation** value specifies that the minimum and maximum operating conditions represent, respectively, the lower and upper bounds of the maximum variation of operating conditions on the chip. All maximum path delays use the maximum operating condition, and all minimum path delays use the minimum operating condition.
- min min_condition**
Specifies the operating condition to use for minimum delay analysis. If you do not specify an operating condition for minimum delay analysis, the tool uses the maximum operating condition. The **-min** option must be used with the **-max** option.
- max max_condition**
Specifies the operating condition to use for maximum delay analysis.
- min_library min_lib**
Specifies the library containing definitions of the operating conditions for minimum delay analysis. This is either a library name or a collection. The

tool selects the first library in the collection containing the definitions.

-max_library *max_lib*

Specifies the library containing definitions of the operating conditions for maximum delay analysis. This is either a library name or a collection. The tool selects the first library in the collection containing the definitions.

-min_phys *min_proc*

Specifies the name of the process resource to search for the resistance and capacitance values for minimum delay analysis. This option must be used in with the **-max_phys** option.

-max_phys *max_proc*

Specifies the name of the process resource to search for the resistance and capacitance values for maximum delay analysis. This argument must be used with the **-min_phys** option.

-library *lib*

Specifies the library containing definitions of the operating conditions for both maximum and minimum delay analysis. This is either a library name or a collection. The tool selects the first library in the collection containing the definitions.

-object_list *objects*

Specifies the cells or ports on which to set operating conditions. If you do not use this option, operating conditions are set on the design. This option accepts both leaf cells and hierarchical blocks. This option is only available for Multi-Voltage features and require the appropriate license.

condition

Specifies conditions that define environmental characteristics to use during maximum and minimum delay analysis.

DESCRIPTION

This command defines the operating conditions (or environmental characteristics) under which to time or optimize the current design. The operating conditions specified must be defined in *lib* or in one of the libraries in the *link_library*. A *local_link_library* set on the current design is added to the beginning of the *link_library*, before the *link_library* is searched. The order for a library search is as follows:

1. *lib*
2. **local_link_library**
3. **link_library**

If you do not specify any operating conditions for a design, the **compile** command searches in the first library of the link library for default operating conditions. If the library does not have default operating conditions, no operating conditions are used.

Operating conditions set by using the **-object_list** option override the operating

conditions set on design or higher levels of hierarchy.

To see the operating conditions that are defined for the current design, and to see which libraries the current design is linked to, use the **report_design** command. To see the operating conditions defined in the specified library, use the **report_lib** command.

To remove operating conditions from the current design, use **set_operating_conditions** without specifying any operating conditions, or use the **reset_design** command.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example shows an operating condition definition, as it appears in the source text of a library:

```
operating_conditions("BCCOM") {    process : 0.6 ;
    temperature : 20 ;
    voltage : 5.25 ;
    tree_type : "best_case_tree" ;
}
```

The name of this set of operating conditions is *BCCOM*. The parameters are defined as follows:

process

A floating-point number that represents the characteristics of a semiconductor manufacturing process.

temperature

A floating-point number that represents the temperature of the defined environment.

voltage

A floating-point number that defines the upper boundary of the voltage range in which the defined environment operates. The lower boundary is always 0.0.

tree_type

The interconnect model for the environment. The **compile** command uses the interconnect model to select a formula for calculating interconnect delays. Three models are available:

- *best_case_tree*, which assumes the net delay to be 0
- *worst_case_tree*, which uses the lumped RC model
- *balanced_tree*, in which all loads share the wire resistance evenly.

When process factor, operating temperature, and operating voltage deviate from their

nominal values, **compile** uses a linear model to compensate for the effect of deviations on such values as cell delays, input loads, and output drives. The nominal values used are defined in the same library that contains the definitions of the set of operating conditions.

The following example sets the operating conditions to *WCIND* if the *link_library* is *my_lib.db*, and the design does not have a *local_link_library* set. The library name for *my_lib.db* is *my_lib_core*.

```
prompt> set_operating_conditions WCIND
```

Using operating condition 'WCIND' found in library "my_lib_core".

In the following example, the *BCIND* values found in *other_lib.db* are used for minimum delay analysis, and *WCIND* values are used for maximum delay analysis. The library name for *other_lib.db* is *other_lib_core*.

```
prompt> set_operating_conditions -min BCIND -max WCIND \  
-library other_lib_core
```

Using operating condition 'BCIND' found in library 'other_lib_core'.

Using operating condition 'WCIND' found in library 'other_lib_core'.

The following example shows how to remove operating conditions defined for the current design:

```
prompt> set_operating_conditions
```

SEE ALSO

```
report_lib(2)  
reset_design(2)  
set_local_link_library(2)  
link_library(3)
```


set_opposite

Defines two input ports as logically opposite.

SYNTAX

```
int set_opposite  
port1  
port2
```

Data Types

```
port1      string  
port2      string
```

ARGUMENTS

```
port1 port1  
      Input port names in the current design that are logically opposite.
```

DESCRIPTION

Defines two input ports in the current design as logically opposite. Use the command to eliminate redundant inverters and improve the quality of optimization. Inconsistencies in logical relations among ports are detected. It is an error to specify such an inconsistency using **set_opposite**.

Use the **reset_design** command to remove this property from a port.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets two input ports named "A" and "B" to be logically opposite.

```
prompt> set_opposite A B
```

SEE ALSO

```
reset_design(2)  
set_equal(2)  
current_design(3)
```

set_optimize_dft_options

Defines options for physical design-for-test (DFT) optimization.

SYNTAX

```
status set_optimize_dft_options  
[-repartitioning_method none | single_directional | multi_directional | adaptive]  
[-single_dir_option horizontal | vertical]
```

ARGUMENTS

-repartitioning_method none | single_directional | multi_directional | adaptive
Specifies the type of the scan chain repartition algorithm used in placement-aware scan optimization.
If you specify **none**, the **optimize_dft** command does not do scan repartitioning.
If you specify **single_directional**, the **optimize_dft** command uses the traditional single-directional scan repartitioning algorithm. By default, the direction is **horizontal**. You can use the **-single_dir_option** option to specify the desired direction.
If you specify **multi_directional**, the **optimize_dft** command uses the multi-directional scan repartitioning algorithm. The multi-directional algorithm is designed to achieve better scan path total wire-length reduction efficiency, especially for designs with a large number of scan chains.
If you specify **adaptive**, the **optimize_dft** command chooses the scan repartitioning algorithm adaptively, according to the number of scan chains, for the best scan wire-length reduction efficiency. For designs with no more than 20 scan chains, the traditional single-directional scan repartitioning algorithm is used; otherwise, the multi-directional scan repartitioning algorithm is used. If the single-directional algorithm is chosen, the partition direction is determined by the setting of the **-single_dir_option** option.
The default value of this option is **adaptive**.

-single_dir_option horizontal | vertical
Specifies the desired direction for the single-directional scan repartitioning algorithm. The option takes effect only when the **-repartitioning_method** is set to **single_directional** or set to **adaptive** on a design with 20 or fewer scan chains.
The default value of this option is **horizontal**.

DESCRIPTION

This command defines the options that control the **optimize_dft** command.

SEE ALSO

`optimize_dft(2)`
`report_optimize_dft_options(2)`

set_optimize_pre_cts_power_options

Specifies options for the **optimize_pre_cts_power** command.

SYNTAX

```
status set_optimize_pre_cts_power_options
[-default]
[-honor_dont_touch]
[-honor_size_only]
[-psyn_constraint_file file_name]
[-cts_constraint_file file_name]
[-split_clock_gates true | false]
```

Data Types

file_name string

ARGUMENTS

-default

Specifies the default options for **optimize_pre_cts_power**. All the options previously set will be reset to the default values.

-honor_dont_touch

By default, the **optimize_pre_cts_power** command can restructure integrated clock-gating (ICG) cells that have the **dont_touch** attribute. When you set this option, such ICGs will not be merged or removed. This option is effective only when you set **set_power_options -clock_gating true**.

-honor_size_only

By default, the **optimize_pre_cts_power** command can restructure integrated clock-gating (ICG) cells that have the **size_only** attribute. When you set this option, such ICGs will not be merged or removed. This option is effective only when you set **set_power_options -clock_gating true**.

-psyn_constraint_file *file_name*

Specifies the name of the file that contains the design constraints for physical synthesis. The specified file should be a valid tcl script that is to be sourced during physical synthesis or placement stage. If this constraint file is found, it will be sourced during physical synthesis stage by **optimize_pre_cts_power**.

-cts_constraint_file *file_name*

Specifies the name of the file that contains the design constraints for clock tree synthesis (CTS). The specified file should be a valid tcl script that is to be sourced during the CTS stage. If this constraint file is found, it will be sourced during the CTS stage by **optimize_pre_cts_power**.

-split_clock_gates true | false

Specifies whether to enable the **-split_clock_gates** option. When set to **true**, **split_clock_gates** is invoked after the initial (trial) clock tree synthesis. It optimizes the timing of the enable pins of clock gates and replicates the

clock gates that affect the worst negative slack (WNS) if they are driving clock buffer trees. Refer to **set_split_clock_gates_options** for more details. This option is set to **false** by default.

DESCRIPTION

The command sets the various strategy options for the **optimize_pre_cts_power** command. If any constraint file names of physical synthesis, clock tree synthesis (CTS), or both are specified, the files will be read in by **optimize_pre_cts_power**. Note that the constraints specified in any of the tcl files will be persistent in the design and thus can affect future operations on the design even after **optimize_pre_cts_power**. You should apply CTS constraints before running any CTS commands.

The **report_optimize_pre_cts_power_options** command reports the option values.

If you set **set_power_options -clock_gating to false** (the default) to disable clock-gating optimization, all the sub-options for clock-gating optimization will be ignored.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows how to set the options by using **set_optimize_pre_cts_power_options**.

```
prompt> set_power_options -clock_gating true
prompt> set_optimize_pre_cts_power_options -honor_dont_touch \
        -psyn_constraint_file psyn_sdc.tcl
prompt> optimize_pre_cts_power
```

SEE ALSO

```
set_power_options(2)
optimize_pre_cts_power(2)
report_optimize_pre_cts_power_options(2)
set_split_clock_gates_options(2)
```

set_output_delay

Sets output delay on pins or output ports relative to a clock signal.

SYNTAX

```
int set_output_delay  
delay_value  
[-clock clock_name [-clock_fall] [-level_sensitive]]  
[-network_latency_included]  
[-source_latency_included]  
[-rise]  
[-fall]  
[-max]  
[-min]  
[-add_delay]  
[-group_path group_name]  
port_pin_list
```

Data Types

<i>delay_value</i>	float
<i>clock_name</i>	string or collection
<i>group_name</i>	string
<i>port_pin_list</i>	list

ARGUMENTS

- delay_value*
Specifies the path delay. The *delay_value* must be in units consistent with the technology library used during optimization. The *delay_value* represents the amount of time that the signal is required before a clock edge. For maximum output delay, this usually represents a combinational path delay to a register plus the library setup time of that register. For minimum output delay, this value is usually the shortest path delay to a register minus the library hold time.
- clock** *clock_name*
Specifies the clock to which the specified delay is related. If **-clock_fall** is used, **-clock** *clock_name* must be specified. If **-clock** is not specified, the delay is relative to time zero for combinational designs. For sequential designs, the delay is considered relative to a new clock with period determined by considering the sequential cells in the transitive fanout of each port.
The clock can be either a string or collection of one object.
- clock_fall**
Specifies that the delay is relative to the falling edge of the clock. If **-clock** is specified, the default is the rising edge.
- level_sensitive**
Specifies that the destination of the delay is a level-sensitive latch. This allows the tool to derive setup and hold relationship for paths to this port

as if it were a level-sensitive latch. If **-level_sensitive** is not used, the output delay is treated as if it were a path to a flip-flop.

-network_latency_included

Specifies that the clock network latency should not be added to the output delay value. If this option is not specified, the clock network latency of the related clock is added to the output delay value. It has no effect if the clock is propagated or the output delay is not specified with respect to any clock.

-source_latency_included

Specifies that the clock source latency should not be added to the output delay value. If this option is not specified, the clock source latency of the related clock is added to the output delay value. It has no effect if the output delay is not specified with respect to any clock.

-rise

Specifies that *delay_value* refers to a rising transition on specified ports of the current design. If neither **-rise** nor **-fall** is specified, then rising and falling delays are assumed to be equal.

-fall

Specifies that *delay_value* refers to a falling transition on specified ports of the current design. If neither **-rise** nor **-fall** is specified, then rising and falling delays are assumed to be equal.

-max

Specifies that *delay_value* refers to the longest path. If neither **-max** nor **-min** is specified, maximum and minimum output delays are assumed to be equal.

-min

Specifies that *delay_value* refers to the shortest path. If neither **-max** nor **-min** is specified, maximum and minimum output delays are assumed to be equal.

-add_delay

Specifies whether to add delay information to the existing output delay, or to overwrite. The **-add_delay** option enables you to capture information about multiple paths leading from an output port that are relative to different clocks or clock edges.

For example, the following command removes all other maximum rise output delay from **OUT1**, since **-add_delay** is not specified. Other output delay with a different clock or with **clock_fall** is removed.

```
prompt> set_output_delay 5.0 -max -rise -clock phil {OUT1}
```

In the following example, **-add_delay** is specified:

```
prompt> set_output_delay 5.0 -max -rise -clock phil \  
-add_delay {Z}
```

If there is an output maximum rise delay for **Z** relative to the clock **phil** rising edge, the larger value is used. The smaller value does not result in critical timing for maximum delay. For minimum delay, the smaller value is used. If there is maximum rise output delay relative to a different clock or different edge of the same clock, it remains with the new delay.

-group_path group_name

Specifies that paths ending at the specified ports or pins are added into the named group. If the group does not already exist, it is created. This is

equivalent to specifying the following command in addition to
set_output_delay:
prompt> **group_path -name group_name -to port_pin_list**
If **-group_path** is not specified, the existing path grouping is not changed.

port_pin_list

A list of output port or internal pin names in the current design to which *delay_value* is assigned. If more than one object is specified, the objects are enclosed in double quotation marks (") or in braces ({}). If output delay is specified on a pin, the cell of the pin is set to size only to leave room for compile applying sizing on it.

DESCRIPTION

This command sets output path delay values for the current design. Used with the **set_load** and **set_driving_cell** commands, the input and output delays characterize the operating environment of the current design.

The **set_output_delay** command sets output path delays on output ports relative to a clock edge. Output ports are assumed to have no output delay unless specified. For inout (bidirectional) ports, you can specify the path delays for both input and output modes.

To describe a path delay to a level-sensitive latch, use the **-level_sensitive** option. If the latch is positive-enabled, set the output delay relative to the rising clock edge. If the latch is negative-enabled, set the output delay relative to the falling clock edge. If time is being borrowed at that latch, subtract that time borrowed from the path delay to the latch when determining output delay.

The **characterize** command automatically sets input and output delay, drive, and load values based on the environment of a cell instance.

The tool adds input delay to path delay for paths starting at primary inputs and output delay for paths ending at primary outputs.

The **-group_path** option modifies the path grouping. Path grouping affects the maximum delay cost function. The worst violator within each group adds to the cost. For optimization, grouping some paths separately may improve their delay cost, but it may also increase design area and compile time.

Use **report_port** to list output delays associated with ports. To list output delays of internal pins, use **report_design**. Use **report_path_group** to list the path groups which are defined.

Use **remove_output_delay** or **reset_design** to remove output delay values. To modify paths grouped with the **-group_path** option, use the **group_path** command to place the paths in another group or the default group.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example sets an output delay of 1.7 relative to the rising edge of CLK1 for all output ports in the design.

```
prompt> set_output_delay 1.7 -clock [get_clocks CLK1] \  
[all_outputs]
```

The following example sets the input and output delays for the INOUT1 bidirectional port. The input signal arrives at INOUT1 2.5 units after the falling edge of CLK1. The output signal is required at INOUT1 at 1.4 units before the rising edge of CLK2.

```
prompt> set_input_delay 2.5 -clock CLK1 -clock_fall {INOUT1}
```

```
prompt> set_output_delay 1.4 -clock CLK2 {INOUT1}
```

In the following example there are three paths from the OUT1 output port. One path is relative to the rising edge of CLK1. Another path is relative to the falling edge of CLK1. The third path is relative to the falling edge of CLK2. The **-add_delay** option is used to indicate that new output delay information will not cause old information to be removed.

```
prompt> set_output_delay 2.2 -max -clock CLK1 \  
-add_delay {OUT1}
```

```
prompt> set_output_delay 1.7 -max -clock CLK1 \  
-clock_fall -add_delay {OUT1}
```

```
prompt> set_output_delay 4.3 -max -clock CLK2 \  
-clock_fall -add_delay {OUT1}
```

The following example specifies two maximum delays and two minimum delays for the Z port using **-add_delay**. Since the information is relative to the same clock and clock edge, only the largest of the maximum values (5.0) and the smallest of the minimum values (1.1) are maintained. If **-add_delay** is not used, the new information overwrites the old information.

```
prompt> set_output_delay 3.4 -max -clock CLK1 \  
-add_delay {Z}
```

```
prompt> set_output_delay 5.0 -max -clock CLK1 \  
-add_delay {Z}
```

```
prompt> set_output_delay 1.1 -min -clock CLK1 \  
-add_delay {Z}
```

```
prompt> set_output_delay 1.3 -min -clock CLK1 \  
-add_delay {Z}
```

The following example uses the **-group_path** option to add ports into a named group. Without this option, paths to these ports are included in the CLK group.

```
prompt> set_output_delay 4.5 -max -clock CLK \  
-group_path busA {busA[*]}
```


SEE ALSO

`all_outputs(2)`
`characterize(2)`
`create_clock(2)`
`current_design(2)`
`group_path(2)`
`remove_output_delay(2)`
`report_design(2)`
`report_path_group(2)`
`report_port(2)`
`reset_design(2)`
`set_driving_cell(2)`
`set_load(2)`
`set_output_delay(2)`

set_pad_physical_constraints

Sets physical constraints per pad instance.

SYNTAX

```
status set_pad_physical_constraints  
-pad_name string pad_name objects  
    | [-side side_number]  
[-order order_number]  
[-offset offset_distance]  
[-orientation reflect | optimizeReflect]  
[-min_left_iospace min_space_left]  
[-min_right_iospace min_space_right]
```

Data Types

<i>objects</i>	collection
<i>side_number</i>	integer
<i>order_number</i>	integer
<i>offset_distance</i>	float
<i>min_space_left</i>	float
<i>min_space_right</i>	float

ARGUMENTS

-pad_name *string*
Specifies the name of the pad cell to which the specified constraints apply. You can specify a single pad cell, which does not have to exist in the design. The constraint is stored in the database even though it might not get applied. In such cases, the constraint is automatically applied when a new pad with this name is added to the design. This option and the *objects* argument are mutually exclusive. You must specify the pad cells by using one of these arguments.

objects
Specifies the pad cells to which the specified constraints apply. You can specify one or more pad cells that currently exist in the design. However, only one object is allowed if you use the **-order** option. This argument and the **-pad_name** option are mutually exclusive. You must specify the pad cells by using one of these arguments.

-side *side_number*
Specifies the die edge on which the pad must reside. Pads are supported only with rectangular floorplans, so there are only four valid sides. The valid values for this option are

- * 0 (default - no side constraints)
- * 1 (left side constraint)
- * 2 (top side constraint)
- * 3 (right side constraint)
- * 4 (bottom side constraint)

The default is 0, which means that the pad does not have a side constraint.

`-order order_number`
 Specifies the placement order number for the pad. The placement order is a clockwise ordering constraint. This number must be a positive integer. The default is 0, which means that the pad does not have an ordering constraint.

`-offset offset_distance`
 Specifies the distance in microns that the pad must be offset from from the die edge on which it is placed. The specified value must be a positive floating point number.

`-orientation reflect | optimizeReflect`
 Specifies the lateral reflection mode. Valid values are:
 * `reflect`
 Reflects the pad instance about the axis through its center that is perpendicular to the edge on which the pad cell is sitting.
 * `optimizeReflect`
 Allows the tool to reflect only if area is optimized as a result.
 If you do not specify this option, no lateral reflection is done.

`-min_left_iospace min_space_left`
 Specifies the minimum spacing in microns to the next I/O pad on the left side. The default value is 0.

`-min_right_iospace min_space_right`
 Specifies the minimum spacing in microns to the next I/O pad on the right side. The default value is 0.

DESCRIPTION

Specifies physical design constraints on pads, such as location and offset between adjacent pads. The specified constraints are saved when the design is saved.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows how to specify physical constraints on a pad.

```
prompt> set_pad_physical_constraints -pad_name "uDFT3/uINP0" \
-side 1 -order 1 -offset 248.34
```

SEE ALSO

```
remove_io_constraints(2)
set_pin_physical_constraints(2)
set_chiplevel_pad_physical_constraints(2)
write_io_constraints(2)
report_io_constraints(2)
```

`read_io_constraints(2)`

set_parameter

Sets parameters for placement and routing operations.

SYNTAX

```
int set_parameter  
-name name  
-value value  
[-module {route | groute | droute | trackAssign | ek | preroute | sr}]  
[-type {{int | integer} | {real | float} | string}]
```

Data Types

<i>name</i>	string
<i>value</i>	integer

ARGUMENTS

-name *name*
Specifies the name of the parameter to be set.

-value *value*
Specifies the value of the parameter to be set.

-module {route | groute | droute | trackAssign | ek | preroute | sr}
{route | groute | droute | trackAssign | ek | preroute | sr}
Specifies the module for which you are setting a parameter. If this option is not specified, then the tool searches the parameters in all of the modules supported by the **set_parameter** command. The tool sets the value if exactly one match is found. If multiple modules have the same parameter name, use the **-module** option to uniquely identify the parameter.

-type {{int | integer} | {real | float} | string}
Specifies the type of the parameter to be set. The three parameter types supported are integer, real, and string. If this option is not specified, the tool searches for the parameter and sets the value if exactly one match is found.

DESCRIPTION

The **set_parameter** command sets parameters for placement and routing operations. Use the **report_parameter** command to view a list of parameters, their values, and their definitions.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets the *timingdriven* parameter to a value of 1:

```
prompt> set_parameter -name timingdriven -value 1
```

The following example sets the *timingdriven* parameter to an integer value of 1 in the **droute** module:

```
prompt> set_parameter -name timingdriven -value 1 -module droute \  
        -type integer
```

SEE ALSO

```
report_parameter(2)  
insert_diode(2)  
insert_metal_filler(2)  
insert_ng_filler(2)  
insert_pad_filler(2)  
insert_stdcell_filler(2)  
insert_well_filler(2)
```

set_partition_data

Manipulates data contained in the partition editor created by **read_partition** command.

SYNTAX

```
status set_partition_data
[-auto_partition instance_count | area]
[-reset overrides | partition | keepouts | all]
[-physical]
[-logical]
[-area sub_block_area]
[-internal_keepout keepout]
[-external_keepout keepout]
[-utilization block_utilization]
[-aspect_ratio float]
[-force]
[module_name_list]
```

Data Types

<i>sub_block_area</i>	float, area in square microns
<i>keepout</i>	positive float, keepout size in micron
<i>block_utilization</i>	float, > 0, <= 1, block utilization percentage
<i>module_name_list</i>	string list

ARGUMENTS

-auto_partition instance_count | area

This option specifies to do an automatic partitioning. This partitioning can be based on *instance_count*, which balances partitions based on equalizing the number of instances in each partition. The other choice is *area*, which balances partitions based on area (size).

-reset overrides | partition | keepouts | all

Resets some or all of the computed parameters in the partition. Possible choices are *overrides*, which resets any areas or utilization settings that were overridden; *partition*, which resets any partitioning choices that were made manually or through autopartitioning; *keepouts*, which resets any keepouts that were set; or *all*, which does all of these things.

-physical

Indicates that the blocks in the *module_name_list* will be labelled as physical, in other words, labelling them as partitions. Note that the top block in a design is by definition physical and logical. This option is mutually exclusive with the **-logical** option.

-logical

Indicates that the blocks in the *module_name_list* will be labelled as logical, in other words labelling them as internal (logical) hierarchy. Note that the top block in a design is by definition physical and logical. This option is mutually exclusive with the **-physical** option.

`-area sub_block_area`

Allows you to set or to override the area for a given module within the hierarchy. This option can be used in a top-down design flow when one or more (presumably large) blocks are missing, but their estimated area is known. This can result in more properly considering the missing blocks expected contribution during area-based autopartitioning and during top-down, black-box shaping and floorplanning. There is no restriction on whether or not the overridden block is a physical block.

In some circumstances a module might already have a well-defined area but the designers are aware that a change is coming that will result in a different area. A designer can get a head start on refloorplanning the design by overriding the area on the current version of the module. Note that in this case, using this option will fail with an error message unless it is used with the **-force** option.

`-internal_keepout keepout`

Sets an internal keepout on the given modules. This keepout will be added to the area of the module. If a module is partitioned into a physical block, it will become larger by the internal keepout. The default internal keepout is 0.

`-external_keepout keepout`

Sets an external keepout on the given modules. This keepout will not cause the module itself to become larger. However, the parents of the module will become larger due to the keepout. The default external keepout is 0.

`-utilization block_utilization`

Allows you to specify the utilization on modules. The set utilization is used along with the leaf cell areas to compute the required area for the physical blocks. Note that in some cases you might need to use the **-force** option to have this work.

`-aspect_ratio float`

Allows you to specify a target aspect ratio (height/width) for the chip to be created. The default aspect ratio is 1.

`-force`

Use this option with the **-area** option to allow the area of a block with a well-defined area (one that has a fully-complete subhierarchy with bindings to all instantiated leaf cells) to be overridden. An error results if you try and set the area of a block that is not missing without using this option. This option is also needed if you use the **-utilization** option on multiply instantiated modules.

`module_name_list`

This is a list of modules contained in the design that are to be operated upon by the other settings of the command.

DESCRIPTION

This command allows you to manipulate a partition editor created by the **read_partition** command. For example, use **-physical** to label a module as a physical partition, ie block. Use **-logical** to remove the block as a physical partition.

This command also allows you to run automatic partitioning. The `instance_count` method creates partitions that typically represent 5-10 percent of the design, as measured by instance count. This is a very quick partitioning that works well for designs with few macro cells and with well-structured RTL. The `area` method creates partitions that typically represent between 5 and 10 percent of the design, as measured by area. This is a very quick partitioning that works better for designs with many hard macros and with well-structured RTL.

This command can be used by making several independent calls, each one making a slight modification.

EXAMPLES

This example shows reading in the Verilog and autopartitioning.

```
prompt> read_partition -input_files {myDesign.v}
Partitioner Name: pfe0
Loading verilog file ../ICC_DATA/Netlist/myDesign.v
MYDESIGN: 73958 instances of 134 modules.
1
prompt> set_partition_data -auto_partition instance_count
After partitioning, list of sub-blocks is BLENDER_0 BLENDER_1 BLENDER_2 BLENDER_3
1
```

This example sets the utilization on a module.

```
prompt> set_partition_data -utilization 0.85 BLENDER_0
1
```

This example sets some modules to be physical partitions.

```
prompt> set_partition_data -
physical "BLENDER_0 BLENDER_1 BLENDER_2 BLENDER_3 BLENDER_4"
1
```

SEE ALSO

```
create_partition(2)
read_partition(2)
set_partition_data(2)
list_partition_data(2)
```

set_pg_pin_model

On-The-Fly setting of library cell pg pins.

SYNTAX

```
Boolean set_pg_pin_model  
cell_name  
[-pg_pin_name {pin_name}]  
[-pg_voltage_name {voltage_name}]  
[-pg_pin_type {pin_type}]  
[-pg_pin_direction {pin_direction}]  
[-pg_physical_connection {physical connection}]  
[-pg_related_bias_pin {related bias pin}]  
[-reset]
```

Data Types

<i>cell_name</i>	string
<i>pin_name</i>	string
<i>voltage_name</i>	string
<i>pin_type</i>	string [primary_power primary_ground
<i>pin_direction</i>	string [input output inout internal]

ARGUMENTS

cell_name
Specify the library cell name to define pg pins.

-pg_pin_name {*pin_name*}
Specify the library cell pg pin names to define.

-pg_voltage_name {*voltage_name*}
Specify the library cell pg pin corresponding voltage name, voltage will be defined in library voltage map.

-pg_pin_type {*pin_type*}
Specify the library cell pg pin type to define.

-pg_pin_direction {*pin_direction*}
Specify the library cell pg pin direction to define.

-pg_physical_connection {*physical connection*}
Specify the library cell pg pin physical connection to define.

-pg_related_bias_pin {*related bias pin*}
Specify the library cell pg pin related bias pin to define.

-reset
reset the library cell pg pins, pin related pg pin will also be reset when the pg pin is reset.

DESCRIPTION

The **set_pg_pin_model** command define the pg pins for the cell. If the pg pin already exists, all the new specifications will override the old ones. If not, the new pg pin will be created for the cell.

If -reset is issued, the pg pins of the cell will be reset. All the associated related pg pins for the cell signal pins will be reset too.

if any option is specified, they must be paired with the pin definition. e.g. pin name, voltage, type, direction, connection and related_bias_pin.

If -reset is combined with other options, reset happens first, other options will be set on the cell after reset.

EXAMPLES

```
prompt> set_pg_pin_model LVLH* -pg_pin_name {VDD VSS} -pg_voltage_name {VDDL VSS} -  
pg_pin_type {primary_power primary_ground}
```

SEE ALSO

```
set_voltage_model(2)  
report_lib(2)  
check_library(2)
```

set_physical_signoff_options

Sets environment settings for IC Validator or Hercules in preparation for running the **signoff_drc** or **signoff_metal_fill** command.

SYNTAX

```
status set_physical_signoff_options
[-default]
[-exec_cmd icv | hercules]
[-drc_runset filename]
[-fill_runset filename]
[-mapfile filename]
[-dp_hosts {list_of_dp_hosts}]
[-num_cpus integer]
```

Data Types

<i>filename</i>	string
<i>list_of_dp_hosts</i>	list

ARGUMENTS

- default
Restores all options to their default values.
- exec_cmd icv | hercules
Specifies the name of executable. To use IC Validator, specify **icv**. To use Hercules, specify **hercules**. The default is none. The tool uses the standard search path to search for the specified executable.
- drc_runset *filename*
Specifies the foundry runset to use for design rule checking (DRC). The default is none.
- fill_runset *filename*
Specifies the foundry runset to use for metal filling. The default is none.
- mapfile *filename*
Specifies the layer mapping file. This file maps Milkyway layers to runset (GDSII) stream layers. Both Milkyway-formatted or application tool-formatted mapping files (such as a Hercules-formatted mapping file) are acceptable. The default is none.
The syntax of the mapping information in a Hercules mapping file is *Milkyway_layer[/Milkyway_data_type] > GDSII_layer[/GDSII_data_type]*
The syntax of the mapping information in a Milkyway mapping file is *Milkyway_layer[:Milkyway_data_type] GDSII_layer[:GDSII_data_type]*
- dp_hosts {*list_of_dp_hosts*}
Specifies the machines to be used for distributed processing. The default is none. If you do not specify this option, one CPU from the local machine is used for processing.

`-num_cpus integer`

Specifies the number of CPUs for distributed processing. The number of CPUs must be greater than or equal to 1 and should be less than or equal to the number of hosts specified in the **-dp_hosts** option.
The default value is 1.

DESCRIPTION

This command sets environment settings for IC Validator or Hercules. You must run this command before running the **signoff_drc** or **signoff_metal_fill** command.

The **report_physical_signoff_options** command reports option values set by the **set_physical_signoff_options** command.

Warning

IC Validator and Hercules run hierarchically and report errors in the top-level design by expanding the lower-level errors without merging them for the purpose of the IC Compiler error browser.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example distributes IC Validator jobs to four CPUs by using one CPU on each of the machines specified by the **-dp_hosts** option. It also uses a foundry runset called test.rs for DRC checking.

```
prompt> set_physical_signoff_options -exec_cmd icv
-dp_hosts {machine_1 machine_2 machine_3 machine_4} \
-num_cpus 4 -drc_runset test.rs
```

The following example distributes IC Validator jobs to four CPUs by using a maximum of two CPUs on machine1 and machine2 and a maximum of one CPU on machine3 and machine4.

```
prompt> set_physical_signoff_options -exec_cmd icv
-dp_hosts {machine_1 machine_1 machine_2 machine_2 machine_3 machine_4} \
-num_cpus 4 -drc_runset test.rs
```

The following example shows a Milkyway-formatted layer mapping file that maps Milkyway data type 0 on Milkyway layer 41 to GDSII data type 0 on GDSII layer 61. The "A" indicates that this mapping applies to both text and data.

```
A 41 61
```

The following example shows a Hercules-formatted layer mapping file that maps all data types on Milkyway layer 31 to GDSII layer 61.

```
31 > 61
```

SEE ALSO

`report_physical_signoff_options(2)`
`signoff_drc(2)`
`signoff_metal_fill(2)`

set_physopt_cpulimit_options

Specifies the options for performing a CPU-limited optimization.

SYNTAX

```
status set_physopt_cpulimit_options
[-name limit_name]
[-mode max_delay | design_rule | max_area | tns | max_static_power]
[-absolute_time absolute_time_value]
[-absolute_cost absolute_cost_value]
[-delta_time delta_time_value]
[-delta_cost delta_cost_value]
[-path_group path_group_name]
[-remove]
```

Data Types

<i>limit_name</i>	string
<i>absolute_time_value</i>	int
<i>absolute_cost_value</i>	float
<i>delta_time_value</i>	int
<i>delta_cost_value</i>	float
<i>path_group_name</i>	string

ARGUMENTS

-name *limit_name*
Specifies the name of limit constraint. The string value *limit_name* will be used to refer to this limit constraint and this name can be used for removing an earlier set limit constraint. **-name** is a required option.

-mode max_delay | design_rule | max_area | tns | max_static_power
Specifies the optimization mode for which the runtime limit is to be applied, wherever appropriate . The argument values represent worst setup, design rule, area recovery, TNS, static power recovery optimization respectively.

-absolute_time *absolute_time_value*
Specifies the cputime limit constraint for optimization in physopt. *absolute_time_value* is in minutes and is an integer value. If runtime for the optimization mode for which **-absolute_time** is specified, exceeds *absolute_time_value*, physopt bails out optimization for that mode. This lets you specify different timelimits for different optimization modes. Multiple **-absolute_time** constraints can be set for the session. But only one mode can be constrained per each **set_physopt_cpulimit_options** command. CPU time during optimization can be displayed by adding **%cpu** keyword to **compile_log_format** variable. **-mode** is optional with **-absolute_time**. If **-mode** is not specified the *absolute_time_value* limit is applied as a sum-total runtime bound for all modes during optimization. **-absolute_time** is exclusive with **-absolute_cost**, **-delta_time**, **-remove**. Other options that **-absolute_time** can be used with is **-mode** along with the required option **-name**.

-absolute_cost *absolute_cost_value*

Specifies an absolute cost limit for an optimization mode. If the *absolute_cost_value* absolute cost criteria is met for that optimization mode, that optimization mode is bailed out, and optimization will continue for other modes. **-absolute_cost** takes a float value. **-absolute_cost** is exclusive with **-absolute_time**, **-delta_time**, **-remove**. Other options that **-absolute_cost** can be used with is **-mode** and **-path_group** along with the required option **-name**. For now, **-absolute_cost** bound will only be supported for *max_delay* since *absolute_cost_value* can be abstract for other modes. **-mode** with *max_delay* has to be specified with **-absolute_cost**. If *path_group_name* is also specified, the cost criteria is checked only for the path group in the design named as *path_group_name*. If no *path_group_name* is specified, the cost criteria is applied to the sum total of the worst negative slacks for all path groups in the design. Above summed worst cost can be displayed by adding **%max_delay** keyword to **compile_log_format** variable.

-delta_time *delta_time_value*

Specifies a time limit in minutes such that if there is less than the specified *delta_cost_value* cost improvement for that optimization mode, within every block-interval of *delta_time_value* specified timelimit, that optimization mode is terminated. This lets user specify a bailout limit based on the rate at which the cost is getting fixed. **-delta_time** and **-delta_cost** have to be specified together. **-delta_time** is exclusive with **-absolute_time**, **-absolute_cost**, **-remove**. Other options that **-delta_time** can be used with is **-mode** along with the required option **-name**. For now, *max_delay* mode option is only supported with **-delta_time**. **-mode** with *max_delay* has to be specified with **-delta_time**. For *max_delay* mode the *delta_cost_value* improvement is calculated as sum total improvement of worst slack over all path groups in the design. Progress of the above summed worst cost can be displayed by adding **%max_delay** keyword to **compile_log_format** variable.

-path_group *path_group_name*

Specifies a path group in the design. Takes a string value.

-remove

This can be used to remove an earlier set **set_physopt_cpulimit_options** limit. The name of the limit has to be specified to remove the limit.

DESCRIPTION

This command allows users to specify limits which are 1. runtime based or 2. cost based or 3. based on the rate at which the cost is getting fixed for various optimization modes and run physopt with these limits. Physopt optimization will bailout that optimization mode if any of the appropriate limit constraint is met. **set_physopt_cpulimit_options** lets you specify runtime bounds for optimization alone, independent of other steps (placement, loading etc.) during physopt. The limit options are not saved in the database and applies only for the current session. Limit options are applied afresh for each physopt Stage, and any prior bailouts and all time counters are reset at start of each physopt Stage.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets various **set_physopt_cpulimit_options** limits to physopt.

```
prompt> set_physopt_cpulimit_options -absolute_time 60 \  
-name cpulim1  
prompt> set_physopt_cpulimit_options -absolute_time 10 \  
-mode max_delay -name cpulim2  
prompt> set_physopt_cpulimit_options -absolute_time 10 \  
-mode design_rule -name cpulim3  
prompt> set_physopt_cpulimit_options -absolute_time 10 \  
-mode max_area -name cpulim4  
prompt> set_physopt_cpulimit_options -absolute_cost 1300 \  
-mode max_delay -name cpulim6  
prompt> set_physopt_cpulimit_options -absolute_cost 500 \  
-mode max_delay -path_group clock -name cpulim8  
prompt> set_physopt_cpulimit_options -delta_time 5 \  
-delta_cost 50 -mode max_delay -name cpulim9  
prompt> physopt -area_recovery
```

The above sequence sets the following limits to physopt, such that if any of the below limits is reached that appropriate optimization mode is terminated, and optimization continues with other modes. On a bailout, information message is printed with keyword **(CPULIMIT)**, and the message includes the cpulimit constraint name that caused the bailout. Note that all bailouts and time counters are reset at start of each physopt Stage . 1. Maximum runtime of 60 minutes for entire optimization (summed over all modes) (cpulim1)

2. Maximum runtime of 10 minutes for max_delay setup time optimization. This limit does not affect other modes. (cpulim2)

3. Maximum runtime of 10 minutes for DRC optimization. This limit does not affect other modes. (cpulim3)

4. Maximum runtime of 10 minutes for area recovery. This limit does not affect other modes. (cpulim4)

5. If sum total of worst slack over all path groups betters 1300, max_delay optimization is terminated. (cpulim6)

6. If worst slack of path group clock 500, max_delay optimization of path group clock is terminated. Optimization will continue for other path groups , other modes. (cpulim8)

7. If the sum total worst negative slack over all path groups in the design does not improve by 50 for every 5 minutes, max_delay WNS optimization will be terminated. (cpulim9)

SEE ALSO

compile_log_format(3)

set_pin_model

On-The-Fly setting of library cell pin related pg pins.

SYNTAX

```
Boolean set_pin_model  
cell_name  
[-pins {pin_name}]  
[-related_power_pin {pin_name}]  
[-related_ground_pin {pin_name}]  
[-related_bias_pin {pin_name}]  
[-power_down_function {function}]  
[-reset]
```

Data Types

<i>cell_name</i>	string
<i>pin_name</i>	string
<i>function</i>	string

ARGUMENTS

cell_name
Specify the library cell name to define pins' related pins and power_down_function.

-pins {*pin_name*}
Specify the library cell pin names to be defined.

-related_power_pin {*pin_name*}
Specify the library cell pin related power pin name.

-related_ground_pin {*pin_name*}
Specify the library cell pin related ground pin name.

-related_bias_pin {*pin_name*}
Specify the library cell pin related bias pin name.

-reset
Reset the library cell pins' related pins and power_down_function.

DESCRIPTION

The **set_pin_model** command define the library cell pins' related power, ground or bias pins. If the related pin already defined, the new specification will override the old one.

If -reset is issued, all the related pins of the pins will be reset. Power_down_function of the pin will also be reset.

If -reset is combined with other options, reset happens first, other options will be

set on the cell after reset.

EXAMPLES

```
prompt> set_pin_model *DRFF* -pins {RETN RETNOUT} -related_power_pin VDDG -  
related_ground_pin VSSG  
prompt> set_pin_model HEAD* -pins {SLEEPOUT} -power_down_function "!VDDG + VSS"
```

SEE ALSO

```
set_voltage_model(2)  
set_pg_pin_model(2)  
report_lib(2)  
check_library(2)
```

set_pin_name_synonym

Defines synonyms for pin names.

SYNTAX

```
Boolean set_pin_name_synonym  
[-full_name]  
[-force]  
pin_name_synonym  
pin_name
```

Data Types

<i>pin_name_synonym</i>	string
<i>pin_name</i>	string

ARGUMENTS

-full_name
Indicates that a pin name synonym is a full name synonym.

-force
Overwrites the existing pin name synonym if a conflict occurs.

pin_name_synonym
Specifies a pin name synonym string for *pin_name*. This argument is required.

pin_name
Specifies a pin name string for which the synonym is defined. This argument is required.

DESCRIPTION

This command defines pin name synonyms, which are supported in pin object queries.

The *pin_name_synonym* is an alternative pin name for a pin object. For example, if *U1/U2/data_in* is defined as the synonym for the *U1/U2/D* pin, pin query commands can find the pin object named *U1/U2/D* when the commands could not find the pin with the given name *U1/U2/data_in*.

Pin name synonyms are only applicable to pins on leaf-level sequential cells.

Two types of pin name synonyms are supported: full name synonyms and simple name synonyms. The **set_pin_name_synonym** command defines simple name synonyms unless the **-full_name** option is used. When defining a full name synonym, you specify the full hierarchical name for the pin object in both the *pin_name_synonym* and *pin_name* fields. Pin query commands use full name synonyms exactly as they are defined. Simple name synonym definitions use short names for pin objects. Pin query commands use a simple pin name synonym to construct a full pin name during a synonym-based pin object search. You can get full name and simple name synonyms for pin objects by querying **full_name** and **name** attributes for pin objects.

Wildcards and regular expressions are not supported in pin name synonym definitions. Simple name synonyms do not support the forward slash (/) since it is considered a hierarchical separator.

When using the pin name synonyms for pin queries, the full name synonym, when applicable, supersedes the simple name synonym.

EXAMPLES

The following examples use **set_pin_name_synonym** to define pin name synonyms, and show how **get_pins** command uses the synonyms:

```
prompt> set_pin_name_synonym SYN_CD CD
1

prompt> set_pin_name_synonym -full_name \
      this/is/a/synonym mid1/FJK2SP_at_mid/TI
1

prompt> set_pin_name_synonym -full_name \
      mid1/FJK2SP_at_mid/J mid2/bot2/LSR0P_at_bot/Q
1

prompt> set synonym_pin_query_verbose true
true

prompt> get_pins mid1/FJK2SP_at_mid/SYN_CD
Information: found 1 pin using simple name synonym definition {SYN_CD CD}.
{mid1/FJK2SP_at_mid/CD}

prompt> get_pins this/is/a/synonym
Information: found 1 pin using full name synonym definition \
      {this/is/a/synonym mid1/FJK2SP_at_mid/TI}.
{mid1/FJK2SP_at_mid/TI}
```

SEE ALSO

```
remove_pin_name_synonym(2)
report_pin_name_synonym(2)
```

set_pin_physical_constraints

Sets physical constraints per pin instance.

SYNTAX

```
status set_pin_physical_constraints
[-pin_name pin_name]
[-cell cell_name]
[-layers layers]
[-width pin_width]
[-depth pin_depth]
[-side side_number]
[-offset offset_distance]
[-order order_number]
[-off_edge {center | location | auto}]
[-location point]
[objects]
```

Data Types

<i>pin_name</i>	string
<i>cell_name</i>	string
<i>layers</i>	list
<i>pin_width</i>	float
<i>pin_depth</i>	float
<i>side_number</i>	integer
<i>offset_distance</i>	float
<i>order_number</i>	integer
<i>objects</i>	list

ARGUMENTS

`[-pin_name pin_name]`

The pin name identifies the pin for which the specified constraint is to be applied. Port or terminal names should be specified in **pin_name**. It is valid to enter pin names even if such a pin does not exist in the design. The constraint is stored in the database even though it might not get applied. In such cases, the constraint is automatically applied when a new pin with this name is added to the design.

This option and the *objects* argument are mutually exclusive. You must specify the pins by using one of these arguments.

`[-cell cell_name]`

Specifies the name of the Milkyway cell or plan group to which the pin belongs. If you specify this option, you must also apply the **-pin_name** option. This option and the *objects* argument are mutually exclusive. You must specify only one.

`[-layers layers]`

Specifies a set of metal layers allowed for the ports or terminals. The argument is a collection of consecutive metal layers. A collection that contains nonconsecutive metal layers is not supported.

set_pin_physical_constraints

2664

`[-width pin_width]`

Specifies the desired pin width in microns. The default is 0. If the width constraint specified is smaller than the minimum width given in the library, the minimum width is applied (unless otherwise noted).

`[-depth pin_depth]`

Specifies the desired pin depth in microns. The default is 0. If the depth constraint specified is smaller than the minimum depth given in the library, the the minimum depth is applied (unless otherwise noted).

`[-side side_number]`

Specifies the block side on which the pin must be placed so that it abuts to the specified die edge.

The side number is a positive integer that starts from 1. Given any shape (rectangular or rectilinear), the lower left-most edge is the starting edge (side number 1). The side number of the next edge, going clockwise, is 2, and so on. You can specify a value of 0 to indicate no side constraint.

`[-offset offset_distance]`

Specifies the distance in microns that the pins must be offset from the die edge on which they are placed. The offset is the distance to the edge of the pin. The specified value must be a positive floating point number.

`[-order order_number]`

Specifies the placement order for the relative positions of pins. The order number must be a positive integer. A value of 0 means the pin has no order constraint. If the edge is horizontal, the smaller the number you specify, the more to the left the pin should be placed. If the edge is vertical, the smaller the number you specify, the lower the pin should be placed. In addition, order constrained pins are allowed to be interleaved by other pins. The default is 0.

`-off_edge {center | location | auto}`

Specifies what type of off-edge pin it is. The default behavior of this command is to set an on-edge pin constraints.

The following is a description of the supported values. These values are mutually exclusive. Use only one value.

center

When **-off_edge center** is specified, the pin is placed in the center of the cell.

location

When **-off_edge location** is specified, you must provide the location for this pin by using the **-location point** option.

The pin is placed at the exact coordinates of provided location. In addition, the pin is marked as fixed once it is created.

auto

When **-off_edge auto** is specified, you must create a pin guide that is associated with this pin and it must be in the center of the block boundary. The pin is placed at the optimal position guided by the pin guide. If you do not provide a pin guide, the pins are placed in the center of the cell, which is the same as **-off_edge center**.

`[-location point]`

Specifies the coordinates of the off-edge pin. When this option is used, you must also apply the **-off_edge location** option. This option and the **-offset fp option** are mutually exclusive. Use only one.

`[objects]`

Specifies the pin on which the specified constraints apply. Multiple objects are supported. However, only one object is allowed if the **-order** option is specified.

You can specify either the **-pin_name** or **objects** option to set the pin constraints. However, these options are mutually exclusive. Use only one.

DESCRIPTION

Sets the physical constraints on pins such as layers and width. The constraints you specify are saved when the design is saved.

If you want the constraints to be honored during pin placement, use the **set_fp_pin_constraints -use_physical_constraints** option.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows how to specify physical constraints on a pin.

```
prompt> set_pin_physical_constraints -pin_name "mypin"  
-layers {M2 M3 M4 M5}  
-width 0.28 -depth 0.28 -side 1 -offset 5
```

SEE ALSO

```
read_io_constraints(2)  
report_io_constraints(2)  
remove_io_constraints(2)  
set_pad_physical_constraints(2)  
set_chiplevel_pad_physical_constraints(2)  
set_fp_pin_constraints(2)  
write_io_constraints(2)
```


set_place_opt_cts_strategy

Specifies clock-tree-related options for the **place_opt -cts** command.

SYNTAX

```
status set_place_opt_cts_strategy  
[-operating_condition min | max(default) | min_max]  
[-no_clock_route]  
[-inter_clock_balance]  
[-fix_hold]  
[-update_clock_latency]
```

ARGUMENTS

- operating_condition [min | max(default) | min_max]
Specify operating condition, default is max.
- no_clock_route
Don't perform routing of clock nets.
- inter_clock_balance
Execute inter-clock delay balancing.
- fix_hold
Executes hold time violation fixing during incremental optimization. This option is off by default.
- update_clock_latency
Updates the latencies on real and virtual clock objects after clock tree synthesis, clock tree optimization, inter-clock balance (if chosen) and clock tree detailed routing. This will effectively execute a set_clock_latency under-the-hood for the clock objects and use the insertion delay of the clock tree as the latency number. If set_latency_adjustment_options command is executed before **place_opt -cts**, then the directives will be obeyed. If no directives are given, then only the latencies of real clock objects will be update. The update mechanism uses the insertion delay of the clock tree as the latency number.

DESCRIPTION

The **set_place_opt_cts_strategy** command defines clock tree related options that will later be used by a subsequently issued **place_opt -cts** command in the same session. Note that these options are applicable only for **place_opt -cts** command. If **-cts** option is not used for running **place_opt**, then all of the values specified during this command are ignored during that invocation of place_opt.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command sets up operating condition as min for a subsequently issued **place_opt -cts** command. It also sets up **place_opt -cts** to execute inter-clock delay balancing.

```
prompt> set_place_opt_cts_strategy -operating_condition min -inter_clock_balance
```

SEE ALSO

place_opt(2)
clock_opt(2)

set_pnet_options

Sets up placement tools to avoid power strap violations.

SYNTAX

```
int set_pnet_options  
[-partial | -complete | -none]  
[-min_height height_threshold]  
[-min_width width_threshold]  
[-no_via_additive]  
[-density value]  
[-see_object object_type_names]  
layer_names
```

Data Types

<i>height_threshold</i>	float
<i>width_threshold</i>	float
<i>value</i>	float
<i>object_type_names</i>	list
<i>layer_names</i>	list

ARGUMENTS

-partial

Allows standard cells to be placed under pnets, but the pins in the standard cells are checked to prevent shorts with the pnets. The **-partial**, **-complete**, and **-none** options control how the tool places the cell under the pnets. These three options are mutually exclusive; you can specify only one. If you do not specify this option, -none is used as the default.

-complete

Prevents cells from ever overlapping with pnets. A pnet is treated as a placement keepout. The **-partial**, **-complete**, and **-none** options control how the tool places the cell under the pnets. These three options are mutually exclusive; you can specify only one. If you do not specify this option, -none is used as the default.

-none

Allows cells to be freely placed under pnets without checking for electrical shorts. The pnets are invisible during placement. The **-partial**, **-complete**, and **-none** options control how the tool places the cell under the pnets. These three options are mutually exclusive; you can specify only one. If you do not specify this option, -none is used as the default.

-min_height *height_threshold*

Specifies the wide metal threshold height value. If a pnet width or pnet height is less than the specified threshold, the tool automatically treats the pnet as partial blockage. You can use this option only if you also specify the **-partial** or **-complete** option. If a pnet width or pnet height is equal to or more than the specified threshold, the pnet is treated as a complete blockage regardless of whether you specify the **-partial** or **-complete** option.

This option is not applied to power rails. Power rails are always treated as partial blockages.

See the DESCRIPTION section for the definition of power rails. By default, when this option is used with the **-complete** option, the default threshold height is 0. When it is used with the **-partial** option, the default threshold is layer dependent. For metal 1 (M1) and metal 2 (M2), the default height threshold is (core_site_height)/2. For metal 3 (M3) and above, the threshold height is 10000 microns.

-min_width *width_threshold*

Specifies the wide metal threshold width value. If a pnet width or pnet height is less than the specified threshold, the tool automatically treats the pnet as partial blockage. You can use this option only if you also specify the **-partial** or **-complete** option. If a pnet width or pnet height is equal to or more than the specified threshold, the pnet is treated as a complete blockage regardless of whether you specify the **-partial** or **-complete** option. This option is not applied to power rails. Power rails are always treated as partial blockages.

-no_via_additive

Turns off via additive checking. Via additive is the difference in width and height between the lower level metal and the higher level metal inside the via. For example, given that VIA12 contains M1 and M2, the size difference between M2 and M1 is the via additive. This option assumes the via additive is 0. (The tool assumes the via can be dropped along the edge of the M1 pin and just checks the M2-M2 spacing rule from the pin to pnet strap). You can use this option only when the physical library includes the via geometry inside the port geometry definition.

-density *value*

Specifies the density of cells under the power nets on a certain layer. The value must be between 0 and 1. If the specified value is 1, the entire area under the pnet is available for cell placement. If the specified value is 0, the area under the pnet is not available for cell placement. If the specified value is 0.3, 30% of the area under the pnet can be covered with placed cells. This option cannot be used with the **-complete** option.

-see_object *object_type_names*

Specifies the route object types to be considered by the legality checker during the **physopt**, **legalize_placement**, and **check_legality** commands. These settings are general to all layers specified for the command. The valid values for this option are the following: **all_types**, **none**, **pg_ring**, **pg_strap**, **pg_pin_conn**, **clk_ring**, **clk_strap**, **bus**, **path**, **user_signal**, **rect**, and **contact_array**. By default, the PG ring, PG strap, PG pin connection, clock ring, clock strap, and path types are enabled; and the bus, user signal, rectangle, and contact array types are disabled.

layer_names

Specifies the pnet layer names, such as {Metal1 Metal2}. You can get the layer names from the report generated by using the **report_lib -physical** command or directly from the physical library file.

DESCRIPTION

This command verifies that there are no metal-to-metal violations between cells and power net segments by configuring the placement tools (**create_placement**, **legalize_placement**, **check_physical_constraints**, **physopt**, and **check_legality**).

This command sets attributes for each routing layer, which are stored in the design database. To determine the current values, use the **report_pnet_options** command. To remove the attributes so that you can re-apply a new set of values, use the **remove_pnet_options** command. Using **set_pnet_options** overwrites all previous settings for that layer. To retain some of the previous settings, define them again. You can specify for each layer to be either a complete blockage or a partial blockage. You can by using the **-none** option to specify whether routing layers are to be ignored during legality checking (). Irrespective of the options specified in this command, the pnets on the routing layers are still used for routing congestion purposes.

When you use the **-partial** option, if there is no standard cell pin on the the routing layer or the layer just below the routing layer, then this pnet layer is ignored because the standard cells can be freely placed under the pnets on the layer. For example, if you specify **-partial {M5}** but there is no M5 or M4 pin in any standard cell, then all the M5 pnets are ignored.

When you use the **-partial** or **-complete** option, power rails are always treated as partial blockages (although the pnet layer is set as a complete blockage, and the threshold is set smaller than the power rails width). A power rail is metal across the row top and bottom line, and the width is smaller than (core site height/2). To adjust the power rail height threshold, set the **physopt_power_rail_height_limit** variable. For example, if you set this variable to **1.2**, the tool recognizes the pnets that cross the site rows and have a height of less than 1.2 microns as power rails.

To apply pnet options for different layers, annotate the floorplan data to the design before using the **set_pnet_options** command. When you again annotate the floorplan data on the design, it resets the pnet options set on the design. After this, you must again use the **set_pnet_options** command to set the pnet options.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets the *metal1* to *metal4* layers as partial blockages. The default minimum width for *metal1* and *metal2* is the average cell width of 10.0. The default minimum height for *metal1* and *metal2* is the core site height (12.0) divided by 2 = 6.0. The default minimum width and height for *metal3* and *metal4* is 10000 microns.

```
prompt> read_ddc design
prompt> set_pnet_options -partial {metal1 metal2 metal3 metal4}
prompt> check_legality -verbose
*****
```

```

Report : pnet options
Design : xyz
Version: 2002.05
Date   : Fri Mar 1 08:01:46 2002
*****

```

Layer	Blockage	Min_width	Min_height	Via_additive	Density
metal1	partial	10.0	6.0	via additive	---
metal2	partial	10.0	6.0	via additive	---
metal3	partial	10000.00	10000.00	via additive	---
metal4	partial	10000.00	10000.00	via additive	---

The following example sets the *metal2* layers as complete blockages. The *min_height* and *min_width* threshold are set at 1.2 and 1.5 micron, respectively. When the pnet is smaller the 1.2 in height or 1.5 in width, it is treated as a partial blockage. Otherwise, it is treated as a complete blockage.

```

prompt> set_pnet_options -partial -min_height 1.2 \
-min_width 1.5 {metal1 metal2}
prompt> report_pnet_options

```

```

*****
Report : pnet options
Design : xyz
Version: 2002.05
Date   : Fri Mar 1 08:01:46 2002
*****

```

Layer	Blockage	Min_width	Min_height	Via_additive	Density
metal1	complete	1.50	1.20	via additive	---
metal2	complete	1.50	1.20	via additive	---
metal3	partial	10000.00	10000.00	via additive	---
metal4	partial	10000.00	10000.00	via additive	---

The following example sets the *metal1* and *metal2* layers as partial blockages (default). The *min_height* and *min_width* threshold are set at 1.2 and 1.5 micron, respectively. When the pnet is smaller the 1.2 in height or 1.5 in width, it is treated as a partial blockage. Otherwise, it is treated as a complete blockage.

```

prompt> set_pnet_options -partial -min_height 1.2 \
-min_width 1.5 {metal1 metal2}

```

The following example shows the use of the **-density** option in the command.

```

prompt> set_pnet_options -partial -density 0.3 \
-min_height 1.2 -min_width 1.5 {metal1 metal2}

```

```

*****
Report : pnet options
Design : xyz

```

```

Version: 2002.05
Date   : Fri Mar  1 08:01:46 2002
*****

```

Layer	Blockage	Min_width	Min_height	Via_additive	Density
metal1	complete	1.50	1.20	via additive	0.3
metal2	complete	1.50	1.20	via additive	0.3

The following example shows the use of the **-see_object** option.

```

prompt> set physopt_enable_astro_legality_checker true
Information: Defining new variable 'physopt_enable_astro_legality_checker'. (CMD-041)

```

```

prompt> set_pnet_options -partial {MET2}

```

```

prompt> report_pnet_options

```

```

*****
Report : pnet options
Design : A7S_top
Version: V-2004.06-PCXG-ALPHA8
Date   : Mon Mar 29 15:24:11 2004
*****

```

Layer	Blockage	Min_width	Min_height	Via_additive	Density
MET1	none	---	---	via additive	---
MET2	partial	---	---	via additive	---
MET3	none	---	---	via additive	---
MET4	none	---	---	via additive	---
MET5	none	---	---	via additive	---

```

Route types considered:
    pg ring, pg strap, pg pin connection, clock ring
    clock strap, path

```

```

prompt> set_pnet_options -partial {MET2} -see_object {all_types}

```

```

prompt> report_pnet_options

```

```

*****
Report : pnet options
Design : A7S_top
Version: V-2004.06-PCXG-ALPHA8
Date   : Mon Mar 29 15:24:36 2004
*****

```

Layer	Blockage	Min_width	Min_height	Via_additive	Density
MET1	none	---	---	via additive	---
MET2	partial	---	---	via additive	---
MET3	none	---	---	via additive	---
MET4	none	---	---	via additive	---
MET5	none	---	---	via additive	---

Route types considered:

pg ring, pg strap, pg pin connection, clock ring
clock strap, bus, path, user signal, rectangle, contact array

SEE ALSO

check_legality(2)
check_physical_constraints(2)
create_placement(2)
legalize_placement(2)
remove_pnet_options(2)
report_pnet_options(2)

set_port_fanout_number

Sets the number of external fanout points driven by specified ports in the current design.

SYNTAX

```
status set_port_fanout_number  
fanout_number  
port_list
```

Data Types

<i>fanout_number</i>	float
<i>port_list</i>	list

ARGUMENTS

fanout_number
Specifies the number of external fanout points driven by the ports in *port_list*. Use this command only with ports. An error message occurs if *port_list* contains any nets. The input value is interpreted as an integer, non-integer values are truncated, and the integer portion is used as the fanout number.

port_list
Specifies a list of ports in the current design whose fanout numbers are to be set.

-max

-min

DESCRIPTION

This command sets the **fanout** attribute on specified ports in the current design. The fanout attribute sets the number of external fanout points driven by each port. This information is used to calculate an external wire load value for the corresponding port. The external wire load value is then added to the wire load of the net connected to the port. Setting this value to 0 when the driver of the port has no other internal fanout causes the net driving the port to be treated as unconnected, so it may be skipped for DRC.

Use the **report_port** command to view the load values on ports.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following sequence of commands describes the external fanout of an output port:

```
prompt> set_port_fanout_number 5 [get_ports 01]
```

```
prompt> set_wire_load_model [get_ports 01] \  
"default_wl"
```

SEE ALSO

- all_outputs(2)
- current_design(2)
- link(2)
- load_of(2)
- remove_attribute(2)
- report_port(2)
- reset_design(2)
- set_drive(2)
- set_load(2)
- target_library(3)

set_port_location

Annotates the specified top-level port with x- and y-coordinates and layer geometry, which the tool uses when running the **reoptimize_design** command.

SYNTAX

```
status set_port_location
[-coordinate {x y}]
[-layer_name layer_name]
[-layer_area {lx ly ux uy}]
[-append]
port_name
```

Data Types

<i>x</i>	float
<i>y</i>	float
<i>layer_name</i>	string
<i>lx</i>	float
<i>ly</i>	float
<i>ux</i>	float
<i>uy</i>	float
<i>port_name</i>	string

ARGUMENTS

-coordinate {x y}
Specifies the absolute location, in microns, of the x- and y-coordinates of the port.

-layer_name *layer_name*
Specifies the layer name of the top-level design port whose layer geometry is to be annotated.

-layer_area {lx ly ux uy}
Specifies the layer geometry of the port, in microns.

-append
Appends the location of port shapes to the same port. If this option is omitted, the location provided by the command overwrites the existing location.

port_name
Specifies the name of the top-level design port whose location is to be annotated.

DESCRIPTION

This command annotates the port location on the specified port. Executing this command without coordinates resets the port location. You can also set the port layer geometry by specifying layer name and layer area coordinates.

The location of the port specified with this command takes precedence over any annotation from a PDEF file using the **read_pdef** or **read_clusters** command. However, if the port location is reset by executing the **set_port_location** command without the x- or y-coordinates, any annotation from the PDEF file is used during optimization.

Layer area geometry is relative to the port origin.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example annotates port Z, a top-level port in the current design, with the location at 100, 5000.

```
prompt> set_port_location -coord {100 5000} Z
```

The following example resets the location of port Z. The tool uses physical information from any PDEF file to determine the location of port Z.

```
prompt> set_port_location Z
```

The following example sets the layer geometry of rectangle 10*20 of port Z.

```
prompt> set_port_location -layer_name METAL1 \  
      -layer_area {-5 -10 5 10} Z
```

SEE ALSO

set_power_gating_signal

Sets the attributes on the ports or pins in the design to indicate the power gating pins and the type of retention registers that will use the ports or pins to connect to or through the corresponding design hierarchy. This command is supported only in non-UPF mode.

SYNTAX

```
status set_power_gating_signal  
[-power_pin_index index]  
[-library_pin library_pin_name]  
[-type style]  
ports_or_pins
```

Data Types

<i>index</i>	integer
<i>library_pin_name</i>	string
<i>style</i>	string
<i>ports_or_pins</i>	list

ARGUMENTS

-power_pin_index *index*
Specifies the value of the **power_pin_class** attribute.

-library_pin *library_pin_name*
Specifies the pin name of of retention registers in the library to get the **power_pin_class** attribute.

-type *style*
Specifies the **power_gating_style** attributes.

ports_or_pins
Specifies a list of ports or pins on which the attributes are set.

DESCRIPTION

Power Compiler can hook up the control pins of retention registers through the design hierarchy to some primary input ports of the design or to some output pins of the driving cells in the design.

There are two steps to perform the stitching process. First, specify the ports or pins on the design hierarchy you want to use to be connected to a certain kind of control pins of a certain style of retention registers. Second, stitch the control pins across the design hierarchy.

The **set_power_gating_signal** command is used for the first step. There are 2 attributes to identify to the ports or pins on each design hierarchy what kind of control pins of what type of retention registers to be connected to. One is the **power_pin_class** attribute. It specifies the index of the control pin of a retention

register. The index is defined in the target library with the **power_gating_pin** complex attribute. As many as 5 control pins are supported for the retention registers. The first value of the complex can be `power_pin_[1-5]`. This corresponds to the 1 - 5 of the **power_pin_class** attribute. The other is **power_gating_style**. It specifies the type of retention registers involved.

The **power_pin_class** attribute is specified by the **-power_pin_index** option. You can use the **-library_pin** option as an alternative. The **-library_pin** option can use the names of control pin of retention registers in the target library to get the value of **power_pin_class** to be used for the command. The pin name is easier to use, but it requires that the corresponding control pins of retention registers in the target library have the same **power_gating_pin** attribute.

The **power_gating_style** attribute is specified by the **-type** option. It is the same as the value of the **power_gating_cell** attribute of some retention registers in the target library. If this option is not set, the port or pin will only be used for stitching the control pin of all types of retention registers.

The command can work on both ports of the current design, hierarchical input pin and output pin of a cell. If the command is used on the output pin of a cell, the stitching process will stop at this pin, otherwise it will go through the hierarchical pins to the primary input port of current design.

EXAMPLES

The following example sets the power gating style to *clock_low* and the power pin class to 1 on all pins with the name *my_pin* of cell U1:

```
prompt> set_power_gating_signal -type clock_low -power_pin_class 1 \  
[get_pin U1/my_pin]
```

The following example sets the **power_pin_class** attribute to a value from the **power_gating_pin** attribute of pin *my_pin* in the *target_library*:

```
prompt> set_power_gating_signal -library_pin my_pin [get_pin U1/pin1]
```

SEE ALSO

```
hookup_power_gating_ports(2)  
report_power_gating(2)  
set_power_gating_style(2)  
power_enable_power_gating(3)  
target_library(3)
```

set_power_gating_style

Sets the **power_gating_style** attributes on designs, cell instances, or HDL blocks, to specify the type of retention flip-flops in the target library that is to be used by the **compile** command.

SYNTAX

```
status set_power_gating_style  
-type style  
[-hdl_blocks hdl_block_list]  
[-hierarchy]  
[cell_or_design_list]
```

Data Types

<i>style</i>	string
<i>hdl_blocks</i>	list
<i>cell_or_design_list</i>	list

ARGUMENTS

-type *style*
Specifies the power gating style of the retention flip-flops in the target library that will be used by the **compile** command. The power gating style is defined in the target library by the **power_gating_cell** cell-level attribute.

-hdl_blocks *hdl_blocks*
Specifies the names of the HDL blocks in which the retention registers with the specified power gating style is to be used.

-hierarchy
Searches through the design hierarchy to find the HDL blocks and sets the power gating style. By default, power gating styles are set only on the HDL blocks in the current design. This option is intended for use with the **-hdl_blocks** option.
If you specify cells to the **set_power_gating_style** command, the **-hierarchy** option is not necessary. By default, the command searches through the design hierarchy for the specified cells.

cell_or_design_list
Specifies a list of leaf-level cells or designs in which to use the retention registers, with the specified power gating style. The default is to use the current design.

DESCRIPTION

This command specifies the power gating style on designs, cells, or HDL blocks to indicate the specific type of retention registers to use, from the target library, to map the sequential cells during the **compile** command. In the target library, the retention registers have the cell-level **power_gating_cell** attribute to specify the power gating styles of the registers.

If two **set_power_gating_style** commands specify the cells or HDL blocks containing the same cell, the last command overrides the previous command. The power gating style set on a design is the default style used on all sequential elements in the design. It has lower priority than the power gating style on HDL blocks and cells.

HDL block gating accepts wildcard characters, but the wildcard characters are only effective in one hierarchy.

This command requires a Power Compiler license and is only effective when the **power_enable_power_gating** variable is set to **TRUE**.

EXAMPLES

The following example specifies that for the current design, the retention registers defined in the target libraries, with the *CLK_FREE* style have to be used to map the sequential cells:

```
prompt> set_power_gating_style -type CLK_FREE
```

The following example indicates that the retention registers in the target libraries, with the *CLK_LOW* style are to be used for all HDL blocks whose name starts with *h2_*:

```
prompt> set_power_gating_style -type CLK_LOW \  
-hdl_blocks h2_*
```

The following example sets the *U1* and *U3* cells to the *CLK_FREE* style and *U2*, *U4*, and *U6* cells to the *CLK_LOW* style:

```
prompt> set_power_gating_style -type CLK_FREE {U1 U2 U3}  
prompt> set_power_gating_style -type CLK_LOW {U2 U4 U6}
```

The following example specifies that the *H1* design HDL block and the *U1*, *U2*, and *U3* cells have the *CLK_FREE* power gating style:

```
prompt> set_power_gating_style -type CLK_FREE\  
-hdl_blocks H1 {U1 U2 U3}
```

In the following example, assume that the *U1*, *U2*, and *U3* cells are in the *H1* HDL block. The following command specifies that the *U1*, *U2* and *U3* cells have the *CLK_FREE* power gating style, and other cells in *H1* have the *CLK_LOW* power gating style:

```
prompt> set_power_gating_style -type CLK_LOW -hdl_blocks H1  
prompt> set_power_gating_style -type CLK_FREE {U1 U2 U3}
```

SEE ALSO

```
current_design(2)  
get_attribute(2)  
remove_attribute(2)  
report_power_gating(2)  
reset_design(2)  
translate(2)
```

set_power_gating_style
2682


```
power_enable_power_gating(3)  
target_library(3)
```

set_power_guide

Sets an existing exclusive movebound as power guide or power well. This power guide will be used as always-on power_guide.

SYNTAX

```
status set_power_guidefp
-name exclusive_movebound_name
[-guard_band_x horizontal_guard_band_width]
[-guard_band_y vertical_guard_band_width]
```

Data Types

<i>exclusive_movebound_name</i>	string
<i>horizontal_guard_band_width</i>	integer
<i>vertical_guard_band_width</i>	integer

ARGUMENTS

-name *exclusive_movebound_name*
Indicates the name of an existing exclusive movebound which needs to be set as power guide or power well.

-guard_band_x *horizontal_guard_band_width*
Facilitates obtaining an integral width for the guard band in the horizontal direction.

-guard_band_y *vertical_guard_band_width*
Facilitates obtaining an integral width for the guard band in the vertical direction.

DESCRIPTION

This command sets an existing exclusive movebound as power guide. Currently this command is only used for making an exclusive movebound as always on power guide or power well.

The always on power guides in shut down voltage areas are used for placing the regular cells on the always on paths so that those cells become always on.

If there is no voltage area exist in the design, no exclusive movebound can be set as power guide. In a design with voltage area, if an exclusive movebound is contained in default voltage area and is set as power guide, addition or removal of a voltage area might results into incorrect inclusion or exclusion of power guides. It is recommended to set power guide only after the design is fixed with respect to voltage areas in it.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows a flow of the **set_power_guide** command. In this flow, AO_WELL physically belongs to voltage area VA_SHUTDOWN, and AO_WELL is set as an always_on power guide.

```
prompt> create_bounds -name AO_WELL\  
-coordinate {10 10 20 20} -exclusive  
prompt> create_voltage_area -name VA_SHUTDOWN\  
-coordinate {5 5 30 30} [get_cells A_INST]  
prompt> set_power_guide -name AO_WELL
```

SEE ALSO

```
create_bounds(2)  
create_voltage_area(2)  
report_power_guide(2)  
unset_power_guide(2)
```

set_power_net_to_voltage_area

Links, unlinks, or reports power nets for the specified voltage area.

SYNTAX

```
status set_power_net_to_voltage_area  
[-action link | unlink | report]  
[-voltage_area voltage_area_name_voltage_area_name]  
[-power_net net_name]
```

Data Types

<i>net_name</i>	string

ARGUMENTS

```
-action link | unlink | report  
    Specify the action of the power net to the specified voltage area. In the  
    "link" action, the p/g net name is linked to the voltage area. In the "unlink"  
    pattern, the p/g net name is unlinked to the voltage area. In the "report"  
    action, the p/g nets currently associated with the voltage area are printed.  
  
-voltage_area voltage_area_name_voltage_area_name  
    Specify the name of the voltage area.  
  
-power_net net_name  
    Specify the net name to be linked/unlinked to/from the voltage area.
```

DESCRIPTION

This command link, unlink, or report p/g net name to the specified voltage area for multi-vdd support.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

```
prompt> set_power_net_to_voltage_area -action link  
                                         -voltage_area test -power_net VDD
```

set_power_options

Specifies options for power optimization.

SYNTAX

```
status set_power_options  
[-default]  
[-leakage true | false]  
[-dynamic true | false]  
[-low_power_placement true | false]  
[-clock_gating true | false]  
[-leakage_effort low | high]  
[-dynamic_effort low | high]
```

ARGUMENTS

-default

Specifies default option for Power Optimization. By default only leakage optimization is set to true. All the options previously set will be reset with this option.

-leakage true | false

Specifies the option to the tool to run Leakage power optimization. By default the option is true.

-dynamic true | false

Specifies the option to the tool to run dynamic power optimization. Dynamic Power optimization runs only when the option is set to true. By default the option is false.

-low_power_placement true | false

Specifies the option to the tool to run power aware placement. When the option is set to true, the tool estimates high activity nets. This estimation is used to perform switching activity-based power aware placement that shortens the wire lengths of the nets with higher switching activities, while optimizing timing, power and congestion, concurrently. This feature is available in place_opt -power, clock_opt -power, and optimize_pre_cts_power commands. When used in clock_opt -power or optimize_pre_cts_power, the tool performs clock tree estimation to consider high activity clock nets as well as signal nets. In case that area is the biggest concern so a very light place_opt is required, low power placement technology is not recommended in place_opt with -effort low; Even though overall dynamic power can be improved, area could be worse, due to more extra steps for placement and optimization in place_opt. By default the option is false.

-clock_gating true | false

Specifies the option to the tool to perform clock gate restructuring in pre-CTS power optimization. This feature is only available in clock_opt -power and optimize_pre_cts_power commands. By default the option is false.

-leakage_effort low | high

Specifies the option to run low effort or high effort leakage opto. Low effort

leakage opto does cell swapping only. High effort leakage opto produces better leakage power but takes additional runtime. By default leakage optimization runs in high effort.

`-dynamic_effort low | high`

Specifies the option to run low effort or high effort dynamic power optimization. By default dynamic power optimization is low-effort. High effort dynamic power optimization causes long overhead in runtime.

DESCRIPTION

This command provides the user a flexible interface to control different power optimization techniques inside the tool. These options work in conjunction with mega commands like `place_opt`, `clock_opt` and `route_opt`, when power optimization is enabled by `-power` switch. It also works with `psynopt` command. Certain power optimization features are not available in certain mega commands. For example, `route_opt` supports only leakage power optimization. When the options are specified, the power constraints are optional. However it is recommended to provide power constraints to the tool, so that the tool spend less runtime trying more and more to optimize for power to meet the constraints.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows how to set power optimization options

```
prompt> set_power_options -leakage true -dynamic true
```

```
prompt> place_opt -power
```

```
prompt> set_power_options -low_power_placement true
```

```
prompt> place_opt -power
```

```
prompt> psynopt -power
```

```
prompt> clock_opt -power
```

```
prompt> route_opt -power
```

SEE ALSO

`place_opt(2)`

`clock_opt(2)`

`route_opt(2)`

set_power_switch_cell

On-The-Fly setting of library power switch cell.

SYNTAX

```
Boolean set_power_switch_cell  
cell_name  
[-cell_type power_switch_type]  
[-is_macro]  
[-switch_pin {pin_name}]  
[-pg_pin {pin_name switch_function pg_function}]  
[-reset]
```

Data Types

<i>cell_name</i>	string
<i>pin_name</i>	string
<i>switch_function</i>	string
<i>pg_function</i>	string

ARGUMENTS

cell_name
Specify the library cell as isolation cell.

-cell_type *power_switch_type*
Specify the power switch cell type, it's either coarse_grain type or fine_grain type.

-switch_pin {*pin_name*}
Specify the library cell power switch pin.

-pg_pin {*pin_name* *switch_function* *pg_function*}
Specify the library cell power/ground pin, it should be in {*pin_name* *switch_function* *pg_function*} format.

-reset
reset the retention cell attributes.

DESCRIPTION

The **set_power_switch_cell** command set all the cells with the same name specified as power switch cell.

If -reset is issued, the retention cell attributes will be cleaned up.

If -reset is combined with other options, reset happens first, other options will be set on the cell after reset.

EXAMPLES

```
prompt> set_power_switch_cell FOOT* -cell_type coarse_grain -switch_pin SLEEPN -  
pg_pin {VSS !SLEEPN VSS} -switch_pin SLEEPN
```

SEE ALSO

```
report_lib(2)  
check_library(2)
```


set_prefer

Sets the **preferred** attribute on specified library cells.

SYNTAX

```
integer set_prefer  
[-min]  
cell_list
```

Data Types

cell_list list

ARGUMENTS

-min
Specifies the library cells that should be preferred during hold violation (minimum path) fixing.

cell_list
Specifies the list of cells on which to set the **preferred** attribute. Cell names must contain a library prefix. If more than one cell name is specified, enclose the list using quotation marks (") or braces ({}).

DESCRIPTION

The **set_prefer** command sets the **preferred** attribute on specified library cells. The cells must be in one of the target libraries, or in a library already loaded into Design Analyzer. If this attribute exists on a library cell, the **compile** command uses the library cell attribute before other attributes in the target library that implement the same function during technology translation using the **translate** command.

Note that the effect of the **preferred** attribute might not be noticeable during optimization because non-preferred cells can be chosen to help meet constraints.

If the **-min** option is used, the library cells specified will be preferred during hold violation fixing. Only buffers and inverters, which are used to fix hold violations, should be included in the cell list. Including cells with any other functionality in the list with the **-min** option will have no impact on the optimization engine. This option should be used carefully, because the preferred cells will be used for minimum path fixing, even if it results in a worse area or cell count solution.

The **set_prefer** command affects only the version of the library that is currently loaded into memory, and has no effect on the version that exists on disk. However, if the library is written using **write_lib**, the **preferred** attribute is also written, causing cells to be permanently preferred.

To remove the **preferred** attribute, use the **remove_attribute** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command specifies that the GA and GB cells from the tech_lib library be preferred during optimization.

```
prompt> set_prefer {tech_lib/GA tech_lib/GB}
```

SEE ALSO

```
remove_attribute(2)  
report_lib(2)  
target_library(3)
```

set_preferred_routing_direction

Sets the preferred routing direction for the specified routing layers.

SYNTAX

```
string set_preferred_routing_direction  
-layers list_of_layers  
-direction horizontal | vertical
```

ARGUMENTS

```
-layers list_of_layers  
    Specify the list/collection of layer(s) for which the preferred routing  
    directions is to be set.  
  
-direction horizontal | vertical  
    Specify the preferred routing direction. The allowed values are: horizontal,  
    vertical, h, v, HORIZONTAL, VERTICAL, H, V. Specify any one.
```

DESCRIPTION

The `set_preferred_routing_direction` command is used to set the preferred direction for the specified routing layers. This command will be useful in overriding the default layer directions specified in the library itself or even the design. The setting of layer direction done through this command will become specific to this design only.

Note: The command will issue suitable warnings whenever adjacent layers have the same direction after the execution of the command.

Note: This command overrides the preferred direction specified in the reference libraries and saves the user setting as persistent data (will be written when user saves the design) in the design database.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example assigns layers "m1" & "m2" the direction horizontal.

```
prompt> set_preferred_routing_direction \  
    -layers {m1 m2} -direction horizontal
```

SEE ALSO

```
remove_preferred_routing_direction(2)  
report_preferred_routing_direction(2)
```

set_preroute_advanced_via_rule

Sets preroute advanced via rule options.

SYNTAX

```
integer set_preroute_advanced_via_rule
[-move_via_to_center]
[-offset_both_sides]
[-x_offset_recommended]
[-y_offset_recommended]
[-x_offset distance]
[-y_offset distance]
[-x_step distance]
[-y_step distance]
[-cut_layer layer]
[-contact_code contact]
[-size_by_via_area {distance distance}]
[-size_by_array_dimensions {int int}]
[-rotation_mode on / required / off]
```

Data Types

distance distance

ARGUMENTS

-move_via_to_center

Move advanced via to center of the wire intersection area. When this option is on, all options related to offsets, are irrelevant. Default is not to move via to center.

-offset_both_sides

Applies the X/Y offsets to both lower-left and upper-right corner of the wire intersection area. By default, the offsets are applied only to the lower-left corner of the wire intersection area.

-x_offset_recommended

Select to apply the X offset, specified in *x_offset*, if possible. If this option is skipped, the X offset is forcibly applied.

-y_offset_recommended

Select to apply the Y offset, specified in *y_offset*, if possible. If this option is skipped, the Y offset is forcibly applied.

-x_offset *distance*

Determines the location for the first advanced via by specifying the X offset from the lower-left corner of the wire intersection area. The default is **0.0**.

-y_offset *distance*

Determines the location for the first advanced via by specifying the Y offset from the lower-left corner of the wire intersection area. The default is **0.0**.

set_preroute_advanced_via_rule

-x_step *distance*

Specifies the distance to put between the advanced vias. The value you type for *distance* is in user units and is the distance between the two left sides of adjacent via arrays in the X direction. When there are design rule checking violations, some vias might not be created. The default is **0.0**.

-y_step *distance*

Specifies the distance to put between the advanced vias. The value you type for *distance* is in user units and is the distance between the two left sides of adjacent via arrays in the Y direction. When there are design rule checking violations, some vias might not be created. The default is **0.0**.

-cut_layer *layer*

Specifies via layer name or number from the technology file. This option is mutually exclusive with *contact_code* option. If either option is skipped, the other options of this command are applied to all non-specified via layers by default.

-contact_code *contact*

Specifies via contact code name or number from the technology file. This option is mutually exclusive with *cut_layer* option. If either option is skipped, the other options of this command are applied to all non-specified via layers by default.

-size_by_via_area {*distance distance*}

Specifies the X by Y via array area in user units. This option is mutually exclusive with *size_by_array_dimensions* option.

-size_by_array_dimensions {*int_int*}

Specifies the number of via columns and rows. This option is mutually exclusive with *size_by_via_area* option.

-rotation_mode on | required | off

Specifies the via rotation mode. It's on by default.

DESCRIPTION

The **set_preroute_advanced_via_rule** command sets the rules of creating via arrays at wire intersections per specified via layer or contact code for some preroute commands. It is sometimes advantageous to create a matrix of small via arrays instead of one large via array at wire intersections. Using smaller via arrays may free some routing space.

It is similar to *special_via_rule* of some preroute commands, except it's advanced in a sense that *special_via_rule* provides the means to specify the same conditions for all layers only, and **set_preroute_advanced_via_rule** allows to specify different conditions for different layers.

In order to create vias in accordance with **set_preroute_advanced_via_rule** settings, user has to turn the "advanced_via_rules" option On in preroute commands that support advanced via rules. Right now only "create_power_straps" and "preroute_standard_cells" commands have the "advanced_via_rules" option.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows how to set **set_preroute_advanced_via_rule** for cut layers V1 through V3, with different array dimension sizes.

```
set_preroute_advanced_via_rule -cut_layer V1 -size_by_array_dimensions {2 1}  
set_preroute_advanced_via_rule -cut_layer V2 -size_by_array_dimensions {2 2}  
set_preroute_advanced_via_rule -cut_layer V3 -size_by_array_dimensions {1 1}
```

SEE ALSO

```
create_power_straps(2)  
preroute_standard_cells(2)
```

set_preroute_drc_strategy

Sets preroute drc options to change the internal rules for design rule checking in the following power and ground commands: **create_rectangular_rings**, **create_pad_rings**, **create_power_straps**, **create_preroute_vias**, **preroute_instances**, and **preroute_standard_cells**.

SYNTAX

```
status set_preroute_drc_strategy
[-spacing_rules radial | manhattan]
[-protect_pin_access_edge]
[-protect_pin_access_edge_within_pin_layer_pitch]
[-protect_pin_access_edge_within_range protect_range]
[-treat_fat_blockage_as_fat_wire]
[-use_fat_via]
[-ignore_std_cells]
[-ignore_top_level_pins]
[-ignore_same_net_check]
[-quick_check]
[-merge_thin_wires]
[-no_design_rule]
[-report_fail]
[-jog_range jog_range_distance]
[-min_layer min_layer]
[-max_layer max_layer]
[-honor_shapes_of_nets collection_of_nets | ALL]
```

Data Types

<i>protect_range</i>	distance
<i>jog_range_distance</i>	distance
<i>min_layer</i>	string
<i>max_layer</i>	string
<i>collection_of_nets</i>	collection

ARGUMENTS

- spacing_rules radial | manhattan
Specifies the mode for design rule checking (DRC). The valid *rule_mode* values are **radial** or **manhattan**. A **radial** rule checks for spacing violations within a circular area. A **manhattan** rule checks for spacing violations within the defined spacing distance horizontally and vertically. The default value is **radial**.
- protect_pin_access_edge
Prevents the signal pin access edge from being blocked by the Prerouter.
- protect_pin_access_edge_within_pin_layer_pitch
Protects the pin access edge within the double pin layer pitch distance. The default is determined by the following formula:
$$2 * \text{the MAX pitch in the tech file.}$$

-protect_pin_access_edge_within_range *protect_range*
 Protects the pin access edge within the specified range. The default is determined by the following formula:

$$2 * \text{the MAX pitch in the tech file}$$

-treat_fat_blockage_as_fat_wire
 Treats fat blockages as fat wires. By default, DRC treats fat blockages as thin wires.

-use_fat_via
 Uses fat vias for non-turning layer changes if either wire meets the width requirement for a fat via.

-ignore_std_cells
 Turns off DRC against objects in standard cells.

-ignore_top_level_pins
 Ignores top-level pins in DRC. The DRC with such pins is disabled.

-ignore_same_net_check
 Ignores the same net metal-to-metal in DRC.

-quick_check
 Skips the final stage of DRC when a possible same net metal-to-metal error is found. On that stage, the DRC engine searches for other metal objects that cover the insufficient gap between the two objects; and, if the gap is covered, the error is considered to be fake. While using this option might decrease runtime, it might cause missing connections.

-merge_thin_wires
 Specifies for the prerouter to check and merge thin wires into fat wires for DRC. Using this option might increase runtime.

-no_design_rule
 Turns off all DRC.

-report_fail
 Reports detailed information about the failed power and ground command operation.

-jog_range *jog_range_distance*
 Sets a jogging range distance. Jogging range defines the maximum distance for a dogleg connection on the same layer; it is not for layer switching. Set a range within which the prerouter might perform limited jogging when necessary. The default is **0.0** (as in no jogging).

-min_layer *min_layer*
 Sets the minimum metal layer used for prerouting. A layerNumber or maskName from the technology file is a valid *min_layer* value. The default value is set to the bottom routing layer.

-max_layer *max_layer*
 Sets the maximum metal layer used for prerouting. A layerNumber or maskName from the technology file is a valid *max_layer* value. The default value is set to the top routing layer.

`-honor_shapes_of_nets collection_of_nets | ALL`

Sets the list of nets for which prerouter commands should honor routing. Use the *collection_of_nets* argument to specify a list or the keyword **ALL** to honor the routing of all nets. By default, if a net is not marked as power or ground, then the command ignores all of its routing shapes.

DESCRIPTION

This command changes the internal rules for design rule checking (DRC) in the following PG commands: **create_rectangular_rings**, **create_pad_rings**, **create_power_straps**, **create_preroute_vias**, **preroute_instances**, and **preroute_standard_cells**

The DRC rules are still taken from Technology File, but this command changes the rule application within the scope of power and ground commands. If the same internal rules are applied for all power and ground commands, the **set_preroute_drc_strategy** command can be executed only once before any of the power and ground commands are executed. These settings are not persistent in the design. This command is optional, if you only want defaults to apply. All the options in this command are optional, as well.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets manhattan rule, prohibits blocking of the pin access edge, directs the prerouter to perform limited jogging in the range of **100** user units, using layers from metal 3 (**M3** in .tf) to metal 5 (**M5** in .tf).

```
prompt> set_preroute_drc_strategy -spacing_rules manhattan \  
-protect_pin_access_edge -jog_range 100 \  
-min_layer M3 -max_layer M5
```

The following example sets the routing of all nets to honor.

```
prompt> set_preroute_drc_strategy -honor_shapes_of_nets ALL
```

The following example sets the routing of the nets named **timer_hmsb[1]** and **io_if/D1935** to be honored.

```
prompt> set_preroute_drc_strategy -honor_shapes_of_nets \  
[get_nets [list timer_hmsb[1] io_if/D1935]]
```

SEE ALSO

`create_pad_rings(2)`
`create_power_straps(2)`

```
create_preroute_vias(2)
create_rectangular_rings(2)
preroute_instances(2)
preroute_standard_cells(2)
report_preroute_drc_strategy(2)
```

set_preroute_special_rules

Defines a set of special rules to be honored by the **create_power_straps** and **preroute_instances** preroute commands.

SYNTAX

```
status set_preroute_special_rules
-name name
[-do_not_connect_pins]
[-extension_layers ext_layers_list]
[-extension_width segments_width]
[-extension_space space_between_segments]
[-extension_count {min_count max_count}]
[-extension_via_array {columns_count rows_count}]
[-extension_try_jog conn_length_threshold]
[-leave_space_for_io_connections]
[-resolve_conflicts_by_jogs]
[-connection_point_default {offset pitch}]
[-connection_point_layers conn_pt_layers_list]
[-connection_point_offsets offsets_list]
[-connection_point_pitches pitches_list]
[-first_strap_offset first_strap_offset]
[-strap_endcap strap_endcap_distance]
[-create_local_straps min_dist_threshold]
[-keep_straps_in_standard_cell_area]
[-macro_cells {macro_cells_collection}]
[-pin_layer pin_layer_name]
[-strap_to_pin_offset strap_to_pin_offset]
[-end_point_offset end_point_offset_distance]
[-target_area {{dx-left dy-bottom} {dx-right dy-top}}]
[-target_layer target_layer_name]
[-jog_horizontal_layers h_layers_list]
[-jog_vertical_layers v_layers_list]
[-jog_width {h_segments_width v_segments_width}]
```

Data Types

<i>name</i>	string
<i>ext_layers_list</i>	string
<i>segments_width</i>	distance
<i>space_between_segments</i>	distance
<i>min_count</i>	integer
<i>max_count</i>	integer
<i>columns_count</i>	integer
<i>rows_count</i>	integer
<i>conn_length_threshold</i>	distance
<i>offset</i>	distance
<i>pitch</i>	distance
<i>conn_pt_layers_list</i>	collection
<i>offsets_list</i>	distance
<i>pitches_list</i>	distance
<i>first_strap_offset</i>	distance
<i>strap_endcap_distance</i>	distance

<i>min_dist_threshold</i>	distance
<i>macro_cells_collection</i>	collection
<i>pin_layer_name</i>	string
<i>strap_to_pin_offset</i>	distance
<i>end_point_offset_distance</i>	distance
<i>target_layer_name</i>	string
<i>h_layers_list</i>	collection
<i>v_layers_list</i>	collection
<i>h_segments_width</i>	distance
<i>v_segments_width</i>	distance

ARGUMENTS

-name *name*

Specifies a name for the set of rules. This name is used to reference the rules in subsequent **create_power_straps** and **preroute_instances** commands.

-do_not_connect_pins

Prevents connections between created routing shapes and pins on other layers.

-extension_layers *ext_layers_list*

Sets a list of metal layers on which the tool must create connecting segments. Each layer is specified by its name.

This extension option controls an aspect of the geometry of power strap connections (created by the **create_power_straps** command) or I/O pads (created by the **preroute_instances** command) when it is impossible to make a connection by using only the primary metal layer. For more information on these options, see the *DESCRIPTION* section. To turn the rules on, you must use all of the following options: **-extension_width**, **-extension_space**, **-extension_count**, and **-extension_via_array**.

-extension_width *segments_width*

Sets the width of the segments in the connection. The minimum and maximum widths specified in the technology file override this setting. If the specified width is less than the minimum width specified in the technology file, the tool uses the minimum width from the technology file. If the specified width is greater than the maximum width specified in the technology file, the tool uses the maximum width from the technology file.

This extension option controls an aspect of the geometry of power strap connections (created by the **create_power_straps** command) or I/O pads (created by the **preroute_instances** command) when it is impossible to make a connection by using only the primary metal layer. For more information on these options, see the *DESCRIPTION* section. To turn the rules on, you must use all of the following options: **-extension_width**, **-extension_space**, **-extension_count**, and **-extension_via_array**.

-extension_space *space_between_segments*

Sets the spacing between connecting segments on the same layer. If the specified spacing is less than the minimum spacing defined in the technology file, the tool uses the minimum spacing from the technology file.

This extension option controls an aspect of the geometry of power strap connections (created by the **create_power_straps** command) or I/O pads (created by the **preroute_instances** command) when it is impossible to make a connection by using only the primary metal layer. For more information on these options,

set_preroute_special_rules

see the *DESCRIPTION* section. To turn the rules on, you must use all of the following options: **-extension_width**, **-extension_space**, **-extension_count**, and **-extension_via_array**.

-extension_count {*min_count* *max_count*}

Sets a range of required parallel connections on one metal layer. If the number of those connections is less than *min_count*, the entire connection is not created. The following condition must be true: $1 \leq \text{min\_count} \leq \text{max\_count}$.

This extension option controls an aspect of the geometry of power strap connections (created by the **create_power_straps** command) or I/O pads (created by the **preroute_instances** command) when it is impossible to make a connection by using only the primary metal layer. For more information on these options, see the *DESCRIPTION* section. To turn the rules on, you must use all of the following options: **-extension_width**, **-extension_space**, **-extension_count**, and **-extension_via_array**.

-extension_via_array {*columns_count* *rows_count*}

Sets the dimensions of the via arrays connecting the segment on the primary metal layer.

This extension option controls an aspect of the geometry of power strap connections (created by the **create_power_straps** command) or I/O pads (created by the **preroute_instances** command) when it is impossible to make a connection by using only the primary metal layer. For more information on these options, see the *DESCRIPTION* section. To turn the rules on, you must use all of the following options: **-extension_width**, **-extension_space**, **-extension_count**, and **-extension_via_array**.

-extension_try_jog *conn_length_threshold*

Sets a threshold for the length of a connection. If the length of a segment in a connection is more than this value, the command tries to find a target in other directions to minimize the length of the connection.

The **fb-extension_try_jog** option is an extension option. To use it, you must also use all of the other extension options: **-extension_width**, **-extension_space**, **-extension_count**, and **-extension_via_array**.

-leave_space_for_io_connections

Specifies that the **create_power_straps** command should not create a strap extension in a location that could be used to connect I/O pads to a power ring.

You can set this option only if *both* of the following conditions are met:

- More than *min_count* (as specified in the **-extension_count** option) parallel connections might be created.
- The extension rules are turned on. To turn the rules on, you must use the following options: **-extension_width**, **-extension_space**, **-extension_count**, and **-extension_via_array**.

-resolve_conflicts_by_jogs

Causes the **preroute_instances** command to create a jogged connection instead of a direct connection when it connects an I/O pad to a power ring if the direct connection would cause a design rule checking (DRC) violation.

You can set this option only if the extension rules are turned on as described in the *DESCRIPTION* section. To turn the rules on, you must use all of the

following options: **-extension_width**, **-extension_space**, **-extension_count**, and **-extension_via_array**.

-connection_point_default *{offset pitch}*

Specifies the offset of the first via array from the low boundary of the target and the pitch between neighbor arrays. If the target is located on a layer specified in the **-connection_point_layers** option, then the values specified in the **-connection_points_offset** and **-connection_points_pitch** options override these values. The default value for the **-connection_point_default** option is given in the format $\{W/2\ W\}$, where W is width of the target, so the *offset* value is determined by the formula $W/2$ and the *pitch* value is W .

This connection option controls one aspect of creating the coordinates for the center of the via arrays that connect a segment to a target for a specific layer. If you do not specify any of these options, vias are created as usual. For more information on control options, see the *DESCRIPTION* section. The control options are: **-connection_point_default**, **-connection_point_layers**, **-connection_point_offsets**, and **-connection_point_pitches**.

-connection_point_layers *conn_pt_layers_list*

Specifies a list of layers. If a target is located on a layer from this list, the offset and pitch for the connection points are taken from the **-connection_points_offset** and **-connection_points_pitch** options, respectively. Each layer is specified by its name.

This connection option controls one aspect of creating the coordinates for the center of the via arrays that connect a segment to a target for a specific layer. If you do not specify any of these options, vias are created as usual. For more information on control options, see the *DESCRIPTION* section. The control options are: **-connection_point_default**, **-connection_point_layers**, **-connection_point_offsets**, and **-connection_point_pitches**.

-connection_point_offsets *offsets_list*

Specifies the offset, as a list of distances, used for each layer specified in the **-connection_point_layers** option. If you do not specify the **-connection_point_offsets** option, the offset is taken from the **-connection_point_default** option. If the length of the list of offsets is less than number of layers specified in the **-connection_point_layers** option, the last value applies to the other layers.

This connection option controls one aspect of creating the coordinates for the center of the via arrays that connect a segment to a target for a specific layer. If you do not specify any of these options, vias are created as usual. For more information on control options, see the *DESCRIPTION* section. The control options are: **-connection_point_default**, **-connection_point_layers**, **-connection_point_offsets**, and **-connection_point_pitches**.

-connection_point_pitches *pitches_list*

Specifies the pitch, as a list of distances, for each layer specified in the **-connection_point_layers** option. If you do not specify the **-connection_point_pitches** option, the pitch is taken from the **-connection_point_default** option. If the length of the list of pitches is less than number of layers specified in the **-connection_point_layers** option, the last value applies to the other layers.

This connection option controls one aspect of creating the coordinates for the center of the via arrays that connect a segment to a target for a specific layer. If you do not specify any of these options, vias are created as usual.

For more information on control options, see the *DESCRIPTION* section. The control options are: **-connection_point_default**, **-connection_point_layers**, **-connection_point_offsets**, and **-connection_point_pitches**.

-first_strap_offset *first_strap_offset*

Specifies the offset of the first power strap created by the **create_power_straps** command. This information is used by the **create_power_straps -configure rows** command when the direction of the straps is the same as the direction of the placement rows. The value should be equal to the difference between central line of the first strap to be created and the low boundary of the first row. The default value is half of width of the strap.

-strap_endcap *strap_endcap_distance*

Specifies information used by the **create_power_straps -configure rows** command when the direction of the straps differs from the direction of the placement rows. It specifies the minimum extension of a strap over the boundary box of the rows. The default value is 0.

-create_local_straps *min_dist_threshold*

Specifies information used by the **create_power_straps -configure rows** command when the direction of the straps differs from the direction of the placement rows. It specifies the minimum distance from the end points of a row where at least one strap should be created. If there are no straps in that area, the tool must create additional local straps.

-keep_straps_in_standard_cell_area

Specifies information used by the **create_power_straps -configure rows** command when the direction of the straps differs from the direction of the placement rows. It controls whether straps can be broken by the areas that are not covered by placement rows. Usually, such areas are dedicated for macro cells.

-macro_cells {*macro_cells_collection*}

Specifies information used by the **create_power_straps -configure macros** command to determine which macro cells to create straps over. If you do not specify this option, straps are created over all macro cells in the design.

-pin_layer *pin_layer_name*

Specifies information used by the **create_power_straps -configure macros** command. It specifies the layer containing the pins for which straps are to be created. Straps are created only for the pins on the specified layer. This option is mandatory when the straps are configured by macro cells.

-strap_to_pin_offset *strap_to_pin_offset*

Specifies information used by the **create_power_straps -configure macros** command. It specifies the offset from the center line of the pin for which the strap is created to the center line of the created strap. The default value is 0.

-end_point_offset *end_point_offset_distance*

Specifies information used by the **create_power_straps -configure macros** command. It specifies the distance from the macro cell boundary to the end points of a strap before its extension (if it is required). The default value is 0.

-target_area *{{dx-left dy-bottom} {dx-right dy-top}}*
 Specifies information used by the **create_power_straps -configure macros** command. It specifies a rectangle where the straps are connected to a target. Coordinates of the rectangle are calculated as follows:
 $X\text{-left} = \text{macro\_X\_left} - dx\text{-left};$
 $Y\text{-bottom} = \text{macro\_Y\_bottom} - dy\text{-bottom};$
 $X\text{-right} = \text{macro\_X\_right} + dx\text{-right};$
 $Y\text{-top} = \text{macro\_Y\_top} + dy\text{-top};$
 If you do not specify this option, no connections are required. If you specify this option, the **-target_layer** option and at least one connection to a target in that area are required.

-target_layer *target_layer_name*
 Specifies information used by the **create_power_straps -configure macros** command. It specifies the name of the layer on which to search for targets. This option is required if you use the **-target_area** option.

-jog_horizontal_layers *h_layers_list*
 Specifies information used by the **create_power_straps -configure macros** command. It specifies a list of layers that can be used for the horizontal segments of jogged connections. The layers are specified by name. This option is required if jogged connections are allowed.

-jog_vertical_layers *v_layers_list*
 Specifies information used by the **create_power_straps -configure macros** command. It specifies a list of layers that can be used for vertical segments of jogged connections. The layers are specified by name. This option is required if jogged connections are allowed.

-jog_width *{h_segments_width v_segments_width}*
 Specifies information used by the **create_power_straps -configure macros** command. It specifies the width of the horizontal and vertical segments of jogged connections. By default, both the horizontal and vertical widths are equal to the width of a strap.

DESCRIPTION

This command creates a set of special rules that must be observed by the **create_power_straps** and **preroute_instances** commands. These commands refer to these special rules by the name specified in the **-name** option. You can specify as many sets of special rules as you want.

Extension Options

Several extension options control the geometries of power strap connections (created by the **create_power_straps** command) or I/O pads (created by the **preroute_instances** command) when it is impossible to make a connection by using only the primary metal layer. The extension options are the following:

-extension_width
-extension_space
-extension_count


```
-extension_via_array
-extension_try_jog
```

Usually, these extension options are needed if the connected straps or pads have a much bigger width than the maximum width allowed on other metal layers and, so, redundant connections are required. To turn the rules on, you must specify all of the extension options, except **-extension_try_jog**.

Connection Options

Several connection options specify the coordinates for the center of via arrays that connect a segment to a target for a specific layer. If you do not specify any of these options, vias are created as usual. The connection options are the following:

```
-connection_point_default
-connection_point_layers
-connection_point_offsets
-connection_point_pitches
```

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates power straps in the same direction (horizontal) as the direction of placement rows. The location of the first (bottom) strap is specified by an offset of its center line from the bottom boundary of the core area (that is, use the **-first_strap_offset 1.48** option). No vias should be dropped to connect any pins. The rules specify the following requirements for connecting end points of straps to targets when it is necessary to use segments on other metal layers:

- At least four parallel segments must be created on each of the M7, M5, and M3 layers (that is, use the **-extension_count {4 99}** option).
- Each of those segments must have a width equal to 1.5.
- The space between neighboring segments on the same layer must be equal to 0.5.
- A jogged connection is used when the length of a segment is more than 64.
- Via arrays connecting a strap with a segment must have a size of 6x2.
- If more than four parallel connections might be created on each layer, they are created if there are no I/O pads of other power and ground nets to connect in that area (that is, use the **-leave_space_for_io_connections** option).

```
prompt> set_preroute_special_rules -name specGS\
-first_strap_offset 1.48\
-do_not_connect_pins\
```

```

-extension_layers {M7 M5 M3}\
-extension_width 1.5\
-extension_space 0.5\
-extension_count {4 99}\
-extension_via_array {6 2}\
-extension_try_jog 64\
-leave_space_for_io_connections

```

```

prompt> create_power_straps\
-nets {VSS VDD}\
-configure rows\
-direction horizontal\
-layer PM\
-width 8.0\
-pitch_within_group 10.08\
-step 20.16\
-extend_low_ends to_first_target\
-extend_high_ends to_first_target\
-keep_floating_wire_pieces\
-ignore_parallel_targets\
-special_rules specGS

```

The following example creates power straps orthogonal to the placement row direction (vertical). No vias should be dropped to connect any pins. If a strap is located under or over a wide target (not a pin), the following special via rules are applied:

- The pitch of the via arrays should be equal to 6.96 and the offset of the first one (from the bottom boundary of the target) should be equal to 0.52.
- Straps are not created in areas if there are no placement rows (that is, use the **-keep_straps_in_standard_cell_area** option).
- Each strap is extended by 0.24 over the boundary of the first and last placement row in the area.

```

prompt> set_preroute_special_rules -name specLMVstd\
-do_not_connect_pins\
-connection_point_default {0.52 6.96}\
-straps_endcap 0.24\
-create_local_straps 14.15\
-keep_straps_in_standard_cell_area

```

```

prompt> create_power_straps\
-nets {VDD VSS}\
-configure rows\
-direction vertical\
-layer M4\
-width 0.44\
-pitch_within_group 0.66\
-step 29.4\
-advanced_via_rules\
-extend_low_ends to_first_target\
-extend_high_ends to_first_target\

```

```

-do_not_route_over_macros\
-keep_floating_wire_pieces\
-ignore_parallel_targets\
-special_rules specLMVstd

```

The following example creates vertical power straps over a macro cell if it has pins on the M4 metal layer. The distance between the end points of a strap and the boundary of the macro cell should be equal to 0.57. If a strap is located under or over a wide target (not a pin), the following special via rules are applied:

- The pitch of the via arrays should be equal to 6.96.
- The offset of the first via array (from the bottom boundary of the target) should be equal to 0.52.

```

prompt> set_preroute_special_rules -name specLMVmem\
-end_point_offset 0.57\
-connection_point_default {0.52 6.96}\
-pin_layer M4

```

```

prompt> create_power_straps\
-nets {VDD VSS}\
-configure macros\
-direction vertical\
-layer M4\
-advanced_via_rules\
-extend_low_ends off\
-extend_high_ends off\
-special_rules specLMVmem

```

The following example connects I/O pads to a ring. If it is necessary to use segments on other metal layers, the rules specify the following requirements for connections:

- At least eight parallel segments must be created on each of the M7, M5, and M3 metal layers (that is, use the **-extension_count {8 99}** option).
- Each of those segments must have width equal to 1.5.
- The space between neighboring segments on the same layer must be equal to 0.5.
- Via arrays connecting a pin with a segment must have a size of 20x2.
- If there are less than eight parallel connections on each layer, the additional ones should be created with jogs (that is, use the **-resolve_conflicts_by_jogs** option).

```

prompt> set_preroute_special_rules -name specIOhor\
-extension_layers {M7 M5 M3}\
-extension_width 1.5\
-extension_space 0.5\
-extension_count {8 99}\
-extension_via_array {20 2}\

```

-resolve_conflicts_by_jogs

```
prompt> preroute_instances\  
-nets VDD\  
-ignore_macros\  
-ignore_cover_cells\  
-skip_bottom_side\  
-skip_top_side\  
-primary_routing_layer pin\  
-route_pins_on_layer PM\  
-special_rules specIOhor
```

SEE ALSO

```
create_power_straps(2)  
preroute_instances(2)
```

set_primetype_options

Sets up PrimeTime options for the **signoff_opt** command.

SYNTAX

```
status set_primetype_options  
[-default]  
[-exec_dir string]  
[-sdc_file string]  
[-max_image string]  
[-min_image string]  
[-setup_slack_limit float]  
[-hold_slack_limit float]  
[-common_file string]  
[-specific_file string]  
[-license_limit integer]
```

ARGUMENTS

[-default]
Resets all options to default.

[-exec_dir *string*]
Specifies the UNIX path to the PrimeTime shell executable `pt_shell`. The path can be either relative or complete.

[-sdc_file *string*]
Specifies the golden signoff SDC file for PrimeTime. This is an optional argument, and if it is missing, PrimeTime will use the constraints stored in the Milkyway.

[-max_image *string*]

[-min_image *string*]
Specifies the saved session directory from a **save_session** command in PrimeTime. **signoff_opt** will load the session as max/min corner with **restore_session**, and continue optimization. Because the PrimeTime was run standalone, it is required that you must maintain the consistency between the PrimeTime run and IC Compiler, such as netlist, constraints and parasitic, when this switch is used. An inconsistent netlist, for example, PrimeTime's netlist has one more buffer than IC Compiler, can lead to unexpected results when running **signoff_opt**.

[-setup_slack_limit *float*]

[-hold_slack_limit *float*]
Specifies the setup/hold slack limit for optimization, such that optimization will only work on the paths that have at least the specified slacks.

[-common_file *string*]
Specifies a tcl file that will be sourced to PrimeTime immediately after PrimeTime is invoked. Only TCL variables are allowed in this file.

`[-specific_file string]`

Specifies a tcl file that will be sourced to PrimeTime after loading the constraints. Only analysis commands are allowed in this file. For BCWC analysis, you can specify a list of two files.

`[-license_limit integer]`

Specifies maximum number of PT/PTSI licenses that will be used. This option is only valid with multi mode multi corner analysis using PT DMSA.

DESCRIPTION

This command sets up PrimeTime options for **signoff_opt**. Use **report_primetime_options** command to examine the options.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example runs the **set_primetime_options** command.

```
prompt> open_mw_cel -lib my_lib my_cel
prompt> set_primetime_options\
-exec_dir /PrimeTime/linux/syn/bin
prompt> save_mw_cel
```

The following example use the **-setup_slack_limit** to tell IC Compiler to optimize paths that have at least -1ns slack.

```
prompt> set_primetime_options -setup_slack_limit -1
```

SEE ALSO

clock_opt(2)
place_opt(2)
report_primetime_options(2)
route_opt(2)
signoff_opt(2)

set_propagated_clock

Specifies propagated clock latency.

SYNTAX

```
string set_propagated_clock  
object_list
```

Data Types

object_list list

ARGUMENTS

object_list
Provides a list of clocks, ports, pins, or cells.

DESCRIPTION

Specifies that delays be propagated through the clock network to determine latency at register clock pins. If not specified, ideal clocking is assumed. Ideal clocking means clock networks have a specified latency (from the **set_clock_latency** command), or zero latency by default. Propagated clock latency is used for postlayout, after final clock tree generation. Ideal clock latency provides an estimate of the clock tree for prelayout.

If **set_propagated_clock** is applied to pins or ports, it affects all register clock pins in the transitive fanout of the pins or ports.

To undo **set_propagated_clock**, use **remove_propagated_clock** or **set_clock_latency** to provide an ideal latency.

To see propagated clock attributes on clocks, use **report_clock -skew**.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

This example specifies to use propagated clock latency for all clocks in the design.

```
prompt> set_propagated_clock [all_clocks]
```

SEE ALSO

remove_propagated_clock(2)

```
set_clock_latency(2)
report_clock(2)
set_input_delay(2)
create_clock(2)
current_design(2)
```


set_pulse_clock_cell

Specifies the pulse type for a pulse clock cell with a single input and a single output.

SYNTAX

```
string set_pulse_clock_cell  
-type pulse_type  
object_list
```

Data Types

object_list list

ARGUMENTS

-type *pulse_type*
Specifies the type of pulse clock applied to all library cells in *object_list*. The possible values for *pulse_type* are: *rise_triggered_high_pulse*, *fall_triggered_high_pulse*, *rise_triggered_low_pulse*, *fall_triggered_low_pulse*

object_list
Lists of library cells which has the specified pulse type. It is required that the library cell has only one single input and one single output.

DESCRIPTION

This command gives user control to override the pulse type automatically inferred by the tool. However, it is not able to override the pulse type defined by library format.

To turn on this command functionality, please set variable *timing_pulse_clock_cell_type_auto_inference* to TRUE.

To check if an instance cell has mismatched pulse type defined from its library cell, please use *check_timing_pulse_clock_cell_type* options.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example specifies a pulse type of *fall_triggered_high_pulse* for a library cell.

```
prompt> set_pulse_clock_cell -type fall_triggered_high_pulse {celllib/  
pulse_rise_high}
```

SEE ALSO

`check_timing(2)`

set_qtm_global_parameter

Sets a global parameter for quick timing models (QTM).

SYNTAX

```
string set_qtm_global_parameter  
[-param parameter]  
[-lib_cell lib_cell]  
[-pin pin_name]  
[-clock pin_name]  
[-value parameter_value]
```

Data Types

<i>parameter</i>	string
<i>lib_cell</i>	string
<i>pin_name</i>	string
<i>parameter_value</i>	float

ARGUMENTS

- param *parameter*
Specifies the global parameter you want to set. The global parameter can be setup, hold, or clk_to_output.
- lib_cell *lib_cell*
Specifies the lib_cell you want to use to mimic the global parameter.
- pin *pin_name*
Specifies the related pin for the global parameter if you are using lib_cell to mimic the behavior. If the parameter is setup/hold, this pin must be an input pin. If the parameter is clk_to_output, this pin has to be an output pin. If you do not use this option, QTM uses the first encountered input pin that is constrained, in case of setup/hold; and the first output pin which has an edge delay arc coming into it, in case of clk_to_output parameter.
- clock *pin_name*
Specifies the constraining pin in case of setup or hold; and the launch pin in case of clk_to_output parameter.
- value *parameter_value*
The value of the global parameter in time units.

DESCRIPTION

This command sets the value of a desired QTM global parameter. The global parameter can be either setup, hold, or clk_to_output. These parameters specify the global parameter which is added to the arcs. To the QTM setup arc, the time is added to arrive at the final delay. To the QTM hold arc, the global hold time is added to arrive at the final delay. To the QTM edge triggered delay arc the global clk_to_output time is added to arrive at the final delay.

You can specify the global parameters of the sequential element by specifying a cell in the library (**-lib_cell** option), or by giving the actual value, by using **-value** option. If the **-lib_cell** option is specified, you can specify the clock pin and the input pin name. If you do not specify one, the first setup, hold, or clk_to_output arc encountered is chosen (based on which parameter is being set). If only **-clock** option is specified, the first setup, hold, or clk_to_output arc (depending on the global parameter) from that clock pin is chosen. If only the **-pin** option is specified, the first setup, hold, or clk_to_output arc (depending on the type of global parameter) coming into the pin is chosen.

To show the information about the current QTM model, use **report_qtm_model** command.

For a basic description about QTM, please refer to **create_qtm_model** man page. For a more detailed description about QTM, please refer to Chapter 12 of the *ICC Design Planning User Guide*.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

In the following examples where ever **-lib_cell** is used, you must have loaded the technology library and set the qtm technology set. This is done through the following sequence of commands:

```
read_db my_technology_library.db
```

```
set_qtm_technology -library my_technology_library
```

The following example sets the global setup time equivalent to the setup time of DFF1.

```
prompt> set_qtm_global_parameter -param setup -lib_cell DFF1
```

The following example sets the global hold time equivalent to the setup time of pin D of DFF1.

```
prompt> set_qtm_global_parameter -param setup -lib_cell DFF1 -pin D
```

The following example sets the global clk_to_out time to be 1.5 time units.

```
prompt> set_qtm_global_parameter -param clk_to_output -value 1.5
```

SEE ALSO

```
create_qtm_constraint_arc(2)
create_qtm_delay_arc(2)
create_qtm_drive_type(2)
create_qtm_load_type(2)
create_qtm_model(2)
create_qtm_path_type(2)
report_qtm_model(2)
```

```
set_qtm_global_parameter
2718
```

save_qtm_model(2)

set_qtm_port_drive

Sets the drive on quick timing model (QTM) ports.

SYNTAX

```
string set_qtm_port_drive  
[-type drive_type]  
[-value drive_value]  
[-input_transition_rise rtrans]  
[-input_transition_fall ftrans]  
port_list
```

Data Types

<i>drive_type</i>	string
<i>drive_value</i>	float
<i>rtrans</i>	float
<i>ftrans</i>	float
<i>port_list</i>	list

ARGUMENTS

-type *drive_type*
Specifies the the global *drive_type* parameter with this option.

-value *drive_value*
Specifies the drive value in terms drive resistance units with this option.

-input_transition_rise *rtrans*
Specifies the input rising transition time associated with the **-input_pin** of the specified drive type to compute the delay for the drive arc for this port. If you do not include this option, the delay table for rising input of the drive arc will be loaded from library as is.
Use the **-input_transition_rise** and **-input_transition_fall** options to capture the transition time associated with the *input_pin* defined for the drive type referred. This can obtain more accurate information on the transition time and delay time for the drive arc to be defined for this port.

-input_transition_fall *ftrans*
Specifies the input falling transition time associated with the *input_pin* of the specified drive type to compute the delay for the drive arc for this port. If you do not include this option, the delay table for falling input of the drive arc will be loaded from library as is.

port_list
Specifies a list of input or inout ports on which to set the attribute. Each element in the list is either a collection of QTM ports or a pattern which matches QTM ports.

DESCRIPTION

This command sets the drive of the output port of the QTM model. There are two methods for setting the drive:

1. Use one the global *drive_type*, if you have drive types defined.
2. Define the drive in terms of the drive resistance units.

Define drive types by using the command **create_qtm_drive_type**.

To show the information about the current QTM model, use the **report_qtm_model** command.

For a basic description about QTM, please refer to **create_qtm_model** man page. For a more detailed description about QTM, please refer to the *ICC Design Planning User Guide*.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following command sets the drive of output port OUT1 to be of drive_type drive1.

```
prompt> set_qtm_port_drive -type drive 1 OUT1
```

The following command sets the drive of the output port OUT2 to be 5 drive units.

```
prompt> set_qtm_port_drive -value 5.0 OUT2
```

SEE ALSO

```
create_qtm_load_type(2)  
create_qtm_model(2)  
report_qtm_model(2)  
save_qtm_model(2)  
set_qtm_port_load(2)
```

set_qtm_port_load

Sets the load on quick timing model (QTM) ports.

SYNTAX

```
string set_qtm_port_load  
[-type load_type]  
[-factor multiplication_factor]  
[-value load_value]  
port_list
```

Data Types

<i>load_type</i>	string
<i>multiplication_factor</i>	float
<i>load_value</i>	float
<i>port_list</i>	list

ARGUMENTS

-type load_type
Specifies the global *load_type* parameter.

-factor multiplication_factor
Specifies the multiplication factor for the load type to set the load on the QTM port.

-value load_value
Specifies the load value in the capacitance units.

port_list
Specifies a list of input or inout ports on which to set the attribute. Each element in the list is either a collection of QTM ports or a pattern that matches QTM ports.

DESCRIPTION

This command sets the load of the input port of the QTM model. There are two methods for setting the load:

1. Use the global *load_type* along with a multiplication factor if you have load types defined.
2. Define the load in terms of the capacitance units.

Define load types using the command **create_qtm_load_type**.

To show the information about the current QTM model, use the **report_qtm_model** command.

For a basic description about QTM, please refer to **create_qtm_model** man page. For a more detailed description about QTM, please refer to the *ICC Design Planning User*

Guide.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following command sets the load on IN1 to be 2 times the global load_type load1.

```
prompt> set_qtm_port_load -type load1 -factor 2 IN1
```

The following command sets the load on IN2 to be 4.5 capacitance units.

```
prompt> set_qtm_port_load -value 4.5 IN2
```

SEE ALSO

```
create_qtm_load_type(2)  
create_qtm_model(2)  
report_qtm_model(2)  
save_qtm_model(2)  
set_qtm_port_drive(2)
```

set_qtm_technology

Sets quick timing model technology variables.

SYNTAX

```
string set_qtm_technology  
[-library name]  
[-max_transition max_trans_value]  
[-min_transition min_trans_value]  
[-max_capacitance max_cap_value]  
[-min_capacitance min_cap_value]  
[-wire_load_model wlm_name]  
[-operating_condition opcond_name]  
[-process process_value]  
[-voltage voltage_value]  
[-temperature temperature_value]
```

Data Types

<i>name</i>	string
<i>max_trans_value</i>	float
<i>min_trans_value</i>	float
<i>max_cap_value</i>	float
<i>min_cap_value</i>	float
<i>wlm_name</i>	string
<i>opcond_name</i>	string
<i>process_value</i>	float
<i>voltage_value</i>	float
<i>temperature_value</i>	float

ARGUMENTS

- library *name*
Specifies the library name for using library elements to define global parameters. The library is the timing library in the Synopsys data base (.db) file, not the Milkyway reference or design library. Substitute the string value you want for *name*.
- max_transition *max_trans_value*
Specifies the maximum transition value to use. You must express it in units consistent with the those in the technology library used during optimization. For example, if the library specifies drive values in kilohms and load values in picofarads, then you must express *max_transition_value* in nanoseconds. Substitute the float value you want for *max_trans_value*.
- min_transition *min_trans_value*
Specifies the minimum transition value to use. You must express it in units consistent with the those in the technology library used during optimization. For example, if the library specifies drive values in kilohms and load values in picofarads, then you must express *min_transition_value* in nanoseconds. Substitute the float value you want for *min_trans_value*.

`-max_capacitance max_cap_value`

Specifies the maximum value to use for capacitance. You must express it in units consistent with the technology library used during optimization. For example, if the library specifies capacitance values in picofarads, then you must express *max_cap_value* in picofarads. Substitute the float value you want for *max_cap_value*.

`-min_capacitance min_cap_value`

Specifies the minimum value to use for capacitance. You must express it in units consistent with the technology library used during optimization. For example, if the library specifies capacitance values in picofarads, then you must express *min_cap_value* in picofarads. Substitute the float value you want for *min_cap_value*.

`-wire_load_model wlm_name`

Specifies the wire load model to use when calculating delays. Substitute the string value you want for *wlm_name*.

`-operating_condition opcond_name`

Specifies the default operating condition name for the quick timing model in a multicorner, multimode (MCMM) design. Must be specified together with `-process`, `-voltage`, and `-temperature`.

`-process process_value`

Specifies the process scaling factor for the default operating condition of the quick timing model in an MCMM design. When you want to use any of the following options, you must use all of them: **-operating_condition**, **-process**, **-voltage**, and **-temperature**. Allowed values are 0.0 through 100.0.

`-voltage voltage_value`

Specifies the voltage value, in Volts, for the default operating condition of the quick timing model in an MCMM design. When you want to use any of the following options, you must use all of them: **-operating_condition**, **-process**, **-voltage**, and **-temperature**.

`-temperature temperature_value`

Specifies the temperature value, in degrees Celsius, for the default operating condition of the quick timing model in an MCMM design. When you want to use any of the following options, you must use all of them: **-operating_condition**, **-process**, **-voltage**, and **-temperature**.

DESCRIPTION

This command sets various quick timing model technology parameters. You can use the **-library** option to define path types, drive types, and load types, or if you set the other global parameters in terms of library cells. Before setting the library, you ensure that it is loaded.

If you have set the wire load model, the path type delays are calculated by using that wire load model.

You can use the **report_qtm_model** command to show information about the current quick timing model.

For a basic description of quick timing models, see the **create_qtm_model** man page. For a more detailed description of quick timing models, see *Creating Quick Timing Models for Black Boxes in IC Compiler User Guide: Design Planning Contents*.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example sets the library to the name `my_lib`.

```
prompt> read_db my_lib.db  
#my_lib.db contain the library my_lib  
prompt> set_qtm_technology -library my_lib
```

The following command sets the `wire_load_model` to the name `wlm1`.

```
prompt> set_qtm_technology -wire_load wlm1
```

SEE ALSO

```
create_qtm_model(2)  
report_qtm_model(2)  
save_qtm_model(2)
```

set_rail_options

Specifies setup options for the **analyze_rail** command.

SYNTAX

```
status set_rail_options
[-default]
[-output_dir dir_name]
[-use_pins_as_pads true | false]
[-pad_master_file file_name]
[-pad_instance_file file_name]
[-user_defined_tap_file file_name]
[-packaging_file file_name]
[-vd_threshold value]
[-switching_activity {type file_name [strip_path]}]
[-host machine_name]
[-pr_exec_dir dir_name]
[-pt_exec_dir dir_name]
[-sdc file_name]
[-spef file_name]
[-verilog file_name]
[-upf file_name]
[-analysis_mode static | dynamic]
[-start_time time_value]
[-end_time time_value]
[-config_file file_name]
[-filler_lib_cells lib_cell_list]
[-decap_lib_cells lib_cell_list]
```

Data Types

<i>dir_name</i>	string
<i>file_name</i>	string
<i>value</i>	float
<i>type</i>	string
<i>machine_name</i>	string
<i>time_value</i>	float
<i>lib_cell_list</i>	list

ARGUMENTS

-default

Resets all the options to their default settings for the PrimeRail analysis flow.

-output_dir *dir_name*

Specifies the output directory for the **analyze_rail** command.

By default, the PrimeRail output directory is created under the current working directory and is named *pr_current_cell_name*. The PrimeRail output directory contains the following content:

- The *synopsys_rail_setup* directory

This directory contains data output from IC Compiler for running PrimeRail.

- The rail_map_data directory

This directory contains PrimeRail map data to be viewed in the IC Compiler GUI.

- The analyze_rail.cmd file

This file contains the script of the PrimeRail run.

- The analyze_rail.log file

This file contains the log file of the PrimeRail run.

- The error text report files

These files follow the naming convention of
<current_cell_name>_<net_name>_<analysis_type>_err.txt.

- Other files, such as PrimeRail and PrimeTime PX run scripts and intermediate files

`-use_pins_as_pads true | false`

Controls whether the top-level power and ground pins are treated as ideal voltage sources in the block-level simulation. The default value is true.

`-pad_master_file file_name`

Specifies an optional pad master file name, which specifies the pad master cells to be used as ideal voltage sources for rail analysis.

You can optionally specify a cell to be treated as a pad master only when analyzing certain power or ground nets. If you do not specify the net name, the command assumes that the specified pad master is used for all the nets being analyzed.

The following is the format used to specify master names and net names:

master_name [*net_name*]

For example, to specify a master name of VDD.FRAME for the VDD net, enter
VDD.FRAME VDD

`-pad_instance_file file_name`

Specifies an optional pad instance file name, which specifies the pad cell instances to be used as ideal voltage sources for rail analysis.

You can optionally specify a cell to be treated as a pad instance only when analyzing certain power or ground nets. If you do not specify the net name, the command assumes that the specified pad instance is used for all the nets being analyzed.

The following is the format used to specify instance names and net names:

instance_name [*net_name*]

For example, to specify an instance name of VDD1 for the VDD net, enter
VDD1 VDD

`-user_defined_tap_file file_name`

Specifies an optional file that defines the physical location (coordinates and layer numbers) of the ideal voltage sources to be used in the rail analysis. This is useful when you are working with a block or a design that does not yet have pins defined, or the location or the design of the pad cells is not yet finalized.

This ASCII file specifies the ideal voltage source coordinates and layer numbers in the following format:

netName layerNumber xCoord yCoord

The coordinates are in microns.

For example, if you want to apply the ideal voltage of VDD net on layer 3 at three locations, create a file with the following lines:

VDD 3 784.000 1826.000

VDD 3 1184.000 1826.000

VDD 3 1584.000 1826.000

-packaging_file *file_name*

Specifies an optional SPICE file that contains packaging parasitics to be analyzed together with the chip.

-vd_threshold *value*

Specifies the voltage drop threshold in mV for violation checking.

If you do not specify this option, the tool sets the threshold to 0.0 (the default) and does not generate a voltage drop violation report.

This option also serves as the target voltage threshold for decoupling capacitance analysis. For details, please refer to the decoupling capacitor analysis section in the **analyze_rail** command man page.

-switching_activity {*type file_name* [*strip_path*]}

Specifies an optional switching activity file to be used in rail analysis.

If you do not specify this option, the default toggle rate is used.

The *file_format* argument can be VCD, SAIF, or User-defined.

VCD or SAIF files are often generated by simulation tools that use a different top-level design than the one used in the IC Compiler session, so the optional *strip_path* argument provides a mechanism to remove the portion of the included object names that are not represented in IC Compiler.

For example, to load a VCD file and strip the /top_design/cup_module prefix from all object names, enter

-switching_activity {VCD top_design.vcd /top_design/cup_module}

If you specify a VCD file, the tool performs rail analysis with all time values included in the file.

When the *file_format* argument is set to User-defined, you need to specify a switching information Tcl file for the PrimeTime PX power analysis. The *strip_path* argument is not used and should be set to "". The following are the supported variables and commands for the switching information file:

Variables:

power_default_toggle_rate (default is 0.1)

power_default_static_probability (default is 0.5)

power_default_toggle_rate_reference_clock [fastest|related] (default is related)

Commands:

set_case_analysis 0 reset

set_switching_activity

set_annotated_power

Refer to the PrimeTime-PX documentation for more information.

-host *machine_name*

Specifies a computer host name for running PrimeRail on a different machine.

By default, PrimeRail is run on the same machine on which the current IC Compiler session is running.

`-pr_exec_dir dir_name`
 Specifies the path to the PrimeRail executable.
 By default, IC Compiler locates the executable by using the UNIX search path.

`-pt_exec_dir dir_name`
 Specifies the path to the PrimeTime executable.
 By default, IC Compiler locates the executable by using the UNIX search path.

`-sdc file_name`
 Specifies an optional SDC script file.
 If you do not specify this option, the tool creates an SDC file based on the data saved in the Milkyway design.

`-spef file_name`
 Specifies an optional parasitic SPEF file.
 If you do not specify this option, the tool creates a parasitic SPEF file based on the data saved in the Milkyway design.

`-verilog file_name`
 Specifies an optional Verilog netlist file.
 If you do not specify this option, the tool creates a Verilog netlist file based on the data saved in the Milkyway design.

`-upf file_name`
 Specifies an optional UPF script file.
 If you do not specify this option, the tool creates a UPF script file based on the data saved in the Milkyway design.

`-analysis_mode static | dynamic`
 Specifies to run either static or dynamic rail analysis through the **analyze_rail** command. In PrimeRail, static rail analysis is also called time-average rail analysis, and dynamic rail analysis is also called transient rail analysis. The default mode is static.
 In static voltage drop and electromigration analysis, the **analyze_rail** command generates the following information:

- o Average voltage drop value in the map file
- o Average electromigration value in the map file
- o Average voltage drop error cell
- o Average electromigration error cell

In dynamic voltage drop and electromigration analysis, the **analyze_rail** command generates the following information:

- o Average and peak voltage drop values in the map file
- o Average, peak and root mean square (RMS) electromigration values in the map file
- o Peak voltage drop error cell
- o Average, peak and RMS electromigration error cells

`-start_time time_value`
 Specifies the start time for a simulation run on a specific subwindow of activity, rather than on the entire simulation time. The start and end time define the time segment. This time segment should be a subset of the current waveform generation time window.
 The *time_value* is in seconds.
 This is an optional argument and is for dynamic rail analysis only. When no time value is given, PrimeRail will automatically determine an appropriate

value.

-end_time *time_value*

Specifies the end time for a simulation run on a specific sub-window of activity, rather than on the entire simulation time. The start and end time define the time segment. This time segment should be a subset of the current waveform generation time window.

The *time_value* is in seconds.

This is an optional argument and is for dynamic rail analysis only. When no time value is given, PrimeRail will automatically determine an appropriate value.

-config_file *file_name*

You can specify a configuration file for PrimeRail. When specified, the IC Compiler-generated PrimeRail script called by the **analyze_rail** command will copy the settings in this configuration file into PrimeRail early in the script processing. This will allow you to have PrimeRail-specific setup for a design without having to use the *-script_only* option.

Note this configuration file only allows define commands in the Scheme syntax, that are used to define files and variables. A generic Scheme script is not allowed.

The following 2 variables are used to specify total steps and step size for the dynamic rail analysis. The total steps variable defines the number of time steps the simulation is run. The step size variable defines the size of the time step used during simulation. The default is one-third of the average transition time in the circuit. All the settings are optional. If a time step and the number of steps are specified, it takes priority over the length of the time window given.

o define TRANSIENT_TOTAL_STEPS *integer*

o define TRANSIENT_STEP_SIZE *float*

The first 3 variables in the following variable list will be used by PrimeRail to read in files that annotate user-defined instance power, instance effective resistance and capacitance, and instance current waveforms, respectively. The 4th variable is used to turn off or on pattern match for port instance names.

o define PORT_INST_POWER *file_name*

o define PORT_INST_RC_EFFECTIVE *file_name*

o define PORT_INST_CURRENT_WAVEFORM *file_name*

define PORT_INST_PATTERN_MATCH No|Yes

The following are the commands supported in the instance power information file:

o defineCellInstancePower *cellId cellInstName value*

o defineCellInstancePowerByPort *cellId cellInstName portName value*

The following are the commands supported in the effective resistance and capacitance file:

o definePortInstanceRCEffective *cellId cellInstName portName R_value C_value*

o definePortInstanceRCEffectiveByMaster *cellId cellInstMasterName portName R_value C_value*

The following is the command supported in the instance current waveform file where current waveforms are described in the piece-wise linear (PWL) format:

o definePortInstanceCurrentWaveform *cellId cellInstName portName numPoints '(t1 i1 t2 i2...)*

Refer to the **poCalculatePower** command in the PrimeRail documentation for m

ore information.

The following variable is used to specify the switch cell power-up event (or sequence) file for the in-rush analysis.

o define SWITCH_CELL_EVENT_FILE *file_name*

The following variable is the percentage of supply voltages considered as the stable controlled voltage level after power-up in in-rush analysis. The default value is 0.01.

o define WAKEUP_THRESHOLD *float*

-filler_lib_cells *lib_cell_list*

Specifies the names of the filler cells.

Filler cell names are required for the decoupling capacitor cell replacement. There is no default value. An error will be issued through the **analyze_rail** command if no filler cell names are available during decoupling capacitor analysis.

Specifies the decoupling capacitor cell names for insertion.

There is no default value. An error will be issued through the **analyze_rail** command if no decoupling capacitor cell names are available during decoupling capacitor analysis.

DESCRIPTION

This command specifies options for executing the **analyze_rail** command. To run the **analyze_rail** command with nondefault settings, you must run this command before running the **analyze_rail** command.

The following sets of options are valid for full-chip analysis:

- None of the following options: *-sdc*, *-spef*, *-verilog*, and *-upf*
- The *-sdc* option and the *-spef* option, along with zero or more of the *-verilog* option and the *-upf* option.

If you use other combinations of these options, the **analyze_rail** command stops and returns an error.

EXAMPLE

The following example resets all the options to their default settings:

```
prompt> set_rail_options -default
```

The following example sets ideal voltage sources to the user-defined taps:

```
prompt> set_rail_options -user_defined_tap vdd_tap.txt
```

The following example sets the switching activity file to the user-defined file called *ptpx_switching_info*:

```
prompt> set_rail_options -switching_activity {User-defined ptpx_switching_info ""}
```

SEE ALSO

`report_rail_options(2)`
`analyze_rail(2)`

set_register_type

Sets the **latch_type** or **flip_flop_type** attributes on designs or cell instances, to specify which sequential cells from the target library are to be used by the **compile** command.

SYNTAX

```
int set_register_type
-latch example_latch -exact |
-flip_flop example_flip_flop [-exact]
[cell_or_design_list]
```

Data Types

<i>example_latch</i>	string
<i>example_flip_flop</i>	string
<i>cell_or_design_list</i>	list

ARGUMENTS

-latch *example_latch*
Specifies a latch to be used by the **compile** command as the default latch type. You can specify both **-latch** and **-flip-flop**, but you must specify at least one.

-exact
Instructs the **compile** command to make an exact mapping to the specified *example_latch* or *example_flip_flop*, if possible. This argument is required when using the **-latch** argument.

-flip_flop *example_flip_flop*
Specifies a flip-flop from the target library to be used by the **compile** command as the default flip-flop type. the *example_flip_flop*, if possible. This argument sets the **default_flip_flop_type** or **default_flip_flop_type_exact** attribute to the *example_flip_flop* on all designs in *cell_or_design_list*; and the **flip_flop_type** or **flip_flop_type_exact** attribute to the *example_flip_flop* on all cells in *cell_or_design_list*. You can specify both **-latch** and **-flip-flop**, but you must specify at least one.

cell_or_design_list
Specifies a list of cells or designs in which to use the latch or flip-flop. The default is the current design.

DESCRIPTION

Specifies latch or flip-flop type information for the **compile** command to use, by setting appropriate attributes on the designs or cell instances. For designs, specifying **-flip_flop** without **-exact**, sets the **default_flip_flop_type** attribute to *example_flip_flop*, indicating to use it as the default flip-flop type for those designs. And specifying **-exact -flip_flop** sets the **default_flip_flop_type_exact** attribute to *example_flip_flop*, indicating to use it as the exact default flip-flop

for those designs.

For example, to set the default flip-flop type for a design to be a D flip-flop, specify the name of any D flip-flop in the target library as the *example_flip_flop*, without **-exact**. If you specify **-exact**, the *example_flip_flop* is interpreted as the exact flip-flop to use.

Similarly, for cell instances, specifying **-flip_flop** without **-exact** sets the **flip_flop_type** attribute to *example_flip_flop*, indicating that it is to be used as the specific flip-flop type for those cells. And specifying **-exact -flip_flop** sets the **flip_flop_type_exact** attribute to *example_flip_flop*, indicating that it is to be used as the exact flip-flop for those cells.

When specifying a latch, you must always use the **-exact** option, so the *example_latch* is always the exact latch used as the default latch for designs and as the specific latch for cells. The **default_latch_type_exact** attribute is set to the specified latch on designs, and the **latch_type_exact** attribute is set to the specified latch on cells.

There are no **latch_type** or **default_latch_type** attributes.

The mapping process for flip-flops and latches consists of two steps:

1. Sequential gates are mapped to an initial latch or flip-flop from the target library. The **compile** command attempts to convert flip-flops or latches tagged by **set_register_type** to the specified flip-flop or latch type. Untagged sequential components are mapped to the default latch or flip-flop type if specified. In the absence of attributes on a design, or a sequential cell instance to control the mapping, the smallest area-cost flip-flop or latch is chosen.
2. If a flip-flop is not specified with the **-exact** option, the tool attempts to individually remap it to a lower-cost component from the target library. If **compile** cannot use the sequential component specified, or if no default flip-flop or latch is specified, **compile** maps these cells into the smallest sequential component possible.

A cell instance can have either latch or flip-flop attributes, but not both. Only latch cells can have latch attributes, and only flip-flop cells can have flip-flop attributes. Designs can have both latch and flip-flop attributes, because these attributes designate the default latch type (for latches in the design) and the default flip-flop type (for flip-flops in the design).

Unmapped sequential components can result when the **translate** command cannot find a close match for a flip-flop or latch in the new target library, or when latches or flip-flops are added to a design by HDL Compiler, VHDL Compiler, or state-machine software.

The **set_register_type** command constrains the mapping process. It does not constrain scan replacement in **insert_scan** or in **compile**.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets the default flip-flop type for the current design to a D flip-flop, and that one example of a D flip-flop in the current library is the component *DFLOP*. This sets the **default_flip_flop_type** attribute to *DFLOP* on the current design. As a result, the default flip-flop used in mapping components that are not otherwise tagged is a D flip-flop.

```
prompt> set_register_type -flip_flop DFLOP
```

The following example sets the default flip-flop type for the current design exactly to the *DFLOP* cell. This sets the **default_flip_flop_type_exact** attribute to *DFLOP* on the current design. As a result, the default flip-flop used in mapping components that are not otherwise tagged is *DFLOP*.

```
prompt> set_register_type -exact -flip_flop DFLOP
```

The following example sets the default latch type for *FOO* and *BAR* designs to exactly a *DLATCH* gate. The **compile** command attempts to map all unmapped latches in these designs to *DLATCH*. This sets the **default_latch_type_exact** attribute to *DLATCH* on the *FOO* and *BAR* designs. The presence of this attribute indicates that an exact mapping is to be attempted for the *DLATCH* latch on the untagged cells in the *FOO* and *BAR* designs. Sequential types for latches must always be set as exact.

```
prompt> set_register_type -exact -latch DLATCH {FOO, BAR}
```

The following example maps a group of cell instances exactly to the *DLATCH* latch. This sets the **latch_type_exact** attribute on the *U1*, *U2*, and *U3* cell instances to the name *DLATCH*.

```
prompt> set_register_type -exact -latch DLATCH {U1, U2, U3}
```

The following example specifies the exact default latch type for the current design as *DLATCH* and the exact default flip-flop type as *DFLOP*. This sets the **default_latch_type_exact** attribute to the name *DLATCH* and the **default_flip_flop_type_exact** attribute to the name *DFLOP* on the current design so that **compile** can map otherwise untagged components in the design. You must use **-exact** when you specify both **-latch** and **-flip_flop** in the same command.

```
prompt> set_register_type -exact -latch DLATCH -flip_flop DFLOP
```

SEE ALSO

```
current_design(2)  
get_attribute(2)  
remove_attribute(2)  
reset_design(2)  
target_library(3)
```

set_related_supply_net

Associates an external supply net to the port of the design.

SYNTAX

```
status set_related_supply_net  
[supply_net_name]  
[-object_list objects]  
[-reset]  
[-ground ground_net_name]  
[-power power_net_name]
```

Data Types

<i>supply_net_name</i>	string
<i>objects</i>	list
<i>ground_net_name</i>	string
<i>power_net_name</i>	string

ARGUMENTS

supply_net_name
Specifies the name of the power net. This is deprecated and supported for backward compatibility only. New scripts should specify the power net using -power option.

-object_list *objects*
Specifies the list of ports to be associated with power/ground nets or reset. If this option is not specified, all the ports are considered.

-reset
Resets the related power and ground nets of the ports specified in object_list. This can't be used in conjunction with -power or -ground options

-ground *ground_net_name*
Specifies the name of the ground net.

-power *power_net_name*
Specifies the name of the power net.

DESCRIPTION

This command is used to associate supply nets with design ports. The tool uses this information for constraining and checking design. The level shifter insertion tool also uses supply nets to determine if level shifter is required between a port and the logic connected to the port.

If this command is not specified, the tool assumes that ports are externally connected to the primary supply net of the domain of the top design.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets the *VDD_EXT* power net and *VSS_EXTfp ground net with INPUT* port.

```
prompt> set_related_supply_net -object_list\  
[get_ports INPUT] -power VDD_EXT -ground VSS_EXT
```

This example resets the power net and ground net on all ports.

```
prompt> set_related_supply_net -reset
```

SEE ALSO

`get_ports(2)`

set_relative_always_on

Sets the relative always-on relationship between power domains.

SYNTAX

```
status set_relative_always_on  
domain_name  
[-relative_to list_of_domains]
```

Data Types

```
domain_name           string  
list_of_domains      list
```

ARGUMENTS

```
domain_name  
    Specifies the name of a relatively-more always-on power domain.  
  
-relative_to list_of_domains  
    Specifies 1 or more power domains that are relatively-less always-on with  
    respect to the domain specified using domain_name.
```

DESCRIPTION

This command creates relative always-on relationships between domains. The tool extracts power sequencing behavior between various domains based on the relationship specified by using this command. The information is used for checking isolation requirements on nets crossing domains.

For an example of how the tool uses a relative always-on relationship between domains consider the following scenario:

If domain A is more always-on than domain B and if an isolation cell on a net drives domain B from A, then the tool can flag this isolation cell as redundant.

The isolation cell is flagged as redundant because domain A will be on when domain B is on, so no isolation is required. Conversely, if there is no isolation cell connected to a net that drives domain A from B, then the tool can flag this as an error condition. The error results because when domain B is off, domain A is on, which might create a floating net condition.

EXAMPLES

The following example sets the PD1 power domain to be more always-on with respect to power domains PD2 and PD3:

```
prompt> set_relative_always_on PD1 B-relative_to {PD2 PD3}  
1
```

SEE ALSO

`create_power_domain(2)`
`remove_power_domain(2)`
`report_power_domain(2)`

set_resistance

Sets the resistance value on nets.

SYNTAX

```
int set_resistance  
value  
[-min] [-max] net_list
```

Data Types

<i>value</i>	float
<i>net_list</i>	list

ARGUMENTS

value

Specifies the resistance value for nets in *net_list*. *value* must be expressed in unit system consistent with what the technology library used during optimization. For example, if the technology library specifies resistance values in ohms, *value* must also be expressed in ohms.

-min

Indicates that the resistance value is to be used for minimum delay analysis. If you do not specify a value for minimum delay analysis, the maximum value is used. If you do not specify a value for maximum delay analysis, the minimum value is ignored. (You cannot annotate only a minimum value.)

-max

Indicates that the resistance value is to be used for maximum delay analysis. If you do not specify **-min** or **-max**, the value is used for both minimum and maximum delay analysis.

net_list

Specifies a list of nets for which the specified resistance values are to be set.

DESCRIPTION

Sets the **ba_net_resistance** attribute, which enables the back-annotation of resistance values on nets in the current design. If the current design is hierarchical, it must have been linked with the **link** command. The specified *value* overrides the internally-estimated net resistance value.

You also can use the **set_resistance** command for nets at lower levels of the design hierarchy. These nets are specified as "BLOCK1/BLOCK2/NET_NAME."

To view resistance values, use the **report_net** command.

To remove a resistance value, use **remove_attribute**. To reset all back-annotated resistance values in a design, use the **reset_design** command.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example sets a resistance of 200 units to nets "a" and "b".

```
prompt> set_resistance 200 {a b}
```

This example sets a resistance of 300 units on net "U1/U2/NET3."

```
prompt> set_resistance 300 U1/U2/NET3
```

This example removes the back-annotated resistance on net "U1/U2/NET3".

```
prompt> remove_attribute [get_nets U1/U2/NET3] ba_net_resistance
```

SEE ALSO

```
current_design(2)  
link(2)  
remove_attribute(2)  
report_net(2)  
report_timing(2)  
reset_design(2)  
set_drive(2)  
target_library(3)
```

set_retention

Defines the UPF retention strategy for the power domains in the design. This command is supported only in UPF mode.

SYNTAX

```
status set_retention  
retention_strategy  
-domain power_domain  
-retention_power_net retention_power_net  
-retention_ground_net retention_ground_net  
[-elements objects]
```

Data Types

<i>retention_strategy</i>	string
<i>power_domain</i>	string
<i>retention_power_net</i>	string
<i>retention_ground_net</i>	string
<i>objects</i>	list

ARGUMENTS

retention_strategy
Specifies the UPF retention strategy name. The retention strategy name should be unique within the specified power domain.

-domain *power_domain*
Specifies the name of the power domain to which to apply this UPF retention strategy.

-retention_power_net *retention_power_net*
Specifies the retention power net for the retention cells that will be under this UPF retention strategy.

-retention_ground_net *retention_ground_net*
Specifies the retention ground net for the retention cells that will be under this UPF retention strategy.

-elements *objects*
Specifies the objects to which this UPF retention strategy applies. The objects can be hierarchical cells, leaf cells, HDL blocks, and nets.

DESCRIPTION

This command defines the UPF retention strategy on the specified power domain under which to map the unmapped sequential cells to retention cells.

If the **-elements** option is not specified, then the retention strategy will be applied to all of the unmapped sequential cells under the power domain. If **-elements** is specified, the UPF retention strategy will be applied to all of the unmapped

sequential cells that are under the objects from **-elements**.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows how to define a UPF retention strategy in the specified power domain PD1. Power domain PD1 is defined on instance shutdown_inst.

```
prompt> set_retention retention_1\  
-domain PD1\  
-retention_power_net PN1 B\  
-retention_ground_net GN1
```

The following example shows how to define a UPF retention strategy in the objects under the specified power domain:

```
prompt> set_retention retention_2\  
-domain PD1\  
-retention_power_net PN1\  
-retention_ground_net GN1\  
-elements shutdown_inst/mid_inst
```

SEE ALSO

map_retention_cell(2)
set_retention_control(2)

set_retention_cell

On-The-Fly setting of library retention cell.

SYNTAX

```
Boolean set_retention_cell  
cell_name  
[-cell_type retention_type]  
[-retention_pin {pin_name pin_type disable_value}]  
[-reset]
```

Data Types

<i>cell_name</i>	string
<i>pin_name</i>	string
<i>pin_type</i>	string [restore save save_restore]
<i>disable_value</i>	Boolean [0 1]

ARGUMENTS

cell_name
Specify the library cell as isolation cell.

-cell_type *retention_type*
Specify the retention cell type.

-retention_pin {*pin_name pin_type disable_value*}
Specify the library cell retention pin, the specification should be in {*pin_name pint_type disable_value*} format.

-reset
reset the retention cell attributes.

DESCRIPTION

The **set_retention_cell** command set all the cells with the same name specified as retention cell. Retention pin option can be specified for the retention cell.

If -reset is issued, the retention cell attributes will be cleaned up.

If -reset is combined with other options, reset happens first, other options will be set on the cell after reset.

EXAMPLES

```
prompt> set_retention_cell *DRFF* -cell_type DRFF -
```

```
retention_pin {RETN save_restore 1}
```

SEE ALSO

```
report_lib(2)  
check_library(2)
```


set_retention_control

Defines the UPF retention control signals for the defined UPF retention strategy. This command is supported only in UPF mode.

SYNTAX

```
status set_retention_control
retention_strategy
-domain power_domain
-save_signal {save_signal save_sense}
-restore_signal {restore_signal restore_sense}
```

Data Types

<i>retention_strategy</i>	string
<i>power_domain</i>	string
<i>save_signal</i>	string
<i>save_sense</i>	high low
<i>restore_signal</i>	string
<i>restore_sense</i>	high low

ARGUMENTS

```
retention_strategy
    Specifies the UPF retention strategy name. The retention strategy should have
    already been defined using set_retention command

-domain power_domain
    Specifies the power domain name to which this UPF retention strategy is
    applied.

-save_signal {save_signal save_sense}
    Specifies the save signal and the save signal sense for the retention cells
    under this retention strategy.

-restore_signal {restore_signal restore_sense}
    Specifies the restore signal and the restore sense for the retention cells
    under this retention strategy.
```

DESCRIPTION

This command defines the UPF retention control signals and control signal sense for this retention strategy of this power domain. The specified control signals are applied to the retention cells under the associated retention strategy.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows how to define the retention control signals on the defined UPF retention strategy *retention_1* of the power domain *PD1*.

```
prompt> set_retention_control retention_1 \  
-domain PD1 \  
-save_signal {save_net high} \  
-restore_signal {restore_net low}
```

SEE ALSO

```
map_retention_cell(2)  
set_retention(2)
```

set_route_flip_chip_options

Sets flip-chip router options.

SYNTAX

```
status set_route_flip_chip_options
[-number_of_loops 1 | 2 | 3]
[-route_with_soft_macros true | false]
[-search_effort easy | normal | hard]
[-layer_spacing {layer_spacing_pairs}]
[-layer_width {layer_width_pairs}]
[-rule_name name]
[-nets | -nets_in_file nets_file]
[-output_unrouted_nets open_net_file]
[-
design_style area_IO | peripheral_without_corner_driver | peripheral_with_corner_driver]
[-min_access_edge_length length]
```

Data Types

<i>layer_spacing_pairs</i>	string float
<i>layer_width_pairs</i>	string float
<i>name</i>	string
<i>collection_of_nets</i>	collection
<i>nets_file</i>	string
<i>open_net_file</i>	string
<i>length</i>	float

DATA TYPES

<i>layer_spacing_pairs</i>	string float
<i>layer_width_pairs</i>	string float
<i>name</i>	string
<i>collection_of_nets</i>	collection
<i>nets_file</i>	string
<i>layer</i>	string
<i>open_net_file</i>	string
<i>length</i>	float

ARGUMENTS

-number_of_loops 1 | 2 | 3
Specifies the number of loops to route all the target nets. The default is 1.

-route_with_soft_macros true | false
Controls whether CEL view macro cells are treated as soft macros, which means that the cell area is blocked, except for pin access. When true, you do not have to convert the CEL view to a FRAM view before routing.
If you have a CEL view from custom editing, you might need to turn this option off so the IC Compiler shape-based router can read the internal blockages of

the macro cell or create a FRAM view for this custom-created macro. In any case, you can choose to create FRAM views for macros and use them in the design for ultimate control of the blockages.
The default is false.

`-search_effort` *easy | normal | hard*

Controls the router's search effort.

Use medium effort (the default), unless your design is complex or congested. Setting the search effort to high makes the router take longer to find a path. Set the effort to a higher level if you want the router to find a path for connecting nets. When performing an engineering change order, you might want to consider a higher level for routing the few remaining nets in a large design.

The default is medium.

`-layer_spacing` *{layer_spacing_pairs}*

Specifies the minimum spacing rules for each metal layer. The format is

{m1 spacing1 m2 spacing2 m3 spacing3 ...}

By default, the router uses the minimum spacing specified in the technology file.

`-layer_width` *{layer_width_pairs}*

Specifies the minimum width rules for each metal layer. The format is

{m1 width1 m2 width2 m3 width3 ...}

By default, the router uses the minimum width specified in the technology file.

`-rule_name` *name*

Specifies the rule to use for metal spacing and width. This option is required when you use the **-layer_spacing** or **-layer_width** options.

`-nets` *{collection_of_nets}*

Specifies the nets to be routed.

By default, all flip-chip nets are routed.

This option and the **-nets_in_file** option are mutually exclusive.

`-nets_in_file` *nets_file*

Specifies the name of a file that contains the list of nets to be routed.

This option and the **-nets** option are mutually exclusive.

`-output_unrouted_nets` *open_net_file*

Specifies the name of a file to which to write the unrouted nets.

By default, the unrouted nets are not output.

`-design_style` *area_IO | peripheral_without_corner_driver | peripheral_with_corner_driver*

Specifies that design style.

The default is **peripheral_without_corner_driver**.

`-min_access_edge_length` *length*

Honor Minimum Access Length Rule: the bump-accessing & driver-accessing wires have min-length to follow.

The default value is 0 um.

`set_route_flip_chip_options`

DESCRIPTION

This command sets the options for flip-chip routing. It does not actually invoke any routing. It only allows you to specify options that are stored in the design database for use by subsequent flip-chip router steps.

EXAMPLES

The following example sets options to prepare for the flip-chip routing.

```
prompt> set_route_flip_chip_options \  
-number_of_loops 1 \  
-route_with_soft_macros false \  
-search_effort high \  
-layer_spacing {m8 2.5 m9 5} \  
-layer_width {m8 5 m9 5} \  
-rule_name rdlRule \  
-nets_in_file RDLNetFile \  
-routing_layer m9 \  
-output_unrouted_nets openNetFile \  
-design_style peripheral_with_corner_driver
```

SEE ALSO

```
write_flip_chip_nets(2)  
route_flip_chip(2)  
create_stack_via_on_pad_pin(2)  
remove_flip_chip_route(2)  
push_flip_chip_route(2)  
display_flip_chip_route_flylines(2)  
optimize_flip_chip_route(2)
```

set_route_mode_options

To set the mode for running zroute.

SYNTAX

```
status set_route_mode_options  
[-zroute true | false]
```

ARGUMENTS

```
-zroute true | false  
    Specifies whether to run zroute.  
    The default is false.
```

DESCRIPTION

The **set_route_mode_options** command sets the mode to run zroute.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets the **-zroute** option to true.

```
prompt> set_route_mode_options -zroute true
```

SEE ALSO

set_route_opt_strategy

Sets the nonpersistent options that influence route_opt.

SYNTAX

```
status set_route_opt_strategy
[-default]
[-fix_hold_mode all | route_base]
[-xtalk_reduction_loops loop_count]
[-route_run_time_limit run_time]
[-search_repair_loops detail_loop_count]
[-optimize_wire_via_search_repair_loops optimize_loop_count]
[-route_drc_threshold threshold]
[-eco_route_search_repair_loops eco_routing_loop_count]
```

Data Types

<i>loop_count</i>	integer
<i>run_time</i>	integer
<i>detail_loop_count</i>	integer
<i>optimize_loop_count</i>	integer
<i>threshold</i>	integer
<i>eco_routing_loop_count</i>	integer

ARGUMENTS

- default**
Resets all route_opt strategies to the default value.
- fix_hold_mode all | route_base**
Specifies to hold fix flow inside route_opt. Inside route_opt are three different stages to fix hold: pre-route, global route, and detail route. The **all** argument specifies all mode, in which route_opt runs all three stages of hold fix optimizations. The **route_base** argument specifies **route_base** mode, in which route_opt runs only global route and detail route base hold fix optimization. The default is route_base.
- xtalk_reduction_loops loop_count**
Specifies the number of crosstalk reduction optimization iterations. When the **route_opt** command is run with the **-effort high** option, then, by default the **set_route_opt_strategy** command uses **3**. When the **route_opt** command is run with the **-effort medium** option, then, by default the **set_route_opt_strategy** command uses **2**.
- route_run_time_limit run_time**
Specifies the runtime limitation in minutes on optimize_wire_via routing. The default is no limit. This option cannot be used with set_route_opt_strategy in Zroute mode.
- search_repair_loops detail_loop_count**
Specifies the number of search and repair loops in detail routing. If using zroute, it overrides the -max_number_iterations droute option during initial

route. The default is **15** for ICC router and **10** for Zroute.

`-optimize_wire_via_search_repair_loops optimize_loop_count`

Specifies the number of search and repair loops in optimizing wire via. The valid values are in a range of **0** to **500**. The default value is **5**. This option cannot be used with `set_route_opt_strategy` in Zroute mode.

`-route_drc_threshold threshold`

Specifies the number of the route violation threshold for the tool to use to decide that the design has too many violations. If the design has too many route violations, the tool spends too much time on the search and repair stage. So, if the tool sees the design has too many violations, it optimizes the runtime for search and repair to reduce runtime impact due to route violation. If you specify **-1** for the *threshold* argument, the tool performs no checking and runs a full search and repair loop. The default value is **3000**.

`-eco_route_search_repair_loops eco_routing_loop_count`

Specifies the number of search and repair loops in engineering change order (ECO) routing. If using zroute, it overrides the `-max_number_iterations` droute option during eco route. The default value is **5** for ICC router and **5** for Zroute.

DESCRIPTION

This command specifies the parameters that influence `route_opt`. The parameters are used for further tuning in `route_opt` and are design dependent. For detailed information on parameters, please see the user guide for the tool.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets the detail routing search and repair loop to **20**.

```
prompt> set_route_opt_strategy -search_repair_loops 20
```

The following two examples both set the detail routing search and repair loop to **20** and the eco routing search and repair loop to **10**.

```
prompt> set_route_opt_strategy -search_repair_loops 20\  
-eco_route_search_repair_loops effort 10
```

or

```
prompt> set_route_opt_strategy -eco_route_search_repair_loops 10  
prompt> set_route_opt_strategy -search_repair_loops 20
```


SEE ALSO

`optimize_wire_via(2)`
`report_route_opt_strategy(2)`
`route_opt(2)`

set_route_options

Sets specific options in the internal router control database.

SYNTAX

```
status set_route_options
[-default]
[-groute_timing_driven true | false]
[-groute_timing_driven_weight number]
[-groute_skew_control true | false]
[-groute_skew_weight number]
[-groute_congestion_weight number]
[-groute_clock_routing normal | comb | balanced]
[-groute_incremental true | false]
[-track_assign_timing_driven true | false]
[-track_assign_timing_driven_weight number]
[-droute_connect_tie_off true | false]
[-droute_connect_open_nets true | false]
[-droute_reroute_user_wires true | false]
[-droute_CTS_nets normal | minor_change_only]
[-droute_single_row_column_via_array center | optimize]
[-droute_stack_via_less_than_min_area forbid | add_metal_stub]
[-droute_stack_via_less_than_min_area_cost number]
[-poly_pin_access auto | off]
[-drc_distance diagonal | manhattan]
[-same_net_notch ignore | check_and_fix]
[-fat_wire_check quick | merge_then_check]
[-merge_fat_wire_on preroute_only | preroute_signal | preroute_signal_blockage]
[-fat_blockage_as thin_wire | fat_wire]
[-wire_contact_eol_rule ignore | check_and_fix]
[-enable_user_enter_sub_route_type true | false]
```

Data Types

number integer

DATA TYPES

number integer

ARGUMENTS

- default
Sets the default options.
- groute_timing_driven true | false
Controls whether global routing is timing driven.
The default default is false.
- groute_timing_driven_weight *number*
Specifies the global routing timing weight. You must specify a value between

set_route_options

1 and 7.

The default value is 4.

`-groute_skew_control true | false`

Controls whether global routing skew control is enabled.

The default is false.

`-groute_skew_weight number`

Specifies the global routing skew weight. You must specify a value between 0 and 10.

The default is 5.

`-groute_congestion_weight number`

Specifies the global routing congestion weight. You must specify a value between 1 and 12.

The default is 4.

`-groute_clock_routing normal | comb | balanced`

Specifies the global routing clock topology.

The default is balanced.

`-groute_incremental true | false`

Controls whether incremental global routing is enabled.

The default is false.

`-track_assign_timing_driven true | false`

Controls whether track assignment is timing driven.

The default is false.

`-track_assign_timing_driven_weight number`

Specifies the track assignment timing weight. You must specify a value between 1 and 10.

The default is 1.

`-droute_connect_tie_off true | false`

Controls whether the detail router connects tie-off nets.

The default is true.

`-droute_connect_open_nets true | false`

Controls whether the detail router connects open nets.

The default is true.

`-droute_reroute_user_wires true | false`

Controls whether the detail router reroutes user wires.

The default is false.

`-droute_CTS_nets normal | minor_change_only`

Control the amount of change the detail router makes to clock nets.

The default is minor_change_only.

`-droute_single_row_column_via_array center | optimize`

Controls how the detail router handles single-row via arrays.

The default is center.

-droute_stack_via_less_than_min_area forbid | add_metal_stub
 Controls how the detail router handles stacked vias with less than the minimum area.
 The default is add_metal_stub.

-droute_stack_via_less_than_min_area_cost *number*
 Specifies the cost of having stacked vias with less than the minimum area.
 You must specify a value between 0 and 10.
 The default is 0.

-poly_pin_access auto | off
 Controls poly pin access.
 The default is auto.

-drc_distance diagonal | manhattan
 Controls whether to calculate DRC distances using the diagonal or manhattan method.
 The default is diagonal.

-same_net_notch ignore | check_and_fix
 Controls whether the router checks same net notch rule.
 The default is ignore.

-fat_wire_check quick | merge_then_check
 Controls how the router checks fat wires.
 The default is merge_then_check.

-merge_fat_wire_on preroute_only | preroute_signal | preroute_signal_blockage
 Controls when fat wires are merged.
 The default is preroute_signal.

-fat_blockage_as thin_wire | fat_wire
 Controls how the router treats fat blockages.
 The default is fat_wire.

-wire_contact_eol_rule ignore | check_and_fix
 Controls whether the router checks the end-of-line rule.
 The default is ignore.

-enable_user_enter_sub_route_type true | false
 Controls whether the subroute type attribute is enabled. When this attribute is enabled, you can access its value.
 The default is false.

DESCRIPTION

This command does not actually invoke any routing. It only allows you to specify options that are stored in a database for use by subsequent router steps.

Many of the options that can be set by using the **set_route_options** command can also be set by the **set_droute_options** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets the clock routing topology to balanced.

```
prompt> set_route_options -groute_clock_topology balanced
```

SEE ALSO

`set_droute_options(2)`
`route_detail(2)`

set_route_type

Sets the route type of a group of objects.

SYNTAX

```
status set_route_type
[-signal detail_route | user]
[-clock ring | strap | tie_off | user]
[-pg ring | strap | tie_off | user | std_cell_pin_conn | macro/IO_pin_conn] objects
objects
```

ARGUMENTS

-signal detail_route | user
Sets the object type of signal wires/vias/paths as detail_route or user.

-clock ring | strap | tie_off | user
Sets the object type of clock wires/vias/paths as ring | strap | tie_off | user. The tie_off type is actually marked as detail_route.

-pg ring | strap | tie_off | user | std_cell_pin_conn | macro/IO_pin_conn *objects*
Sets the object type of power/ground wires/vias/paths as ring | strap | tie_off | user | std_cell_pin_conn | macro/IO_pin_conn. The tie_off type is actually marked as detail_route.

DESCRIPTION

This command allows you to set the route type of a group of objects. The objects must be provided with one of the options.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

```
prompt> set_route_type -signal detail_route\  
[get_net_shapes -of_objects net_names]
```

SEE ALSO

get_net_shapes(2)

set_route_zrt_common_options

Sets the options common to all phases of Zroute routing.

SYNTAX

```
status set_route_zrt_common_options
[-enforce_voltage_areas off | strict | relaxed]
[-reroute_clock_shapes true | false]
[-reroute_user_shapes true | false]
[-plan_group_aware off | all_routing | top_level_routing_only]
[-child_process_net_threshold int]
[-verbose_level int]
[-read_user_metal_blockage_layer true | false]
[-wide_macro_pin_as_fat_wire true | false]
[-route_top_boundary_mode stay_half_min_space_inside | stay_inside | ignore]
[-standard_cell_blockage_as_thin true | false]
[-freeze_layer {{layer true | false}...}]
[-freeze_via_to_frozen_layer true | false]
[-max_layer_mode soft | allow_pin_connection | hard]
[-min_layer_mode soft | allow_pin_connection | hard]
[-extra_preferred_direction_wire_cost_multiplier {{layer multiplier}...}]
[-extra_nonpreferred_direction_wire_cost_multiplier {{layer multiplier}...}]
[-extra_via_cost_multiplier {{layer multiplier}...}]
[-number_of_vias_over_max_layer int]
[-number_of_vias_under_min_layer int]
[-via_array_mode off | swap | rotate | all]
[-post_detail_route_redundant_via_insertion off | low | medium | high]
[-concurrent_redundant_via_mode off | reserve_space | insert_at_high_cost]
[-concurrent_redundant_via_effort_level low | medium | high]]
[-eco_route_concurrent_redundant_via_mode off | reserve_space]
[-eco_route_concurrent_redundant_via_effort_level low | medium | high]]
[-rotate_default_vias true | false]
[-off_grid_routing_mode allow_off_grid | allow_end_points_off_grid | no_off_grid]
[-single_connection_to_pins off | standard_cell_pins | all_pins]
[-mark_clock_nets_minor_change true | false]
[-route_soft_rule_effort_level off | min | low | medium | high]
[-threshold_noise_ratio ratio]
[-track_auto_fill true | false]
[-connect_within_pins {{layer mode}...}]
[-tie_off_mode all | rail_only]
[-max_number_of_threads int]
[-default true | false]
```

Data Types

<i>int</i>	integer
<i>layer</i>	string
<i>multiplier</i>	float
<i>ratio</i>	float
<i>mode</i>	string

ARGUMENTS

`-enforce_voltage_areas` *off* | *strict* | *relaxed*

Specifies the level of enforcement of voltage areas. The definition for each mode is

- **off**

Voltage area constraints are turned off for routing.

- **strict**

Uses strict voltage area constraints for routing.

- **relaxed** (default)

Uses relaxed voltage area constraints for routing.

`-reroute_clock_shapes` *true* | *false*

Specifies whether the router can reroute fixed clock wires and vias. The default is *false*.

`-reroute_user_shapes` *true* | *false*

Specifies whether the router can reroute user-created wires and vias. The default is *false*.

`-plan_group_aware` *off* | *all_routing* | *top_level_routing_only*

Specifies whether the plan-group constraints are obeyed. The definition for each mode is

- **off** (the default)

The router ignores plan groups and routes the design flat.

- **all_routing**

The router routes all nets and preserves the hierarchy and pin constraints for the pin-cutting flow.

- **top_level_routing_only**

The router preserves the hierarchy and pin constraints, but it ignores intraplan group nets and routes only those nets that have top-level connections. This mode saves CPU time, but might be less accurate because it ignores the congestion due to intraplan group nets.

When you specify a mode other than **off**, the router gives priority to clock net routing. During floorplanning you might want to route the clocks flat to generate clock pins on the module boundaries. You should buffer these clocks later in the flow. The clock nets are routed before any other nets so that their pin placement has priority over normal signal pins.

`-child_process_net_threshold` *int*

Specifies the minimum number of nets to trigger a separate router process. The default is *-1*, which means that the tool decides when to trigger a separate router process based on the number of nets in the design and the peak memory of the parent process when the routing command is run.

`-verbose_level` *int*

Sets the verbosity level for the routing log file. The value range is from 0 to 1. The default is 0.

`set_route_zrt_common_options`

`-read_user_metal_blockage_layer true | false`
 Specifies whether shapes on metal blockage layers should be honored (true) or ignored (false).
 The default is false.

`-wide_macro_pin_as_fat_wire true | false`
 Specifies whether a wide macro pin or pad pin should have an implicit depth the same as its width.
 The default value is false.

`-route_top_boundary_mode stay_half_min_space_inside | stay_inside | ignore`
 Controls the routing behavior near the top boundary.
 The definition for each mode is

- **stay_half_min_space_inside**
 The routing must be inside the boundary by half of the minimum spacing.
- **stay_inside** (the default)
 The routing must be inside or abut the boundary.
- **ignore**
 Wires can extend outside the boundary.

`-standard_cell_blockage_as_thin true | false`
 Specifies whether to treat wide blockages in standard cells as thin wires.
 The default is false.

`-freeze_layer {{layer true | false}...}`
 Controls whether routing layers are frozen to prevent them from changing during routing.
 The routing layers are specified as m[1-15], where m1 is the bottommost routing layer. m[1-15] are logical names that might not match the names in the technology file.
 When false (the default), the layer is not frozen.
 When true, the layer is frozen.

`-freeze_via_to_frozen_layer true | false`
 Controls the treatment of vias that touch a frozen layer on one side.
 When false (the default), vias adjacent to a frozen layer can be changed.
 When true, vias adjacent to a frozen layer are frozen and cannot be changed.

`-max_layer_mode soft | allow_pin_connection | hard`
 Controls the application of maximum routing layer constraints. Note that the **-number_of_vias_over_max_layer** option can relax the maximum layer constraint as well, but only for vias.
 The definition for each mode is

- **hard** (default)
 No routing can occur above the maximum layer.
- **soft**
 Routing above the maximum layer is discouraged, but not disallowed.
- **allow_pin_connection**
 Routing is allowed above the maximum layer only for pin connections.

`-min_layer_mode soft | allow_pin_connection | hard`
 Controls the application of minimum routing layer constraints. Note that the **-number_of_vias_under_min_layer** option can relax the minimum layer constraint as well, but only for vias.
 The definition for each mode is

- **soft** (default)
 Routing below the minimum layer is discouraged, but not disallowed.
- **hard**
 No routing can occur below the minimum layer.
- **allow_pin_connection**
 Routing is allowed below the minimum layer only for pin connections.

`-extra_preferred_direction_wire_cost_multiplier {{layer multiplier}...}`
 Specifies the cost multiplier for routing in the preferred direction.
 The routing layers are specified as m[1-15], where m1 is the bottommost routing layer. m[1-15] are logical names that might not match the names in the technology file.
 The multiplier value must be between 0.0 and 20.0. The default is 0.0.

`-extra_nonpreferred_direction_wire_cost_multiplier {{layer multiplier}...}`
 Specifies the cost multiplier for routing in the nonpreferred direction.
 The routing layers are specified as m[1-15], where m1 is the bottommost routing layer. m[1-15] are logical names that might not match the names in the technology file.
 The multiplier value must be between 0.0 and 20.0. The default is 0.0.

`-extra_via_cost_multiplier {{layer multiplier}...}`
 Specifies the cost multiplier for vias.
 The routing layers are specified as m[1-15], where m1 is the bottommost routing layer. m[1-15] are logical names that might not match the names in the technology file.
 The multiplier value must be between 0.0 and 20.0. The default is 0.0.

`-number_of_vias_over_max_layer int`
 Allows the router to use stacked vias to access pins or fixed routes above the maximum layer. If the maximum layer mode is set to **soft**, this option is ignored, because in that case access above the maximum layer is allowed.
 The value range is between 0 and 15.
 The default is 1.

`-number_of_vias_under_min_layer int`
 Allows the router to use stacked vias to access pins or fixed routes below the minimum layer. If the minimum layer mode is set to **soft**, this option is ignored, because in that case access below the minimum layer is allowed.
 The value range is between 0 and 15.
 The default is 1.

`-via_array_mode off | swap | rotate | all`
 Specifies the types of rotated via arrays that can be used during signal routing and redundant via insertion.
 The definition for each mode is

- **off**

Do not rotate or swap via arrays.

- **swap** (the default)

Use 1 x N and N x 1 via arrays as rotated equivalent vias.

- **rotate**

Rotate line via arrays instead of swapping row and column numbers.

- **all**

Use all four possible via arrays.

`-post_detail_route_redundant_via_insertion off | low | medium | high`

Enables automatic redundant via insertion after each detail routing step.

The default setting is **off**.

If the value is set to **low**, **medium**, or **high**, the tool performs redundant via insertion after each detail routing change, including initial detail routing, ECO routing, and incremental routing. Enabling this option keeps the redundant vias in the design up-to-date with routing changes. The **low**, **medium**, or **high** setting specifies the effort level used for redundant via insertion.

`-concurrent_redundant_via_mode off | reserve_space | insert_at_high_cost`

Specifies the concurrent redundant via insertion mode for initial routing.

The valid modes are

- **off** (the default)

Concurrent redundant via insertion is turned off.

- **reserve_space**

Concurrent redundant via insertion reserves space for redundant vias but does not instantiate them. The space reservation is done by generating internal soft rules for redundant vias and performing soft-rule-based optimization. The actual instantiation is done later by explicitly calling the **insert_zrt_redundant_vias** command or by setting the **-post_detail_route_redundant_via_insertion** common Zroute option to a value other than **off**.

- **insert_at_high_cost**

Concurrent redundant via insertion creates hard design rules for redundant vias and tries to insert them. Use this mode if only you need a nearly 100 percent conversion rate. It can lead to very long runtimes if the design is congested or the technology parameters are not friendly to redundant via insertion. This mode should be used with caution.

`-concurrent_redundant_via_effort_level low | medium | high`

Specifies the effort level for the concurrent redundant via insertion flow. The default effort level is low. The higher effort levels result in a better redundant via conversion rate at the expense of runtime.

This option takes effect only if the **-concurrent_redundant_via_mode** option is set to a value other than **off**.

`-eco_route_concurrent_redundant_via_mode off | reserve_space`

Specifies the concurrent redundant via insertion mode for ECO routing.

The definition for each mode is

- **off** (the default)

Concurrent redundant via insertion is turned off.

- **reserve_space**

Concurrent redundant via insertion reserves space for redundant vias in ECO routing but does not instantiate them. The space reservation is done by generating internal soft rules for redundant vias and performing soft-rule-based optimization. The actual instantiation is done later by explicitly calling the **insert_zrt_redundant_vias** command or by setting the **-post_detail_route_redundant_via_insertion** common Zroute option to a value other than **offP**.

-eco_route_concurrent_redundant_via_effort_level low | medium | high

Specifies the effort level for the concurrent redundant via insertion flow in ECO routing.

The default effort level is low. The higher effort levels result in a better redundant via conversion rate at the expense of runtime.

This option takes effect only if the **-eco_concurrent_redundant_via_mode** option is set to a value other than **off**.

-rotate_default_vias true | false

Specifies whether the router can rotate default vias.

The default is true.

-off_grid_routing_mode allow_off_grid | allow_end_points_off_grid | no_off_grid

Restricts routing use of off-routing-track wires.

The definition for each mode is

- **allow_off_grid** (default)

Routing can be off the routing tracks.

- **allow_end_points_off_grid**

The router routes most of the vias on grid points and most of the wires on track. The end points of wires can be off track.

- **no_off_grid**

The router routes most of the wires, including end points and vias, on track. You can also restrict off-grid routing by setting the onGrid attribute in the Layer section in the technology file.

The router does on-grid routing if either the **-off_grid_routing_mode** option or the onGrid attribute are set to enforce on-grid routing.

-single_connection_to_pins off | standard_cell_pins | all_pins

Specifies whether the router can connect to a pin only once.

Valid values are

- **off** (the default)

Connect to any pin any number of times.

- **standard_cell_pins**

Connect to standard cell pins only once.

- **all_pins**

Connect to any pin only once.

`-mark_clock_nets_minor_change true | false`
 Controls whether changes on clock tree nets are limited during routing.
 When true (the default), the router can make only limited changes to nets marked as clock tree nets in the database.
 When false, nets marked as clock tree nets in the database are routed normally.

`-route_soft_rule_effort_level off | min | low | medium | high`
 Specifies the effort level for rip up and reroute to fix soft rule design rule violations. You can also specify the optimization effort level for each soft rule by setting the **-optimize_soft_rule_effort_level** option. The minimum effort level of the two settings is used for each soft rule.
 The definition for each effort level is

- **off**
 Does not perform rip up and reroute passes to resolve soft rule design rule violations.
- **min**
 Uses the smallest number of rip up and reroute passes to resolve soft rule design rule violations.
- **low**
 Uses a small number of rip up and reroute passes to resolve soft rule design rule violations.
- **medium** (medium)
 Uses a medium number of rip up and reroute passes to resolve soft rule design rule violations.
- **high**
 Treats soft rule design rule violations the same as regular design rule violations during rip up and reroute.

`-threshold_noise_ratio ratio`
 Specifies the noise threshold as a fraction of the operating voltage.
 The value range is from 0 to 1.
 The default is 0.35.

`-track_auto_fill true | false`
 Specifies whether to fill empty space with routing tracks.
 The default is true.

`-connect_within_pins {{layer mode}...}`
 Specifies, for each layer, whether to connect to pins by using vias or wires contained within the pin shapes.
 The routing layers are specified as `m[1-15]`, where `m1` is the bottommost routing layer. `m[1-15]` are logical names that might not match the names in the technology file.
 The valid mode values are

- **off** (default)
 Connect by using wires or vias anywhere on the pins.
- **via_standard_cell_pins**

Connect to standard cell pins by using vias contained within the pin shapes.

- **via_wire_standard_cell_pins**

Connect to standard cell pins by using vias and wires contained within the pin shapes.

- **via_all_pins**

Connect to any pins by using vias contained within the pin shapes.

- **via_wire_all_pins**

Connect to any pin by using vias and wires contained within the pin shapes.

-tie_off_mode all | rail_only

Controls whether the router can connect tie-off nets to any power or ground (PG) structure, such as straps, rails, or pins or only to a PG rail. The definition for each mode is

- **all** (default)

Tie-off nets can be connected to any PG structure.

- **rail_only**

Tie-off nets can be connected only to a PG rail.

-max_number_of_threads int

Specifies the maximum number of concurrent threads the router can use. The actual number of threads used might be less than this number if a sufficient number of license keys is not available at the time the routing command is run. The default is 1.

-default true | false

If true, this option resets all options to their default values. The default is false.

DESCRIPTION

The **set_route_zrt_common_options** command sets Zroute options that are common to global routing, track assignment, and detail routing.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets the **-enforce_voltage_areas** option to off.

```
prompt> set_route_zrt_common_options -enforce_voltage_areas off
```

The following example sets the **-threshold_noise_ratio** option to 0.2.

```
prompt> set_route_zrt_common_options -threshold_noise_ratio 0.2
```

The following example sets the **-freeze_layer** option to true for m3 and m4.

```
prompt> set_route_zrt_common_options -freeze_layer {{m3 true} {m4 true}}
```

SEE ALSO

```
get_route_zrt_common_options(2)  
report_route_zrt_common_options(2)
```

set_route_zrt_detail_options

Sets the options for detail routing.

SYNTAX

```
status set_route_zrt_detail_options
[-antenna true | false]
[-antenna_on_iteration num]
[-antenna_verbose_level num]
[-check_antenna_on_pg true | false]
[-check_pin_min_area_min_length true | false]
[-check_port_min_area_min_length true | false]
[-elapsed_time_limit limit]
[-default true | false]
[-default_diode_protection real]
[-default_gate_size real]
[-default_port_external_antenna_area real]
[-default_port_external_gate_size real]
[-diagonal_min_width true | false]
[-diode_insertion_mode new_and_spare | spare | new]
[-diode_libcell_names lib_cells]
[-eco_route_use_soft_spacing_for_timing_optimization true | false]
[-force_max_number_iterations true | false]
[-generate_extra_off_grid_pin_tracks true | false]
[-generate_off_grid_feed_through_tracks off | low | medium | high]
[-ignore_drc {{drc_name true | false}...}]
[-ignore_var_spacing_to_blockage true | false]
[-ignore_var_spacing_to_pg true | false]
[-insert_diodes_during_routing true | false]
[-max_antenna_pin_count int]
[-merge_gates_for_antenna true | false]
[-optimize_tie_off_effort_level off | low | high]
[-optimize_wire_via_effort_level off | low | medium | high]
[-pin_taper_mode default_width | pin_width | off]
[-port_antenna_mode float | jump | top_layer]
[-save_after_iterations iter_list]
[-save_cell_prefix prefix]
[-reshield_rerouted_nets off | unshield | reshield]
[-reuse_filler_locations_for_diodes true | false]
[-timing_driven true | false]
[-top_layer_antenna_fix_threshold int]
[-use_default_width_for_min_area_min_len_stub true | false]
[-use_wide_wire_to_input_pin true | false]
[-use_wide_wire_to_macro_pin true | false]
[-use_wide_wire_to_output_pin true | false]
[-use_wide_wire_to_pad_pin same_as_macro_pin | true | false]
[-use_wide_wire_to_port same_as_macro_pin | true | false]
[-user_defined_partition {llx lly urx ury}]
[-var_spacing_to_same_net true | false]
```

Data Types

num integer

<i>limit</i>	integer
<i>real</i>	float
<i>lib_cells</i>	collection
<i>int</i>	integer
<i>iter_list</i>	collection
<i>prefix</i>	string
<i>llx</i>	float
<i>lly</i>	float
<i>urx</i>	float
<i>ury</i>	float

ARGUMENTS

-antenna true | false
Enables (true) or disables (false) antenna analysis.
The default is true.

-antenna_on_iteration *num*
Specifies the routing iteration at which antenna analysis and optimization is turned on.
The range is from 1 to 19 (the routing iteration number starts at 0).
The default is 1.

-antenna_verbose_level *num*
Specifies the verbosity level to print extra information for antenna analysis and optimization.
The range is from 0 to 1.
The default is 1.

-check_antenna_on_pg true | false
Enables (true) or disables (false) analysis and optimization of antennas on power and ground nets.
The default is false.

-check_pin_min_area_min_length true | false
Enables (true) or disables (false) checking for minimum area and minimum length rules on pins.
The default is false.

-check_port_min_area_min_length true | false
Enables (true) or disables (false) checking for minimum area and minimum length rules on physical ports (terminals).
The default is true.

-elapsed_time_limit *limit*
Specifies the limit for elapsed wall-clock time in minutes.
The range is from -1 to MAX_INT.
The default is -1, which means no limit.

-default true | false
If true, this option resets all options to their default values.
The default is false.

`-default_diode_protection real`
 Specifies the diode protection value used for pins during antenna analysis if the value is not specified in the cell.
 The range is from 0.0 to 1,000,000.00
 The default is 0.0.

`-default_gate_size real`
 Specifies the gate size used for pins during antenna analysis if the gate size is not specified in the cell.
 The range is from 0.0 to 1,000,000.00
 The default is 0.0.

`-default_port_external_antenna_area real`
 Specifies the antenna area value used for ports (top-level pins) during antenna analysis if the antenna area is not specified in the cell.
 The range is from 0.0 to 1,000,000.00
 The default is 0.0.

`-default_port_external_gate_size real`
 Specifies the gate size used for ports (top-level pins) during antenna analysis if the gate size is not specified in the cell.
 The range is from 0.0 to 1,000,000.00
 The default is 0.0.

`-diagonal_min_width true | false`
 Specifies whether to use diagonal distance for minimum width violations.
 If true (the default), diagonal distance is used.
 If false, manhattan distance is used.

`-diode_insertion_mode new_and_spare | spare | new`
 Specifies whether to fix antenna violations by using both new and spare diodes (**new_and_spare**), spare diodes only (**spare**), or new diodes only (**new**).
 If you do not specify this option, Zroute uses both new and spare diodes.

`-diode_libcell_names lib_cells`
 Specifies the diode library cells to use for antenna fixing.
 The default is an empty list, in which case diodes are detected automatically.

`-eco_route_use_soft_spacing_for_timing_optimization true | false`
 Enables (true) or disables (false) soft-rule-based timing optimization during ECO routing.
 This option works only if the **-timing_driven** option is set to true.
 The default is true.

`-force_max_number_iterations true | false`
 Controls whether the maximum number of iterations must be run when DRCs do not converge.
 The default is false.

`-generate_extra_off_grid_pin_tracks true | false`
 Specifies whether to generate extra off-grid routing tracks for pin connections.
 The default is false.

`-generate_off_grid_feed_through_tracks` `off` | `low` | `medium` | `high`
 Specifies the effort level used to generate extra off-grid routing tracks for feedthrough nets between blockages.
 The default is `low`.

`-ignore_drc` `{{drc_name true | false}}...`
 Controls whether the router ignores specific design rule constraints (DRCs).
 When a DRC is ignored, the router neither analyzes nor fixes those violations.
 The valid DRC keywords are

- **`same_net_metal_space`**
 Ignore all same net metal space violations.
- **`same_net_enclosed_cut_space`**
 Ignore same net enclosed cut space violations.
- **`all_same_net_drc_for_frozen_net`**
 Ignore all same net violations on frozen nets.
 The default is `false` for all DRC types, which means that none of them are ignored.

`-ignore_var_spacing_to_blockage` `true` | `false`
 Specifies whether variable routing rule spacing is ignored against blockages.
 The default is `false`.

`-ignore_var_spacing_to_pg` `true` | `false`
 Specifies whether variable routing rule spacing is ignored against power and ground nets.
 The default is `false`.

`-insert_diodes_during_routing` `true` | `false`
 Specifies whether the router can use diode insertion to fix antenna violations.
 The default is `false`.

`-max_antenna_pin_count` `int`
 Specifies the maximum number of pins on a net on which antenna checking is performed. If a net has more than the specified number of pins, antenna checking is skipped; otherwise, antenna checking is performed.
 The range is from -1 to 1,000,000.
 The default is -1 (which means no limit).

`-merge_gates_for_antenna` `true` | `false`
 Enables (true) or disables (false) merging gates for antenna analysis.
 The default is `true`.

`-optimize_tie_off_effort_level` `off` | `low` | `high`
 Specifies the effort level used to optimize wire length and via counts for tie-off nets.
 The valid values are

- **`off`**
 No extra effort to optimize wire length and via counts for tie-off nets.
- **`low`** (the default)

Low effort to optimize wire length and via counts for tie-off nets.

- **high**

High effort to optimize wire length and via counts for tie-off nets.

`-optimize_wire_via_effort_level` `off` | `low` | `medium` | `high`

Specifies the effort level used to optimize wire length and via counts.
The valid values are

- **off**

No extra effort to optimize wire length and via counts.

- **low** (the default)

Low effort to optimize wire length and via counts.

- **medium**

Medium effort to optimize wire length and via counts.

- **high**

High effort to optimize wire length and via counts.

`-pin_taper_mode` `default_width` | `pin_width` | `off`

Specifies the pin tapering mode.
The valid values are

- **default_width** (the default)

Variable rule connections to pins are tapered to the default rule width.

- **pin_width**

Variable rule connections to pins are tapered to the pin width.

- **off**

Pin tapering is not performed.

`-port_antenna_mode` `float` | `jump` | `top_layer`

Specifies how the ports (top-level pins) are treated for antenna considerations.
The valid values are

- **float** (the default)

Treat ports as floating wires.

- **jump**

Treat ports as if they are connected to a huge antenna and break the antenna, which requires a jumper at the top metal layer.

- **top_layer**

Connect all input pins to ports only through the highest routing layer of the net.

`-save_after_iterations` *iter_list*

Saves the intermediate cell after the specified iterations.
The default is an empty list.

-save_cell_prefix *prefix*
 Specifies the prefix to be used in the saved cell name.
 The default is "DR".

-reshield_rerouted_nets off | unshield | reshield
 Specifies whether to automatically unshield or reshield rerouted shielded nets during routing.
 The valid values are

- **off** (the default)
 Do not unshield or reshield rerouted shielded nets.
- **unshield**
 Delete existing shielding that is associated with rerouted shielded nets and reconnect the floating shielding if necessary.
- **reshield**
 Unshield rerouted shielded nets, automatically reshield them, and then reconnect the floating shielding. If the router cannot reconnect the floating shielding, it removes it.

-reuse_filler_locations_for_diodes true | false
 Specifies whether to reuse filler cell locations for inserting diodes.
 The default is true.

-timing_driven true | false
 Specifies whether timing-driven routing is enabled.
 The default is false.

-top_layer_antenna_fix_threshold *int*
 Specifies the threshold for fixing antenna violations on the top routing layer. Antenna violations on the top routing layer are fixed if they are within $int * 0.1$ percent of the safe ratio.
 The range is from -1 to 10.
 The default is -1 (which means no limit).

-use_default_width_for_min_area_min_len_stub true | false
 Specifies whether to use default width stubs to fix minimum area and minimum length violations.
 When false (the default), stubs are the same width as the via surround. When true, stubs are the default width.

-use_wide_wire_to_input_pin true | false
 Specifies whether pin tapering to input pins is disallowed.
 The default is false.

-use_wide_wire_to_macro_pin true | false
 Specifies whether pin tapering to macro pins is disallowed.
 The default is false.

-use_wide_wire_to_output_pin true | false
 Specifies whether pin tapering to output pins is disallowed.
 The default is false.

`-use_wide_wire_to_pad_pin same_as_macro_pin | true | false`

Controls pin tapering to pad pins.

The valid values are

- **same_as_macro_pin** (the default)

The setting is taken from the `-use_wide_wire_to_macro_pin` option.

- **true**

Disallow pin tapering to pad pins.

- **false**

Allow pin tapering to pad pins.

`-use_wide_wire_to_port same_as_macro_pin | true | false`

Controls pin tapering to ports.

The valid values are

- **same_as_macro_pin** (the default)

The setting is taken from the `-use_wide_wire_to_macro_pin` option.

- **true**

Disallow pin tapering to ports.

- **false**

Allow pin tapering to ports.

`-user_defined_partition {llx lly urx ury}`

Specifies the coordinates, in database units, of a rectangular partition `{llx lly urx ury}` to be added for routing. In general, the detail router automatically generates partitions to fix the DRCs in the design. In some situations, it might not generate the right-sized partitions. For example, when fixing DRCs in a river routing (single layer) situation over a macro, it might need the partitions to be thin and long. In such situations, you can assist the router by adding the right-sized partitions.

`-var_spacing_to_same_net true | false`

Specifies if variable routing rule spacing is to be applied to shapes of the same net.

The default is false.

DESCRIPTION

The **set_route_zrt_detail_options** command sets the options for detail routing.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets the **-elapsed_time_limit** option to 60 minutes.

set_route_zrt_detail_options

2776

```
prompt> set_route_zrt_detail_options -elapsed_time_limit 60
```

The following example sets the **-antenna** option to false.

```
prompt> set_route_zrt_detail_options -antenna false
```

The following example shows how you can use the **-ignore_drc** option to allow the router to ignore same net metal space violations but not same net enclosed cut space violations.

```
prompt> set_route_zrt_detail_options \  
-ignore_drc {{same_net_metal_space true} \  
{same_net_enclosed_cut_space false}}
```

The following example uses the **-macro_pin_antenna_mode** option to set the macro pin antenna mode to **jump**.

```
prompt> set_route_zrt_detail_options -macro_pin_antenna_mode jump
```

The following example adds a user-defined partition for routing by using the **-user_defined_partition** option.

```
prompt> set_route_zrt_detail_options \  
-user_defined_partition {100 200 500 3000}
```

SEE ALSO

```
get_route_zrt_detail_options(2)  
report_route_zrt_detail_options(2)  
route_opt(2)  
route_zrt_auto(2)  
route_zrt_detail(2)
```

set_route_zrt_global_options

Sets the options for Zroute global routing.

SYNTAX

```
status set_route_zrt_global_options
[-clock_topology normal | comb]
[-comb_distance int]
[-comb_max_connections int]
[-congestion_map_only true | false]
[-layer_based_congestion_map true | false]
[-timing_driven true | false]
[-crosstalk_driven true | false]
[-effort minimum | low | medium | high]
[-force_full_effort true | false]
[-extra_blocked_layer_utilization_reduction int]
[-macro_boundary_track_utilization int]
[-macro_boundary_width int]
[-macro_corner_track_utilization int]
[-default true | false]
```

Data Types

int integer

ARGUMENTS

-clock_topology normal | comb

Specifies the global-routing clock topology. Valid values are

- **normal** (default)

Use normal routing for clock nets.

- **comb**

Use comb routing for clock nets with a mesh or trunk.

-comb_distance *int*

Specifies the number of global route cells (gcells) within which to connect clock pins to the clock mesh.

The range is from 0 to 50.

The default is 2.

-comb_max_connections *int*

Specifies the maximum number of pins that can be connected to any clock strap.

The range is from -1 to 50.

The default is -1, which means no limit.

-congestion_map_only true | false

Controls whether glinks are created during global routing. If false (the default), the global router creates both glinks and a congestion map. If true, the global router creates the congestion map without creating glinks.

-layer_based_congestion_map true | false
Controls whether both layer-based and aggregate congestion maps are created. If true (the default), both layer-based and aggregate congestion maps are created. If false, only an aggregate congestion map is created.

-timing_driven true | false
Enables (true) or disables (false) timing-driven global routing. The default is false.

-crosstalk_driven true | false
Enables (true) or disables (false) crosstalk-driven global routing. The default is false.

-effort minimum | low | medium | high
Specifies the global routing effort. Valid values are

- **minimum**
Very fast run time. Runs a maximum of 2 phases.
- **low**
Fast run time. Runs a maximum of 3 phases.
- **medium** (default)
Medium run time. Runs a maximum of 4 phases.
- **high**
Slow run time, but better quality of results. Runs a maximum of 5 phases.

-force_full_effort true | false
Forces the maximum number of phases specified by the **-effort** option. The default is false.

-extra_blocked_layer_utilization_reduction *int*
Specifies the percentage of additional utilization to be reduced for areas where either the upper or lower layer is blocked and switching to the opposite layer would take more than one track resource. The range is from 0 to 100. The default is 0.

-macro_boundary_track_utilization *int*
Specifies the percentage of tracks to be used along macro boundaries. The range is from 0 to 100. The default is 100.

-macro_boundary_width *int*
Specifies the width of the macro boundary in terms of global route cells. The range is from 0 to 10. The default is 1.

-macro_corner_track_utilization *int*
Specifies the percentage of tracks to be used along macro corners. The range is from 0 to 100. The default is 100.

-default true | false
Resets all options to their default values when set to true.

The default is false.

DESCRIPTION

The **set_route_zrt_global_options** command sets the options for Zroute global routing.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets the **-timing_driven** option to true.

```
prompt> set_route_zrt_global_options -timing_driven true
```

The following example sets the **-clock_topology** option to comb.

```
prompt> set_route_zrt_global_options -clock_topology comb
```

SEE ALSO

```
report_route_zrt_global_options(2)  
route_opt(2)  
route_zrt_global(2)
```

set_route_zrt_track_options

Sets the options for track assignment.

SYNTAX

```
status set_route_zrt_track_options  
[-crosstalk_driven true | false]  
[-default true | false]  
[-timing_driven true | false]
```

ARGUMENTS

-crosstalk_driven true | false
Enables (true) or disables (false) crosstalk-driven track assignment.
The default is false.

-default true | false
If true, this option resets all options to their default values.
The default is false.

-timing_driven true | false
Enables (true) or disables (false) timing-driven track assignment.
The default is false.

DESCRIPTION

The **set_route_zrt_track_options** command sets the options for track assignment.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets the **-timing_driven** option to true.

```
prompt> set_route_zrt_track_options -timing_driven true
```

SEE ALSO

report_route_zrt_track_options(2)

set_row_type

Sets a specified row type attribute on the specified rows.

SYNTAX

```
status set_row_type  
[-site site_name row_name_list]  
-type row_type
```

Data Types

<i>site_name</i>	string
<i>row_name_list</i>	list
<i>row_type</i>	int

ARGUMENTS

-site *site_name*
Specifies the name of the site, the specified *row_type* is set on all rows with this name. This option and the *row_name_list* are mutually exclusive; use only one.

row_name_list
Specifies one or more valid rows on which to set the row type attribute. The specified row names must exist in the floorplan. This argument and the **-site** option are mutually exclusive; use only one.

DESCRIPTION

The **set_row_type** command sets a row type attribute on a specified list of rows or on the rows of the specified site.

When you want to associate all the rows in a site to a particular *row_type*, use the **-site** option.

A row can only have one row type attribute.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows how to set a row type of 1 on the specified rows.

```
prompt>set_row_type -type_id 1 {ROW_0 ROW_1 ROW_2 ROW_4}
```

The following example shows how to set a row type for all rows in a specified site.

```
prompt>set_row_type -type_id 1 -site CORE1
```

SEE ALSO

```
set_cell_row_type(2)
```

set_rp_group_options

Sets relative placement group attributes on the specified relative placement groups.

SYNTAX

```
collection set_rp_group_options  
rp_groups  
[-alignment bottom-left | bottom-pin | bottom-right]  
[-pin_align_name pin_name]  
[-utilization percentage]  
[-ignore]  
[-x_offset float]  
[-y_offset float]  
[-compress]  
[-cts_option fixed_placement | size_only]  
[-route_opt_option fixed_placement | in_place_size_only]  
[-psynopt_option fixed_placement | size_only]  
[-move_effort low | medium | high]  
[-allow_keepout_over_tapcell false | true]
```

Data Types

<i>rp_groups</i>	list or collection
<i>pin_name</i>	string
<i>percentage</i>	float
<i>float</i>	percentage

ARGUMENTS

rp_groups

Specifies the relative placement groups for which you want to change the attributes. This is a required argument.

-alignment bottom-left | bottom-pin | bottom-right

Specifies the default alignment method to use when placing leaf cells and relative placement groups in the specified relative placement groups. If you do not specify this option, the tool uses bottom-left alignment. You can override the default alignment method for a specific leaf cell or relative placement group when you add it to a relative placement group (by using the **add_to_rp_group** command).

-pin_align_name *pin_name*

Specifies the default alignment pin for the specified relative placement groups. During placement, the tool uses the alignment pin's location to align the cells within a column when you use the bottom-pin alignment type. You can override the alignment pin for a specific leaf cell when you add the cell to the group (by using the **add_to_rp_group** command).

-utilization *percentage*

Specifies the utilization percentage to use for placement of the specified relative placement groups. Utilization is represented as a floating-point number between 0.0 (representing 0%) and 1.0 (the default, representing

set_rp_group_options

100%).

-ignore

Indicates that the tool is to ignore the specified relative placement groups during placement and not treat them as relative placement groups. Instead, the tool places all of the group's parts individually, as it would normally do when there is no relative placement.

-origin

-x_offset *float*

Specifies a left coordinate (anchor) for the group, in microns, relative to the lower-left corner in the core area, for top level relative placement group. It is ignored for a sub-level relative placement group. Specifying only an x-offset allows a group to slide in the y-direction.

-y_offset *float*

Specifies a lower coordinate (anchor) for the group, in microns, relative to the lower-left corner in the core area, for top level relative placement group. It is ignored for a sub-level relative placement group. Specifying only a y-offset allows a group to slide in the x-direction.

-compress

Applies compression in the horizontal direction to a relative placement group during placement. Setting this option places each row of a relative placement group without any gaps between leaf cells, lower-level hierarchical relative placement groups, or keepouts. Column alignment is not maintained when you use **-compress**.

If you use **-alignment bottom-right** or **-alignment bottom-pin** and specify **-compress**, the tool issues a warning that **-alignment** is observed and **-compress** is ignored.

If both **-utilization** and **-compress** are specified, the utilization constraints are observed with gaps between leaf elements in a relative placement row.

The **-compress** option does not propagate from a parent group to child groups.

-cts_option *fixed_placement* | *size_only*

Specifies how to treat the cells in the relative placement group during clock tree synthesis and clock tree optimization.

If you specify *fixed_placement*, the cells in the relative placement group are treated as fixed during **compile_clock_tree**, **optimize_clock_tree**, and **fbclock_opt** activity and cannot be sized or moved.

If you specify *size_only*, the cells in the relative placement group can only be sized. This option applies to the **compile_clock_tree**, **optimize_clock_tree**, and **clock_opt** commands.

This option applies only to top-level relative placement groups and is ignored for hierarchical relative placement groups. The top-level value is propagated to the hierarchical relative placement groups.

-route_opt_option *fixed_placement* | *in_place_size_only*

Specifies how to treat the cells in the relative placement group during **route_opt** and **psynopt on_route**.

If you specify *fixed_placement*, the cells in the relative placement group are treated as fixed during **route_opt** and **psynopt on_route** and cannot be sized or moved.

If you specify *in_place_size_only*, sizing changes are constrained for minimal

engineering change order (ECO) placement changes. For example, a cell is sized to improve timing or design rule costs only if the newly-sized cell can fit into any available space adjacent to the original cell location. The resulting transformation is verified to ensure it is legal. This option applies only to top-level relative placement groups and is ignored for hierarchical relative placement groups. The top-level value is propagated to the hierarchical relative placement groups.

`-psynopt_option fixed_placement | size_only`

Specifies how to treat the cells in the relative placement group during **psynopt** and **place_opt**.

If you specify `fixed_placement`, the cells in the relative placement group are treated as fixed during **psynopt** and cannot be sized or moved. This value is not recommended for `fbplace_opt`.

If you specify `size_only`, the cells in the relative placement group can only be sized.

This option applies only to top-level relative placement groups and is ignored for hierarchical relative placement groups. The top-level value is propagated to the hierarchical relative placement groups.

`-move_effort low | medium | high`

This option controls the extent to which a relative placement group can be moved from its initial location to preserve the relative placement without violating relative placement constraints. The size of the region searched for placement of a relative placement group is progressively reduced as you reduce the the effort from high to medium to low.

Coarse placement estimates an initial location for every top-level relative placement group. If those groups are placed at the same locations returned by coarse placement, the relative placement groups might need to be broken, which would violate the relative placement constraints for those groups.

Specifying high effort allows the higher movement of a relative placement group from its initial location in search of a location compared to low and medium effort, to maintain the relative placement constraints and potentially avoid breaking the relative placement group. The tradeoff for maintaining the relative placement group is a potential loss of QoR, because the group is moved from the location initially determined by coarse placement.

Specifying low effort indicates that the relative placement group is placed as close as possible to the initial location returned by coarse placement. The tradeoff for maintaining the location is a higher potential for breaking the relative placement groups and violating the relative placement constraints for some relative placement groups.

`-allow_keepout_over_tapcell false | true`

Specifies that the hard keepout in the relative placement group should be overlapped with tap cells if needed. By default, the value is false.

`-compression`

DESCRIPTION

The **set_rp_group_options** command sets the attributes of the specified relative placement groups. The same attributes can be set during the creation of the relative placement groups by using the **create_rp_group** command. This command is supported only for designs that do not contain multiply-instantiated designs.

This command returns a collection containing the relative placement groups for which the attributes have changed. If attributes have not changed for any object, the empty string is returned.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses **set_rp_group_options** to change the attributes for a relative placement group.

```
prompt> set_rp_group_options grp_ripple -utilization 0.95 -ignore  
{ripple::grp_ripple}
```

SEE ALSO

```
add_to_rp_group(2)  
all_rp_groups(2)  
create_rp_group(2)  
get_rp_groups(2)  
remove_from_rp_group(2)  
remove_rp_groups(2)  
report_rp_group_options(2)  
write_rp_groups(2)
```

set_scaling_lib_group

Specifies the scaling_lib_group to use for the current design, or a subdesign.

SYNTAX

```
status set_scaling_lib_group  
[-min min_group]  
[-max max_group]  
[-object_list objects]  
[group]
```

Data Types

<i>min_group</i>	string
<i>max_group</i>	string
<i>objects</i>	list
<i>group</i>	string

ARGUMENTS

-min *min_group*
Specifies the scaling library group to use for minimum delay analysis. If you do not specify a scaling library group for minimum delay analysis, the maximum group is used. The **-min** option cannot be used without the **-max** option. If neither of these options are specified, the library group applies to both max and min delay analysis.

-max *max_group*
Specifies the scaling library group to use for maximum delay analysis. If this option is not specified, the scaling group will be used for both maximum and minimum delay analysis.

-object_list *objects*
Specifies the cells or top level ports on which to specify the scaling library group(s). If you do not use this option, the group(s) are specified for the top level design. This option accepts both leaf cells and hierarchical blocks.

group
Specifies the scaling library group to use for both maximum and minimum delay analysis.

DESCRIPTION

The **set_scaling_lib_group** command set a scaling library group on the design objects, such that the tool can do interpolation between libraries in this group for voltage and/or temperature. The user can set scaling library for max and/or min mode. It is possible to set multiple scaling lib groups on an object.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

```
prompt> define_scaling_lib_group -name g1 {0.9.db 1.1.db 1.3.db}  
prompt> set_scaling_lib_group -object_list top/inv g1
```

SEE ALSO

```
define_scaling_lib_group(2)  
remove_scaling_lib_group(2)  
create_scenario(2)  
set_operating_conditions(2)
```

set_scan_pin_type

Sets the scan-in or scan-out type for the specified pin.

SYNTAX

```
status set_scan_pin_type  
-type in | out  
-pin pin | -ref_pin physical_lib_pin
```

Data Types

<i>pin</i>	collection
<i>physical_lib_pin</i>	collection

ARGUMENTS

```
-type in | out  
    Sets the type to scan-in or scan-out.  
  
-pin pin  
    Specifies a collection of one pin.  
  
-ref_pin physical_lib_pin  
    Specifies a collection of one physical lib pin.
```

DESCRIPTION

This command sets the scan-in or scan-out type on a given pin prior to executing the **trace_scan_chain** command. Scan-in and scan-out types are required to trace through multi-input cells in the scan path that are not scan cells. The scan type is set on the pin of the reference library cell. You must set either a pin or a physical lib pin as input.

The command returns 1 on success and 0 on failure.

You can specify the type as **in** or **out** for scan-in and scan-out, respectively.

To check the scan type of a pin, you can use the **is_scan_in** and **is_scan_out** attributes for scan-in and scan-out, respectively.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets the *I* pin of the *buffd1* buffer in the *cb18os120.mw* library to have scan-in type.

```
set_scan_pin_type  
2790
```

```
prompt> set_scan_pin_type -type in -ref_pin\  
[get_physical_lib_pins cb18os120.mw/buffd1/I]
```

The following example sets the *I* pin of the *timer/buffd1* buffer to have scan-in type and uses the **get_attribute** command to see if the type is set.

```
prompt> set_scan_pin_type -type in -pin [get_pins timer/buffd1/I]  
1  
prompt> get_attribute [get_pins timer/buffd1/I] is_scan_in  
true
```

SEE ALSO

```
get_attribute(2)  
get_physical_lib_pins(2)  
get_pins(2)  
remove_scan_pin_type(2)  
trace_scan_chain(2)
```

set_scenario_options

Sets the scenario options for one or more scenarios.

SYNTAX

```
status set_scenario_options  
[-scenarios scenario_list]  
[-leakage_only true | false]  
[-hold_only true | false]  
[-setup true | false]  
[-hold true | false]  
[-leakage_power true | false]  
[-dynamic_power true | false]  
[-reset_all true | false]
```

Data Types

scenario_list list

ARGUMENTS

- scenarios *scenario_list***
Specifies the list of scenarios to apply the scenario options. When this option is not specified the command applies the scenario options to the current scenario.
- leakage_only true | false**
When set to true, specifies that the scenario is designated only for leakage power cost. All other constraints from the scenario are ignored and their costs considered as 0 (zero). Default value of this option is false. This option can be applied only to active scenarios.
- hold_only true | false**
When set to true, specifies that the setup cost and DRC checking are turned off for the scenario. But the leakage power cost is considered. Default value of this option is false. This option can be applied only to active scenarios.
- setup true | false**
When set to true, enables setup(max delay) cost for the scenario; disables when set to false. When set to false, optimization ignores setup(max_delay) violation in the scenario. Default value of this option is true.
- hold true | false**
When set to true, enables hold(min delay) cost for the scenario; disables when false. When set to false, optimization ignores hold(min_delay) violation in the scenario. Default value of this option is true.
- leakage_power true | false**
When set to true, enables leakage power cost for the scenario; disables when false. Default value of this option is true.

`-dynamic_power true | false`

When set to true, enables dynamic power cost for the scenario; disables when false. Default value of this option is true. When you set this option to true, you must also set the `-setup` option to true, for dynamic power calculation.

`-reset_all true | false`

When set to true, resets all scenario options to default values. When set to false, the scenario options remain unchanged.

DESCRIPTION

This command sets options to restrict a scenario to be used for specific costs. Combinations of **`-setup`**, **`-hold`**, **`-leakage_power`**, and **`-dynamic_power`** can be used to restrict a scenario. The **`-leakage_only`** option sets the scenario for leakage power, but not for minimum or maximum delays and DRCs. The **`-hold_only`** option disables the scenario for max delays and DRCs. All scenario options can be applied to active and inactive scenarios. All scenario options, except the **`-leakage_only`** and **`-hold_only`**, can be applied to active and inactive scenarios. The **`report_scenarios`** command can be used to see the current settings for **`-leakage_only`** and **`-hold_only`** options for all active scenarios. The **`report_scenario_options`** command can be used to report the options set by this command.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example uses **`set_scenario_options`** command to set the MODE1 scenario as a leakage_only scenario that will be used only for leakage power in the multicorner-multimode flow:

```
prompt> create_scenario MODE1
```

```
Warning: Discarding all scenario specific information previously defined in this session. (UID-1008)
```

```
Current scenario is: MODE1
```

```
1
```

```
prompt> set_scenario_options -leakage_only true
```

```
1
```

```
prompt> report_scenarios
```

```
*****
Report : scenarios
Design : TEST03
Scenario(s): MODE1
Version: C-2009.06
Date   : Wed Mar 19 14:57:08 2008
*****
```

```
All scenarios (Total=1): MODE1
```

All Active scenarios (Total=1): MODEL
Current scenario : MODEL
CTS scenario : not defined.

Scenario #0: MODEL is active.
Scenario options: Leakage-only
Has timing derate: No

Library(s) Used:
cb_max (File: /remote/lib/cb_max.db)

Operating condition(s) Used:
Analysis Type : bc_wc
No operating condition was set in scenario MODEL

TLU+ Files Used:
There is no TLU+ settings available.

Number of leakage-only scenario(s): 1
Warning: All the active scenarios are leakage-only scenario. This is not
allowed. Please check your settings!
Number of hold-only scenario(s): 0
1

The following example uses the **set_scenario_options** command to restrict the scenario
MODEL only for hold timing in the multicorner-multimode flow:

```
prompt> create_scenario MODEL
Warning: Discarding all scenario specific information previously defined in this
session. (UID-1008)
Current scenario is: MODEL
1
```

```
prompt> set_scenario_options -setup false -hold true \
-leakage_power false -dynamic_power false
1
```

```
prompt> report_scenario_options
```

```
*****
Report : scenario options
Design : TEST03
Version: C-2009.06
Date   : Wed Mar 19 14:57:08 2008
*****
```

Scenario: MODEL is active.

```
leakage_only : false
hold_only    : false
setup        : false
hold         : true
leakage_power : false
dynamic_power : false
```


SEE ALSO

`report_scenario_options(2)`
`create_scenario(2)`
`current_scenario(2)`
`report_scenarios(2)`
`set_fix_hold(2)`
`set_max_leakage_power(2)`

set_scope

Specifies the current UPF scope. This command is supported only in UPF mode.

SYNTAX

```
string set_scope  
[instance]
```

Data Types

instance string

ARGUMENTS

instance

Specifies the working instance in dc_shell.

- If *instance* is not specified, the focus returns to the top level of the current design.
- If *instance* is ".", the tool returns to the working instance.
- If *instance* is "..", the context is moved up one level in the hierarchy of the current design.
- If *instance* begins with "/", the tool returns to the working instance of the design whose name is after the "/".
- Multiple levels of hierarchy can be traversed in a single call to the **set_scope** command, by separating the cell names with a slash (/). More complex examples of *instance* arguments are described in the EXAMPLES section below.

DESCRIPTION

The **set_scope** command sets the current UPF scope. Functionally, this command is a subset of the **current_instance** command, but has a different return value. Upon success, the **set_scope** command returns the current UPF scope prior to the execution of this command as a full path string relative to the current design. The command returns a null string upon failure.

The **set_scope** command does not work on leaf cells. If you attempt to run the **set_scope** command on a leaf cell, the tool issues the following error message:

```
Error: A leaf cell instance A/B/leaf has been specified as the scope. (UPF-012)
```

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets the UPF scope to the U1/U2 hierarchy, if the U1 cell exists in the current hierarchy and the U2 cell exists in the U1 cell.

```
prompt> set_scope U1/U2
```

The following example sets the scope to two levels of hierarchy above the current instance, and then down one level to the MY_INST cell.

```
prompt> set_scope ../../MY_INST
```

SEE ALSO

`current_design(2)`
`current_instance(2)`

set_si_options

Defines signal integrity options used for analysis or optimization.

SYNTAX

```
status set_si_options
[-delta_delay true | false]
[-static_noise true | false]
[-timing_window true | false]
[-min_delta_delay true | false]
[-static_noise_threshold_above_low threshold_value]
[-static_noise_threshold_below_high threshold_value]
[-route_xtalk_prevention true | false]
[-route_xtalk_prevention_threshold threshold_value]
[-analysis_effort low | medium]
[-max_transition_mode normal_slew | total_slew]
[-reselect true | false]
```

Data Types

threshold_value float

ARGUMENTS

- delta_delay true | false
Specifies whether crosstalk delta delay is considered in timing and optimization. When set to **false**, crosstalk delta delay is not considered. This options is disabled by default
- static_noise true | false
Specifies whether static noise is considered in optimization. When set to **false**, static noise is not considered. This options is disabled by default
- timing_window true | false
Specifies whether timing window overlapping is considered in crosstalk analysis. When set to **true**, timing window is considered. This options is disabled by default and can only be enabled when -delta_delay (or -static_noise) option is set to "true".
- min_delta_delay true | false
Specifies whether min delta delay is considered in timing and optimization. When set to **true**, min delta delay enables crosstalk calculation for min timing paths reporting hold. This option also enables min noise propagation for capture clocks in max paths for setup, thereby speeding up the capture clock timing. This options is disabled by default and can only be enabled when -delta_delay is set to "true".
- static_noise_threshold_above_low *threshold_value*
Specifies above low static noise threshold, measured in relative units, as the portion of Vdd. The actual absolute constraint used is obtained as the product of this threshold and Vdd.

`-static_noise_threshold_below_high threshold_value`
 Specifies below high static noise threshold, measured in relative units, as the portion of Vdd. The actual absolute constraint used is obtained as the product of this threshold and Vdd.

`-route_xtalk_prevention true | false`
 Specifies track assign xtalk prevention. This options is disabled by default.

`-route_xtalk_prevention_threshold threshold_value`
 Specifies the threshold during track assign xtalk prevention, measured in relative units, as the portion of Vdd.

`-analysis_effort low | medium`
 Specifies the effort level of the xtalk or noise analysis. The low effort runs faster than the medium effort. For better accuracy, the medium effort should be used. The default is low effort.

`-max_transition_mode normal_slew | total_slew`
 Specifies whether delta slew is included when estimating max transition violations.
 If `total_slew` is selected, the delta slew is added in maximum transition violation computations. If `normal_slew` is selected, the delta slew is not added.
 The default is `normal_slew`.

`-reselect true | false`
 Specifies whether net reselection is enabled in timing window overlapping analysis. This option is only effective when '`-timing_window`' option is true.

DESCRIPTION

The **set_si_options** command defines signal integrity options used for analysis or optimization.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes all annotated delays, specifies crosstalk delta delay and static noise calculation and defines the threshold.

```
prompt> set_si_options -delta_delay true -static_noise true -
static_noise_threshold_below_high 0.3
```

SEE ALSO

`report_timing(2)`
`report_si_options(2)`

set_size_only

Sets a list of attributes on specified leaf cells so that they can be sized only in optimization during compile.

SYNTAX

```
int set_size_only  
[-all_instances]  
object_list  
flag
```

Data Types

<i>object_list</i>	list
<i>flag</i>	Boolean

ARGUMENTS

-all_instances

Specifies that a cell in the *object_list* is an instance whose parent cell is not unique. All similar instance leaf cells under the parent design will automatically acquire the **size_only** attribute. This is especially useful for XG mode, since the *current design* does not have to be changed in order to issue this command in different designs.

object_list

Specifies a list of leaf cell instances on which the attributes is to be set.

flag

Determines the value to which the **size_only** attributes are to be set. The valid values are **true** or **false**. When you enter **set_size_only U1**, it is the same as entering **set_size_only U1 true**. By default, **true** is assumed.

DESCRIPTION

This command places the attributes **local_optz_off**, **const_prop_off**, **del_unloaded_gate_off**, and **resyn_off** on the specified cells, and sets their values to **true** or **false**. Setting the value to **true** allows only sizing optimizations to be performed on these cells during compile. A combination of these four attributes decides whether a particular cell is size only or not. Setting the value to **false** resets it to its default value.

To remove the **size_only** attributes, set **size_only** to **false**.

When the **set_size_only** command is issued with the **true** value on a cell, optimization of that part is treated specially. In general, this command directs synthesis to perform only a sizing operation on a cell. The **set_size_only** command can be used only on leaf cell instances.

NOTE: The **false** value on the **local_optz_off**, **const_prop_off**, **del_unloaded_gate_off**, and **resyn_off** attributes could be overridden by the tool implied **size_only** attribute

when necessary. Use the **report_cell** command to see if the cell is size only, only or not.

For example, if you flag a cell instance as **size_only**, the **compile** command attempts to optimize this cell by replacing it with a cell of the same function.

Setting **size_only** on a cell does not prevent changes to the net driven by this cell. For example buffers may be added to this net. To prevent changes to the net, it should be marked **dont_touch** using **set_dont_touch** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets the attributes **local_optz_off**, **const_prop_off**, **del_unloaded_gate_off**, and **resyn_off** to **true** on a cell named *U1*.

```
prompt> set_size_only U1
```

The following example sets the attributes **local_optz_off**, **const_prop_off**, **del_unloaded_gate_off**, and **resyn_off** to **false** on a cell named *U1*.

```
prompt> set_size_only U1 false
```

The following example uses the **-all_instances** option.

```
prompt> set_size_only -all_instances mid1/bot1/c1
Information: Set size_only for all instances of cell 'c1'
in subdesign 'bot'. (UID-193)
1
```

SEE ALSO

```
get_attribute(2)
report_attribute(2)
report_cell(2)
```

set_skew_group

Creates a new user-defined skew group.

SYNTAX

```
set_skew_group  
[-name skew_group_name]  
[-target_skew desired_skew]  
[-target_early_delay minimum_insertion_delay]  
list_of_sink_pin or net_name
```

Data Types

<i>skew_group_name</i>	String
<i>desired_skew</i>	float
<i>minimum_insertion_delay</i>	float

ARGUMENTS

DESCRIPTION

This command creates a user-defined skew group. Sink pins specified in a skew group will be balanced against each other to meet the specified desired skew and target early delay.

Incremental addition of sink pins in a skew group definition is not supported.

EXAMPLES

The following example creates a skew group named grp1 with reg_g2/D1/CK and reg_g2/D2/CK as sink pins:

```
prompt> set_skew_group -name grp1 {reg_g2/D1/CK reg_g2/D2/CK}
```

SEE ALSO

```
report_skew_group(2)  
remove_skew_group(2)
```


set_spacing_label_rule

Sets intercell spacing constraint between reference cells that have been assigned labels with `set_lib_cell_spacing_label` command.

SYNTAX

```
status set_spacing_label_rule  
-labels {list_of_label_names}  
{min max}
```

Data Types

list_of_label_names list of strings

ARGUMENTS

```
-labels {list_of_label_names}  
    Lists inter-cell constraint labels. The list must contain exactly two labels  
    previously specified by the set_lib_cell_spacing_label command.  
  
{min max}  
    Specifies illegal spacing range given in units of sites, where  $\text{max} \geq \text{min}$ .
```

DESCRIPTION

This command assigns inter-cell spacing constraints between reference cells that have been assigned labels with the `set_lib_cell_spacing_command`. Cells with the given labels are prohibited from being placed adjacent to one another within the range given by *min* and *max*. The range is specified in units of placement sites. Given two adjacent standard cells with an empty space in between, each label from the right side of the left cell is checked against each label from the left side of the right cell to see if any spacing rules are violated. Such pair wise checking of labels is applied to all adjacent cells to ensure placement legality of a chip.

A label typically represents some layout features inside a standard cell. A label is associated with the left or right boundary of a standard cell in the North orientation when specified. When a cell is flipped, its associated labels will also be flipped.

All spacing rules specified by `set_spacing_label_rule` commands need to be satisfied to produce a legal placement. With inter-cell spacing rule specified, it is no longer assumed that any two standard cells can be placed next to each other with a certain spacing.

The labels and spacing rules can be used to express spacing requirements that originate from mask rules and the layout of standard cells. The labels and spacing rules are stored in the library and will be applied to all designs of the library once specified.

Built-in label 'SNPS_BOUNDARY' may be used to specify spacing rules between library cells and various types of placement boundaries: chip boundary, hard macro edge,

hard macro keepout margin, hard placement blockage edge, or voltage area guard bands.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

In the following example, cells with labels "X" and "Y" are prohibited to abut (illegal spacing of 0).

```
prompt> set_spacing_label_rule -labels {X Y} {0 0}
```

In the following example, cells with labels "X" are prohibited to be spaced within range of 3 to 5 placement sites.

```
prompt> set_spacing_label_rule -labels {X X} {3 5}
```

In the following example, cells with labels "X" are prohibited to be spaced within range 0 to 1 placement sites from placement boundary.

```
prompt> set_spacing_label_rule -labels {X SNPS_BOUNDARY} {0 1}
```

SEE ALSO

```
set_lib_cell_spacing_label(2)  
report_spacing_rules(2)  
remove_all_spacing_rules(2)
```

set_split_clock_gates_options

Set options for the **split_clock_gates** command.

SYNTAX

```
status set_split_clock_gates_options  
[-slack_margin margin_value]  
[-honor_dont_touch]  
[-honor_size_only]
```

ARGUMENTS

-slack_margin *margin_value*

Specifies the timing margin for timing slack. When the timing slack on the enable pin of an integrated clock-gating (ICG) cell is less than the specified nonzero value, it is seen as a timing violation. By default, if the timing slack on the enable pin is a positive value or zero, the ICG will not be replicated (split).

-honor_dont_touch

By default, **split_clock_gates** can replicate ICGs that have the **dont_touch** attribute. When you set this option, such ICGs will not be replicated.

-honor_size_only

By default, **split_clock_gates** can replicate ICGs that have the **size_only** attribute. When you set this option, such ICGs will not be replicated.

DESCRIPTION

This command sets the options for **split_clock_gates**.

EXAMPLES

The following example shows that **split_clock_gates** will replicate any clock gates with a negative slack of 0.05 or worse if the clock gates drive buffer trees.

```
prompt> set_split_clock_gates_options -slack_margin -0.05  
prompt> split_clock_gates
```

SEE ALSO

```
split_clock_gates(2)  
reset_split_clock_gates_options(2)  
report_split_clock_gates_options(2)
```

set_starrcxt_options

Sets up StarRC options for the **signoff_opt** command.

SYNTAX

```
status set_starrcxt_options
[-default]
[-exec_dir string]
[-max_nxtgrd_file string]
[-min_nxtgrd_file string]
[-max_image string]
[-min_image string]
[-map_file string]
[-num_parts integer]
[-mode 100 | 150 | 200 | 400]
[-option_file string]
```

ARGUMENTS

[-default]
Resets all options to the default.

[-exec_dir *string*]
Specifies the UNIX path to the StarRC executable StarXtract. The path can be either relative or complete.

[-max_nxtgrd_file *string*]
Specifies the StarRC technology file (nxtgrd) for max corner extraction.

[-min_nxtgrd_file *string*]
Specifies the StarRC technology file (nxtgrd) for min corner extraction. This argument is optional.

[-max_image *string*]

[-min_image *string*]
Specifies the run directory from a previous **StarRC** run. The **signoff_opt** command loads the run directory as max/min corner, and continues optimization. Because **StarRC** was run standalone, you must maintain the netlist consistency between the **StarRC** run and **IC Compiler**. An inconsistent netlist, for example, StarRC's netlist has one more buffer than IC Compiler, can lead to unexpected results when running **signoff_opt**.

[-map_file *string*]
Specifies the StarRC layer mapping file.

[-num_parts *integer*]
Specifies the number of partitions to run **StarRC** with distributed processing capability. The default is NOT to use **StarRC**'s distributed processing capability.

`[-mode 100 | 150 | 200 | 400]`

Specifies the extraction mode used by **StarRC**. Default is 200.

`[-option_file string]`

Specifies an option file that is used by **StarRC**. The option file contains a list of options that are used in the command file.

DESCRIPTION

The **set_starrcxt_options** command sets up StarRC options for **signoff_opt**. Use **report_starrcxt_options** command to examine the options.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example runs the **set_starrcxt_options** command.

```
prompt> open_mw_cel -lib my_lib my_cel
prompt> set_starrcxt_options -exec_dir /StarRCXT/linux/bin
      -max_nxtgrd_file /design/max.nxtgrd -map_file /design/map_file
prompt> save_mw_cel
```

SEE ALSO

report_starrcxt_options(2)
signoff_opt(2)
place_opt(2)
clock_opt(2)
route_opt(2)

set_switching_activity

Sets switching activity annotation on nets, pins, ports and cells of the current design.

SYNTAX

```
int set_switching_activity
[-static_probability sp_value]
[-toggle_rate tr_value]
[-state_dep state_condition]
[-path_dep path_sources]
[-rise_ratio ratio_value]
[-period period_value | -clock clock_name]
[-select select_types]
[-hier]
[-instances instances]
[object_list]
[-verbose]
```

Data Types

<i>sp_value</i>	float
<i>tr_value</i>	float
<i>state_condition</i>	string
<i>path_sources</i>	list
<i>ratio_value</i>	float
<i>period_value</i>	float
<i>clock_name</i>	string
<i>select_types</i>	list
<i>instances</i>	list
<i>object_list</i>	list

ARGUMENTS

- static_probability** *sp_value*
Specifies the static probability value. The static probability is a floating point number between 0.0 and 1.0 and represents the percentage of the time the signal is at the logic state 1. For example, a static probability value of 0.25 indicates that the signal is in the logic state 1 for 25% of the time.
- toggle_rate** *tr_value*
Specifies the toggle rate value. The toggle rate is a positive floating point number that represents the number of 0->1 and 1->0 transitions that the signal makes during a period of time. The period is by default 1 units and can be specified with the **-period** option. The time units used are the main library time units. Alternatively, a related clock can be annotated using the **-clock** argument, and the specified toggle rate will be relative to the related clock period.
- state_dep** *state_condition*
Specifies the state condition when annotating state-dependent toggle rates on pins or state-dependent static probabilities on cells. State dependent

toggle rates can be annotated when the internal power of the library cell pin is characterised with state-dependent power tables. State-dependent static probabilities can be annotated when the cell leakage power is characterised with state-dependent power tables. The state condition specified with this argument must be logically equivalent to a state condition in the internal / leakage power characterization.

-path_dep *path_sources*

Specifies the path sources when annotating path-dependent toggle rates on pins. This can be used when the library cell pin has path-dependent internal power characterization. The path source(s) specified with this argument must be the same as those in the internal power characterization.

-rise_ratio *ratio_value*

Specifies the ratio of rise transitions to total transitions for the specified toggle rate when annotating pins that are characterised with both rise and fall internal power. The *ratio_value* argument is a floating point number between 0.0 (all transitions are falling) and 1.0 (all transitions are rising). You need to specify a toggle rate in order to use this option. The default value is 0.5.

-period *period_value*

Specifies the time period for which the number of transitions given in the toggle rate *tr_value* occur; The time units for this value are those specified in the main technology library. When this argument is used, the *tr_value* specified by the **-toggle_rate** argument is divided by the given *period_value*; The resulting toggle rate is then annotated to the object(s) given to the **set_switching_activity** command. If the **-period** argument is not specified, a default *period_value* of 1.0 is assumed.

-clock *clock_name*

Specifies a clock to which *tr_value* is related. This clock is referred to as the object's related clock. When specified, the related clock is annotated to the design object(s), and during power calculations the annotated toggle rate value is divided by the related clock period. The resulting toggle rate is then used for calculating power. The clock period can be changed between different runs of **report_power** and the resulting change in the toggle rate of the design objects will be used automatically without re-annotating the switching activity. The user can also specify **"*"** as the *clock_name* value of the **-clock** argument. This specifies that Power Compiler will infer the related clock on the design object automatically before power calculations. The mechanism used to infer related clocks is described in the man page of **propagate_switching_activity**.

-select *select_types*

Specifies a list of object types that will be used for implicitly selecting the objects that will be annotated with the specified switching activity information. The *select_types* argument can be a list of one or more of the following object types: **regs** (sequential cell outputs), **tris** (tristate cell outputs), **black_boxes** (black box cell outputs), **inputs** (input design ports / hierarchical instance pins), **outputs** (output design ports / hierarchical instance pins), **inout** (inout design ports / hierarchical instance pins), **ports** (design port / hierarchical instance pins), **nets** (nets), **regs_on_clocks** *clock_list* (outputs of flip-flops clocked by the clocks in *clock_list*). When the **-select** argument is used all the objects in the current instance (current

design, if no current instance is set) that satisfy the selection criteria will be annotated with the specified switching activity. If the **regs** or **regs_on_clocks** selection type is used, both the non-inverting (Q) and inverting (QN) sequential cell outputs are annotated. The non-inverting outputs are annotated with the specified toggle rate and static probability. The inverting outputs are annotated with the specified toggle rate. The annotated static probability on the inverting outputs is $1.0 - sp\_val$ where *sp_val* is the specified static probability. The **regs_on_clocks** selection type needs the argument *clock_list* which is a list of clock names. This selection type selects all the registers that are clocked by a clock whose name is in *clock_list*. A register is clocked by a clock *clk* if the clock pin of the register is in the transitive fanout of the source port/pin of the clock *clk*. The **regs_on_clock** selection type can only be used in XG mode. Note that although the **regs** selection type applies to both flip-flops and latches, the **regs_on_clock** selection type applies only to flip-flops. Both the **regs** and **regs_on_clocks** selection types exclude clock-gating logic cells, which may have internal latches and are therefore sequential cells. Note that the *object_list* argument for explicitly specifying the objects to be annotated, and the **-select** argument for implicitly specifying the objects to be annotated, are mutually exclusive.

-hier

Used with the **-select** argument and specifies that all the objects in all the hierarchy that satisfy the selection criteria will be annotated. If not specified only the top level objects in the current instance will be considered.

-instances *instances*

Used with the **-select** argument and specifies that all the objects in the given list of instances that satisfy the selection criteria will be annotated. This option can also be used with the **-hier** option.

object_list

Specifies a list of nets, pins, ports or cells in the current design on which the **static_probability** and **toggle_rate** switching activity values are to be set.

Note that the *object_list* argument for explicitly specifying the objects to be annotated, and the **-select** argument for implicitly specifying the objects to be annotated, are mutually exclusive.

-verbose

Provides a more verbose output when switching activity is annotated.

DESCRIPTION

This command can be used to annotate design nets, ports, pins and cells with the different kinds of switching activity. These include simple toggle rate and static probability on nets, ports and pins; state and path dependent toggle rates on cell pins, and state dependent static probabilities on cells.

Toggle rates and static probabilities are annotated using the **-toggle_rate** and **-static_probability** arguments respectively. The toggle rate and static probability can be made state dependent by specifying the state condition with the **-state_dep** argument. The toggle rate can be made path dependent by specifying the path

source(s) with the **-path_dep** argument. A related clock can be specified using the **-clock** argument. The annotated toggle rate is divided by the related clock period during power calculation to calculate the effective toggle rate. When none of the **-toggle_rate**, **-static_probability** and **-clock** arguments are given, then annotated switching activity is removed from the specified objects.

The design objects that will be annotated with switching activity can be specified explicitly as a list of objects. The objects can also be specified implicitly by the **-select**, **-hier** and **-instance** arguments.

For statistics on the switching activity annotation on the current design use the **report_saif** command.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following **set_switching_activity** command annotates a simple toggle rate value of $33 / 1320 = 0.025$ and a static probability value of 0.015 to all design input ports.

```
prompt> set_switching_activity -toggle_rate 33 -period 1320 -  
static_probability 0.015 [get_inputs]
```

The following example annotates a toggle rate value of 0.25 and a static probability value of 0.015 to all design input ports. The clock CLK is specified as the related clock for all inputs.

```
prompt> create_clock CLK -period 10  
prompt> set_switching_activity -toggle_rate 0.25 -clock CLK -  
static_probability 0.015 -select inputs
```

When power is calculated, the annotated toggle rate value 0.25 is divided by the related clock period (10) and the resulting toggle rate (0.025) is used.

The following example annotates state dependent static probabilities on the cell orl:

```
prompt> set_switching_activity -static_probability 0.10 -state "A & B"  
[get_cell orl]  
prompt> set_switching_activity -static_probability 0.35 -state "A & ! B"  
[get_cell orl]  
prompt> set_switching_activity -static_probability 0.25 -state "! A & B"  
[get_cell orl]  
prompt> set_switching_activity -static_probability 0.30 -state "! A & ! B"  
[get_cell orl]
```

The following example annotates simple and path dependent toggle rates on the output pin Y of the cell xor1:

```
prompt> set_switching_activity -toggle_rate 0.022 [get_pin xor1/Y]  
prompt> set_switching_activity -toggle_rate 0.020 -path "A" [get_pin xor1/Y]
```

```
prompt> set_switching_activity -toggle_rate 0.002 -path "B" [get_pin xor1/Y]
```

The following example removes the switching annotation from all sequential cell outputs in the current design:

```
prompt> set_switching_activity -select regs -hier
```

The following example annotates switching activity on all the registers in the current design that are clocked by the clock clk1 or clk2.

```
prompt> set_switching_activity -toggle_rate 0.1 -static_probability 0.5 -  
select {regs_on_clock {clk1 clk2}} -hier
```

The following example annotates all primary inputs with a toggle rate of 0.2, a static probability of 0.5 and a related_clock attribute with value "*". When power is calculated during the **report_power** command, the related clocks on the design inputs are inferred automatically and the toggle rate value of 0.2 is scaled by the related clock period. The man page of the **propagate_switching_activity** command gives more details on the mechanism used to infer related clocks.

```
prompt> set_switching_activity -toggle_rate 0.2 -static_probability 0.5 -  
clock "*" -select inputs prompt> report_power
```

SEE ALSO

```
reset_switching_activity(2)  
report_saif(2)  
propagate_switching_activity(2)  
report_power(2)  
create_clock(2)
```

set_synlib_dont_get_license

Specifies a list of synthetic library part licenses that are not automatically checked out.

SYNTAX

```
int set_synlib_dont_get_license  
license_list
```

ARGUMENTS

license_list
Lists synthetic library part licenses that are not automatically checked out.

DESCRIPTION

Specifies a list of synthetic library part licenses that are not automatically checked out. By default, all synthetic library part licenses are automatically checked out if there is a possibility that they will be used. Add licenses to this list if they should not be automatically checked out when only the possibility of their use exists. The exception that licenses on this list may be checked out is, but only when read a design that **required** to have these licenses, or when manually requested via the **get_license** command. To determine what licenses are required for reading a design, use freport_design command.

If all licenses in the list present in the key file, the command sets synlib variable fsynlib_dont_get_license to be the same list of licenses. To determine the current value of fsynlib_dont_get_license, type **list synlib_dont_get_license**.

The **set_synlib_dont_get_license** command fails if any of the license key is mistyped in the command or not present in the key file.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example issues an error message because the license key is mistyped in the command or not present in the key file.

```
prompt> set_synlib_dont_get_license \  
        {"Synlib_advMath" "Synlib_ALU"}
```

Warning: Unrecognized key 'Synlib_advMath' found in 'synlib_set_dont_get_license' command. (UISN-30)

Warning: Unrecognized key 'Synlib_ALU' found in 'synlib_set_dont_get_license' command. (UISN-30) 0

The following example specifies that licenses named DesignWare-Foundation-Ultra and

SynLib-AdvMath are not to be automatically checked out.

```
prompt> set_synlib_dont_get_license \  
        {DesignWare-Foundation-Ultra SynLib-AdvMath}  
  
{"DesignWare-Foundation-Ultra", "SynLib-AdvMath"}  
1
```

SEE ALSO

```
get_license(2)  
report_design(2)
```

set_target_library_subset

Restricts optimization of a block to use a given subset of the target_library, for cases where operating condition alone is not sufficient to define the subset.

SYNTAX

```
int set_target_library_subset  
[-object_list cells]  
[-top]  
library_list  
[-milkyway_reflibs milkyway_reflib_paths]
```

ARGUMENTS

-object_list *cells*

Specifies the cells on which to set the target library subset. The cells should be instances of hierarchical designs, and the subset restriction applies to those instances and their children. Target library subset does not apply onto leaf level cells. You must specify at least one from **-top** and **-object_list**. If neither **-top** nor **-object_list** is specified, nothing is applied.

-top

If specified, the target library subset is set on the root design. You must specify at least one from **-top** and **-object_list**. If neither **-top** nor **-object_list** is specified, nothing is applied.

library_list

The list of library filenames to use for optimization of the identified design instances. The list of libraries must be a subset of the libraries specified in \$target_library. This option is a required option.

-milkyway_reflibs *milkyway_reflib_paths*

Specifies a list of Milkyway reference library paths to associate with the provided instance objects. When specifying more than one reference library, enclose them in braces ({}). The library list specified here must be a subset of the designs reference library list. For DC, we don't do any checking on this option, DC just preserve this option through nid (mw database) and **write_script** command.

DESCRIPTION

Sets a subset of the target_library which may be used for optimization of the given instances. Normally, optimization will select any lib_cell which is suitable for the local operating conditions. This command is intended only for situations where operating conditions alone are not sufficient to define which lib_cells to use in optimization.

The library_list must be a subset of the libraries listed in \$target_library.

The subset applies to the identified instances, and their children. Another subset

at a lower level completely overrides any subset specified at a higher level.

The subset restriction only applies to new cells that are created or mapped during optimization. It does not affect any cells that are already mapped. This allows changes to be restricted without affecting the unchanged portions of the design.

To remove a `target_library_subset` from the design or a design instance, use **`remove_target_library_subset`** command or use **`reset_design`**.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets the `target_library` for the design to the full set of libraries "lib1.db lib2.db", but restricts the `lib_cells` used in block u1 to those in library lib2.db.

```
prompt> set target_library "lib1.db lib2.db" prompt>
set_target_library_subset "lib2.db" \ -object_list [get_cells u1]
```

SEE ALSO

```
remove_target_library_subset(2)
report_target_library_subset(2)
check_target_library_subset(2)
set_operating_conditions(2)
target_library(3)
```

set_timing_derate

Sets derate factors on the current design or specified objects. Derate factors specify upper and lower limits on delays for a particular operating condition.

SYNTAX

```
status set_timing_derate
[-rise]
[-fall]
[-min]
[-max]
[-early]
[-late]
[-clock]
[-data]
[-net_delay]
[-cell_delay]
[-cell_check]
value
object_list
```

Data Types

<i>value</i>	float
<i>object_list</i>	list

ARGUMENTS

- rise** Indicates that the derate_value specified should be applied to the delays of paths having a rising transition at the specified objects. If neither the **-rise** or **-fall** option is specified, the derate value is applied to both rise and fall.
- fall** Indicates that the derate_value specified should be applied to the delays of paths having a falling transition at the specified objects. If neither the **-rise** or **-fall** option is specified, the derate value is applied to both rise and fall.
- min** Specifies that the derate value is set for minimum operating condition. If neither the **-min** or **-max** option is specified, the derate value is applied to both operating conditions.
- max** Specifies that the derate value is set for maximum operating condition. If neither the **-min** or **-max** option is specified, the derate value is applied to both operating conditions.
- early** Specifies that the value is the minimum derate factor. This factor defines

how early the signal could arrive. If neither the **-early** or **-late** option is specified, the derate value is applied to both.

-late

Specifies that the value is the maximum derate factor. This factor defines how late the signal could arrive. If neither the **-early** or **-late** option is specified, the derate value is applied to both.

-clock

Indicates that the derate value is to apply to elements on the clock network only. If neither the **-clock** or **-data** option is specified, the derate value is applied to both clock path and data path.

-data

Indicates that the derate value is to apply to elements on the data network only. If neither the **-clock** or **-data** option is specified, the derate value is applied to both clock path and data path.

-net_delay

Applies the derate value to net delay, which is the delay from the driver to the receiver of each net. This command cannot be used with an object list. There is only one set of derate values for net delays and they apply to the whole design.

-cell_delay

Applies the derate value to cell delay, which is the delay from the input to the output of a cell.

-cell_check

Applies the derate value to the setup and hold time requirements of cells: cell setup and cell recovery times for late derating or cell hold and cell removal times for early derating.

value

Specifies the derate value. It has to be in the range of 0.1 and 2.0.

object_list

Specifies a list of the names of leaf cells, instances or library cells for which the derate factor is to be set.

DESCRIPTION

The `set_timing_derate` command sets derate factors for the current design or specified objects. If timing derate factors are specified, some or all path delays are multiplied by derate factors. Derate factors define lower and upper ranges of timing for a certain operating condition. The derate factor specified with the **-late** option multiplies data paths delays for maximum delay (setup) check and clock paths delays for minimum delay (hold) check. The derate factor specified with the **-early** option multiplies data paths for hold checks and clock paths for setup checks.

The options **-net_delay**, **-cell_delay** and **- cell_check** apply the specified derate value to net delays, cell delays, or cell setup/hold time requirements, respectively. The **-net_delay** option can apply to the whole design only, so it cannot be used with an object list.

`set_timing_derate`

2818

If none of these three options is used and an object list is provided, the derate value applies to cell delay only; or if no object list is provided, the value applies to both cell delay and net delay.

Setup timing check is calculated using maximum operating condition. Hold timing check is calculated using minimum operating condition.

Input delay and ideal clock network latency is not derated.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example specifies an early derate value of 0.9 and a late derate value of 1.1 for all clock network nets in maximum operating condition.

```
prompt> set_timing_derate -max -early -net_delay 0.9  
prompt> set_timing_derate -max -late -net_delay 1.1
```

The next example specifies an early derate of 0.8 and a late derate value of 1.2 for leaf cell U1 for maximum operating condition.

```
prompt> set_timing_derate -max -early 0.8 U1  
prompt> set_timing_derate -max -late 1.2 U1
```

SEE ALSO

```
report_timing_derate(2)  
report_timing(2)
```

set_timing_ranges

Sets timing ranges for the current design.

SYNTAX

```
int set_timing_ranges  
[timing_ranges]  
[-library library_name]
```

Data Types

library_name string

ARGUMENTS

timing_ranges
One or two timing ranges to be applied to the current design.

-library *library_name*
Specifies the name of the library that contains definitions of the timing ranges used.

DESCRIPTION

Specifies the names of one or two timing ranges to be used for timing the current design. The specified timing ranges must be defined in *library_name*, or in one of the libraries in the **link_library**. If a **local_link_library** is set on the current design, it is prepended to (added to the beginning of) the **link_library** before the **link_library** is searched. That is, the order for a library search is:

1. *library_name*
2. **local_link_library**
3. **link_library**

Timing ranges define scaling factors that are used to scale timing path totals. You can model operating condition variations using timing ranges, by defining a range of relative slow and fast times.

The *slowest_factor* is the largest value of any of the specified timing ranges. The *fastest_factor* is the smallest value of any of the specified timing ranges. Maximum delays of data paths are scaled by the *slowest_factor*. Minimum delays of data paths are scaled by the *fastest_factor*. Timing ranges do not affect the clock network. The effect of this command can be seen with the **report_timing**, **report_constraint**, and **compile** commands. Timing ranges affect path delays for reports, such as **report_timing** and **report_constraints**, and affect delay cost computation during optimization.

The following is an example of a timing range definition, as it appears in the source text of a library:

```
timing_range("BEST_CASE") {     faster_factor : 0.95
```

```

        slower_factor : 0.99
    }

```

The name of this timing range is "BEST_CASE". The fields are defined as follows:

faster_factor

A floating-point number by which arrival times are scaled to model faster times due to environmental variations.

slower_factor

A floating-point number by which arrival times are scaled to model slower times due to environmental variations.

The **report_lib -timing_arcs** command shows the timing ranges that are defined in the technology library.

Each time the **set_timing_ranges** command is used, it overwrites the previous specifications. To remove the timing ranges defined for the current design, use the **set_timing_ranges** command with no parameters, or use the **reset_design** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

To check a library for timing ranges, use the following command:

```

prompt> report_lib test_lib
.
.
.
Timing ranges:
Name           Library      slower_factor  faster_factor
-----
BEST_CASE      test_lib      0.98          0.96
WORST_CAS      test_lib      1.1           1.0

```

In the following example, timing ranges from "my_lib.db" for the current design are selected:

```

prompt> read my_lib.db
prompt> set_timing_ranges WORST_CASE -library my_lib.db

```

The following example specifies two timing ranges for the current design if the **link_library** is "other_lib.db" and there is no **local_link_library** set on the current design:

```

prompt> set_timing_ranges {WORST_CASE BEST_CASE}

```

SEE ALSO

`current_design(2)`
`report_constraint(2)`
`report_lib(2)`
`report_timing(2)`
`reset_design(2)`
`set_local_link_library(2)`
`link_library(3)`

set_tlu_plus_files

Sets the files used for TLUPlus extraction.

SYNTAX

```
int set_tlu_plus_files  
[-max_tluplus max_tluplus_string]  
[-min_tluplus min_tluplus_string]  
[-max_emulation_tluplus max_emul_string]  
[-min_emulation_tluplus min_emul_string]  
[-tech2itf_map mapping_file]
```

Data Types

<i>max_tluplus_string</i>	string
<i>min_tluplus_string</i>	string
<i>max_emul_string</i>	string
<i>min_emul_string</i>	string
<i>mapping_file</i>	string

ARGUMENTS

-max_tluplus *max_tluplus_string*
Specifies the TLUPlus file used for maximum condition calculation of resistance and capacitance using TLUPlus.

-min_tluplus *min_tluplus_string*
Specifies the TLUPlus file used for minimum condition calculation of resistance and capacitance using TLUPlus.

-max_emulation_tluplus *max_emul_string*
Specifies the emulation TLUPlus file used for maximum condition calculation of resistance and capacitance using TLUPlus.

-min_emulation_tluplus *min_emul_string*
Specifies the emulation TLUPlus file used for minimum condition calculation of resistance and capacitance using TLUPlus.

-tech2itf_map *mapping_file*
Specifies the layer name mapping file between the technology library and the Interconnect Technology Format (ITF) file. This switch is optional only if the names are the same.

DESCRIPTION

This command specifies the files used for virtual route and postroute extraction using TLUPlus. At a minimum, you must specify the maximum TLUPlus file and the mapping file.

The **-min_tluplus** option is optional and, if specified, the command runs an extraction for the minimum condition, as well.

The **-tech2itf_map** option maps from layer names in the technology library to the layer names in the Interconnect Technology Format (ITF) file. It should contain at least two sections, `conducting_layers` and `via_layers`; `conducting_layers` maps a `routing_layer` to a conductor layer in itf, and `via_layers` maps a `contact_layer` to a via layer in ITF. If the names between your technology library and ITF file are the same, you do not have to specify this option. If you do not specify this option, the tool assumes that the names are the same. Thus, if the names are actually different, the tool will error out later.

You can also use this command to define two (optional) emulation TLUPlus files, which are used to perform emulation metal fill extraction. Emulation TLUPlus files include metal fill-related information, such as `FILL_RATIO`, `FILL_SPACING`, `FILL_WIDTH`, and `FILL_TYPE`.

If you use the same names in the different directories for max and min `tlu_plus_file`, then they will be considered to be the same file. The file names need to be different if the files are not the same.

Note that tlu plus files are stored as physical attributes of the design. When `import_design` is used, tlu plus files setting must come after `import_design`. Design without floorplan is considered logical design. When open a logical design, only logical attributes are reloaded. It is important to do `initialize_floorplan` or `read_def` with valid floorplan def data after tlu plus file setting before save, to insure when reopen the physical attributes are reloaded.

Multicorner-Multimode Support

This command uses information from the current scenario only. If using emulation, it needs to be set accross all scenarios.

EXAMPLES

The following example has, in the physical library:

```
routing_layer ( "METAL1" ) { ...
}
contact_layer ( "VIA1" );
...
```

and has in the Interconnect Technology Format (ITF) file:

```
CONDUCTOR metal1 {THICKNESS=0.53 WMIN=0.23 SMIN=0.23 RPSQ=0.078 \
CAPACITIVE_ONLY_ETCH=0}
...
VIA via1 {FROM=metal1TO=metal2 AREA=0.0169 RPV=1.6}
```

Thus, the mapping file should look like the following:

```
conducting_layers
METAL1 metal1
```

```
...
via_layers
VIA1 via1
...
```

The following example has the same ITF file in the first example, so the mapping file should be the same as in the first example, too, if you have the following information in the Interconnect Technology Format (ITF) file:

```
Layer "METAL1" {  maskName = "metal1"
  ...
}
Layer "VIA1" {  maskName = "via1"
  ...
}
```

The following example runs the **set_tlu_plus_files** command.

```
prompt> set_tlu_plus_files -max_tluplus tlu.max \
        -tech2itf_map design.map
```

SEE ALSO

```
extract_rc(2)
set_extraction_options(2)
```

set_true_delay_case_analysis

Sets the `true_delay_case_analysis` attribute, which specifies the input vector value to use for specified pins or ports of the current design with the `-true` and `-justify` options of `report_timing`.

SYNTAX

```
status set_true_delay_case_analysis  
0 | 1 | r | f | none  
port_pin_list
```

Data Types

`port_pin_list` list

ARGUMENTS

0 | 1 | r | f | none
Specifies the transition value at which to set the **true_delay_case_analysis** attribute. The first 4 arguments represent transition values and **none** removes the attribute. The valid transition values are **0** (logic 0), **1** (logic 1), **r** (rise, X to 1), and **f** (fall, X to 0). For details on these values, see the `DESCRIPTION` section below.

`port_pin_list`
Specifies a list of port or pin names of the current design for which the **true_delay_case_analysis** attribute is to be set. If more than one object is specified, they must be included in braces ({}).

DESCRIPTION

The **set_true_delay_case_analysis** command sets the string attribute **true_delay_case_analysis** to 1 of 4 transition values, or removes the attribute if set to **none**. The attribute can also be removed with the **remove_attribute** command.

Case analysis is normally used on path startpoints, which are input ports and clock pins of registers. Internal pins can also be preset. They are then considered to be path startpoints (no signals propagate through them).

The `-justify` and `-true` options of **report_timing** attempt to prove that timing paths in the design are true by finding an input vector that sensitizes the paths. The **set_true_delay_case_analysis** command can be used to explicitly specify some values in the input vector. Setting a case analysis value at a pin blocks paths through that pin, so no value should be specified for pins on the path being sensitized.

By default, the true delay algorithm searches for full input vectors. However, you can preset all or part of the input vector manually using the **set_true_delay_case_analysis** command. Presetting part of the input vector can dramatically reduce the search space, which makes runtimes faster. You can also use **set_true_delay_case_analysis** to manually sensitize particular paths.

Note that **r** defines a rising value and **f** defines a falling value. These values are the X to 1 and X to 0 transitions applied at the inputs. In **report_timing -true** or **report_timing -justify**, the input vector is also reported this way, using **r** and **f** for the input transitions. On the report, an asterisk (*) identifies values that have been preset.

You can set signals to a constant value using the **0** and **1** arguments. This is different from the X to 1 and X to 0 transition that **r** and **f** imply. A constant value means that the signal is always 1 or 0.

To see the difference between presetting an **r** and presetting a **1**, consider the case of an OR gate that has an early-arriving controlling value **r** on one input, and the other input is a late-arriving preset value. Although the preset value was **r**, the earlier controlling value still propagates because the **r** is just a later arriving controlling value. However, if the preset value was a **1** then the output of the gate is a constant **1**.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following command specifies that the input vector will consider only rising transitions at port C for the purposes of **report_timing -true** or **report_timing -justify**:

```
prompt> set_true_delay_case_analysis r C
```

This example sets port A to a constant logic zero for **report_timing -true** or **report_timing -justify**:

```
prompt> set_true_delay_case_analysis 0 A
```

SEE ALSO

```
current_design(2)  
remove_attribute(2)  
report_timing(2)  
reset_design(2)
```

set_unconnected

Lists output ports to be unconnected.

SYNTAX

```
int set_unconnected  
port_list
```

Data Types

```
port_list      list
```

ARGUMENTS

```
port_list
```

A list of port names in the current design that are to be left unconnected. If more than one port is specified, they must be enclosed either in quotes or in braces ({}).

DESCRIPTION

Assigns an **output_not_used** attribute to ports in *port_list*. This information is used by **compile** to create smaller designs by eliminating logic that drives an unconnected output port and therefore might not need to be maintained during optimization. After a design with an unconnected output port is compiled, the port usually is not driven by anything inside the design.

This attribute can be removed using the **remove_attribute** command.

Note: The **set_unconnected** command cannot be used on input ports. Input ports can be tied to logic one, logic zero, or don't-care using **set_logic_one**, **set_logic_zero**, and **set_logic_dc**.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example specifies that output ports "CARRY_OUT" and "err_1" are not to be used.

```
prompt> set_unconnected {CARRY_OUT err_1}
```

SEE ALSO

```
get_attribute(2)  
remove_attribute(2)  
report_port(2)
```

```
set_unconnected  
2828
```

```
reset_design(2)
set_logic_one(2)
set_logic_zero(2)
set_logic_dc(2)
```

set_undoable_attribute

Sets an attribute to a specified value on the specified list of objects with support for undo.

SYNTAX

```
collection set_undoable_attribute
[-class class_name]
object_list
attribute_name
[-quiet]
```

Data Types

<i>class_name</i>	string
<i>object_list</i>	string_list
<i>attribute_name</i>	string

ARGUMENTS

-class *class_name*
Specifies the class name for the object specified in *object_list*, if the element of *object_list* is a name. Valid classes are design, port, cell, net, etc.

object_list
A list of objects on which the attribute is to be set. Each element in the list is either a collection or a pattern that is combined with the *class_name* to find the objects.

attribute_name
Specifies the name of the attribute to be set.

-quiet
Turns off the warning message that would otherwise be issued if the attribute or objects are not found.

RETURNS

collection
collection of objects affected

DESCRIPTION

This command sets the value of an attribute on an object. For a complete listing of attributes, refer to the attributes man page.

This command creates a collection of objects that have the specified attribute value set. A returned empty string indicates that no object has been set.

The resultant attribute value change can be undone using the undo command

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example defines an attribute named X for cells, then sets the value on all cells in this level of the hierarchy:

```
prompt> define_user_attribute -type int -class cell Xcell  
prompt> set_undoable_attribute [get_cells *] X 30{"U1"}
```

SEE ALSO

```
collections(2)  
define_user_attribute(2)  
get_attribute(2)  
list_attributes(2)  
remove_attribute(2)  
set_attribute(2)  
undo(2)
```

set_ungroup

Sets the **ungroup** attribute on specified designs, cells, or references, indicating that they are to be ungrouped during **compile**.

SYNTAX

```
int set_ungroup  
object_list true | false
```

Data Types

object_list list

ARGUMENTS

object_list
Specifies a list of cells, references, or designs to be ungrouped during **compile**.

true | false
The value with which to set the **ungroup** attribute. The default is *true*.

DESCRIPTION

Sets the **ungroup** attribute on the specified objects so that they are ungrouped (collapsed to one design level) before they are optimized. The **compile** command does not ungroup cells or designs with the **dont_touch** attribute.

This attribute is removed with the **remove_attribute** or **reset_design** commands.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

In the first example, the **ungroup** attribute is set on the cell "U1".

```
prompt> set_ungroup U1
```

If the **ungroup** attribute is specified on a reference object, cells using that reference are ungrouped during **compile**. In this example, all the instances of ADDER in the current design are ungrouped.

```
prompt> set_ungroup [get_references ADDER]
```

A design with this attribute is ungrouped whenever **compile** encounters it.

```
prompt> set_ungroup [get_designs ND17]
```

SEE ALSO

`current_design(2)`
`remove_attribute(2)`
`reset_design(2)`
`ungroup(2)`
`ungroup_keep_original_design(3)`

set_units

Sets the units used for resistance, capacitance, timing, leakage power, current and voltage. The units must be consistent with the main library units.

SYNTAX

```
string set_units
[-resistance ohm|kohm|mohm|Mohm|10ohm|100ohm]
[-capacitance f|ff|pf|nf|uf|mf|10ff|100ff]
[-time s|fs|ps|ns|us|ms|10ps|100ps]
[-power w|fw|pw|nw|uw|mw|10uw|100uw|10mw|100mw|10pw|100pw|10nw|100nw]
[-current A|fA|pA|nA|uA|mA|10uA|100uA|10mA|100mA]
[-voltage v|fv|pv|nv|uv|mv|10mv|100mv]
```

ARGUMENTS

```
-resistance
    Sets the resistance unit.

-capacitance
    Sets the capacitance unit.

-time
    Sets the time unit.

-power
    Sets the power unit.

-current
    Sets the current unit.

-voltage
    Sets the voltage unit.
```

DESCRIPTION

The **set_units** command sets the units for SDC file. In the Z-2007.03 release, the units cannot conflict with the main library units; therefore, it is not recommended to not use the **set_units** command interactively.

When the tool writes an SDC file, the **set_units** command is always written out as the first SDC command. If no units are set, the tool gets the units from the main library (the first link library).

EXAMPLES

The following example specifies the **set_units** command that is written to the SDC file.

```
prompt> set_units -time ns -resistance kohm -capacitance pf \
```

```
set_units
2834
```


-power uW -voltage V -current A <plain

SEE ALSO

report_units(2)

set_user_grid

Sets the user grid for this session.

SYNTAX

```
status set_user_grid
[-x_offset float]
[-y_offset float]
[-x_step float]
[-y_step float]
[-user_grid {{x_offset y_offset} {x_step y_step}}]
[-x_get_from_layer layer]
[-y_get_from_layer layer]
[-reset]
[design]
```

Data Types

<i>float</i>	floating-point
<i>layer</i>	collection
<i>design</i>	string

ARGUMENTS

-x_offset *float*

Specifies the horizontal (X) distance (in user units) by which you want to offset the cursor snap-to points from the grid point. The value must be multiple of litho grid. This option is exclusive with -user_grid and -x_get_from_layer option. You can only use one of them.

-y_offset *float*

Specifies the vertical (Y) distance (in user units) by which you want to offset the cursor snap-to points from the grid point. The value must be multiple of litho grid. This option is exclusive with -user_grid and -y_get_from_layer option. You can only use one of them.

-x_step *float*

Specifies the horizontal (X) distance (in user units) you want between cursor snap-to points. The value must be multiple of litho grid. This option is exclusive with -user_grid and -x_get_from_layer option. You can only use one of them.

-y_step *float*

Specifies the vertical (Y) distance (in user units) you want between cursor snap-to point. The value must be multiple of litho grid. This option is exclusive with -user_grid and y_get_from_layer option. You can only use one of them.

-user_grid {{*x_offset y_offset*} {*x_step y_step*}}

Specifies the user grid in list representation. The element value must be multiple of litho grid. This option is exclusive with all other options except *design*.

set_user_grid

2836

`-x_get_from_layer layer`

Indicate to have the tool automatically calculate the user grid `x_offset` `x_step` from the wire track on the specified layer. The value could be layer name or layer number or layer collection. This option is exclusive with `-x_offset` `-x_step` and `-user_grid` option. You can only use one of them.

`-y_get_from_layer layer`

Indicate to have the tool automatically calculate the user grid `y_offset` `y_step` from the wire track on the specified layer. The value could be layer name or layer number or layer collection. This option is exclusive with `-y_offset` `-y_step` and `-user_grid` option. You can only use one of them.

`-reset`

Reset the user grid to litho grid. This option can only be used with design option.

`design`

Specifies in which design to set the user grid information. If this option is not specified, it will set global user grid information which is used for default value of successive run of this command.

DESCRIPTION

This command set user grid for this session of the tool. You can use the `-user_grid` option to set the user grid in one shot. Or else, you can use `-x_offset`, `-y_offset`, `-x_step`, `-y_step` to specify individual part of the user grid. If any of these four options are not specified, then the corresponding value is not changed. Finally, you can use `-x_get_from_layer` and/or `-y_get_from_layer` to have the tool automatically calculate the user grid from the wire track on the given layer.

The user grid must be multiple of litho grid. If it's not, this command will fail with error message MWUI-212.

If design is not specified, then this command will set global user grid. The global user grid will be used as default value for design specific user grid until you explicitly set user grid on the design.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example set global user grid in one shot:

```
prompt> set_user_grid {{0 0} {0.010 0.010}}
1
```

The following example set `x_offset` and `y_offset` of the global user grid only:

```
prompt> set_user_grid -x_offset 0.5 -y_offset 0.6
1
```

The following example set user grid from wire track information on given design top.

```
prompt> set_user_grid -x_get_from_layer M1 -y_get_from_layer M2 top
1
```

The following example failed because the value specified is not multiple of litho grid, suppose litho grid is 0.001:

```
prompt> set_user_grid -x_offset 0.0005
Error: Value '0.0005' is not multiple of litho grid '0.001'. (MWUI-212)
0
```

SEE ALSO

`get_user_grid(2)`

set_via_array_size

Modifies the array size of an existing via or via array.

SYNTAX

```
collection set_via_array_size  
-array_size {row col}  
via_collection
```

Data Types

via_collection collection

ARGUMENTS

-array_size {row col}
Specifies the new array_size of the via/via_array. The value of *row* and *col* must be positive integer. If you specify {1 1} to a via_array, the via_array will be converted to a via. If you specify another value except {1 1} to a via, the via will be converted to a via_array.
You must specify this option.

via_collection
Specifies a list of via/via_arrays that to be assigned with new array_size value.

DESCRIPTION

The **set_via_array_size** command enables you to convert a via to a via_array and vice versa. You may also use this command to change the value of array_size of a via_array. The return value is a collection contains the modified via/via_array.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example modifies a via to a via_array with array_size {2 3}

```
prompt> set_via_array_size -array_size {2 3} VIA#4075 {VIA_ARRAY#86054}
```

SEE ALSO

```
create_via(2)  
get_vias(2)  
remove_via(2)
```

set_voltage

Applies an operating voltage on a list of supply nets objects.

SYNTAX

```
int set_voltage  
max_case_voltage  
[-min min_case_value]  
-object_list list_of_supply_nets
```

Data Types

<i>max_case_voltage</i>	float
<i>list_of_supply_nets</i>	list

ARGUMENTS

```
max_case_voltage  
    Specifies the operating voltage for the maximum (worst) case.  
  
-min min_case_value  
    Specifies the operating voltage for the minimum (best) case.  
  
-object_list list_of_supply_nets  
    Specifies the list of supply nets that will have the operating voltages as  
    specified in this command. Collection input is supported.
```

DESCRIPTION

Defines the operating voltage on the supply nets and their corresponding power nets so that the parts of the design powered by these supply nets or power nets are timed and optimized at the specified voltage. If you do not specify any operating voltage for a supply net, the part of the design connected by this supply net will continue to be timed and optimized based on the available operating condition settings.

To see the operating voltages of supply nets, use **report_supply_net**.

The operating voltage of a supply net, once defined, can only be overridden with another **set_voltage**.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example shows a typical context of use for the **set_voltage** command.

```
prompt> create_power_domain CNTL
```

```
set_voltage  
2840
```

```
1
prompt> create_supply_net sn1 -domain CNTL
1
prompt> create_supply_net sn2 -domain CNTL
1
prompt> set_voltage 0.9 -object_list {sn1 sn2}
1
```

SEE ALSO

```
create_supply_net(2)
create_power_domain(2)
set_domain_supply_net(2)
connect_supply_net(2)
report_supply_net(2)
report_power_domain(2)
```

set_voltage_model

On-The-Fly setting of the voltage model on the library specified.

SYNTAX

```
Boolean set_voltage_model  
library_name  
[-voltage {voltage_name voltage_value}]  
[-reset]
```

Data Types

<i>library_name</i>	string
<i>voltage_name</i>	string
<i>voltage_value</i>	float

ARGUMENTS

library_name
Specify the library name for the voltage definition.

-voltage {*voltage_name* *voltage_value*}
Set the voltage name and value pair for the library.

DESCRIPTION

The **set_voltage_model** command update the library voltage map. If the voltage is already defined in the library, the new value will override the old value. If the voltage is not defined, this new definition will be added into the voltage map for the library.

EXAMPLES

following example shows the setting for single rail library

```
prompt> set_voltage_model standard_lib -voltage {VDD 1.2} -voltage {VSS 0.0}
```

following example shows the setting for multiple rail library

```
prompt> set_voltage_model standard_lib -voltage {VDD 1.2} -voltage {VSS 0.0} \  
prompt> -voltage {VDD_BACK 0.7}
```

SEE ALSO

set_vt_filler_rule

Sets multiple threshold voltage filler cell insertion rules.

SYNTAX

```
integer set_vt_filler_rule  
-threshold_voltage vt_type_1_vt_type_2  
-lib_cell cell_list
```

Data Types

cell_list collection

ARGUMENTS

```
-threshold_voltage vt_type_1_vt_type_2  
    Specifies threshold voltage types to be used in a gap between two standard  
    cells of vt_type_1 and vt_type_2. The name vt_type_1 and vt_type_2 must be a  
    string previously specified by the set_cell_vt_type command. If vt_type_2 is  
    omitted, it refers to the default vt type, referring to cells that have never  
    been marked with any threshold voltage type.
```

```
-lib_cell cell_list  
    The collection cell_list refers to a collection of library cells. The  
    specified library cells will be used as fillers between a gap characterized  
    by vt_type_1 and vt_type_2.
```

DESCRIPTION

Use this command to set multiple threshold voltage filler rules of the design. To specify the rule, the command `set_cell_vt_type` command should have been used to mark the threshold voltage type of a standard cell. The insertion rule is used by the `insert_stdcell_filler` command. Only the specified filler cells will be inserted in a gap characterized by *vt_type_1* and *vt_type_2*. The relative position of the two standard cell types is not important. A gap with *vt_type_1* on the left and *vt_type_2* on the right is the same as that with the two types switched. Cells not marked with any threshold voltage type have the "default threshold voltage type" which is a valid type by itself.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example specifies a filler rule between library cells of "vtTypeA" and "vtTypeB", and uses filler cells f1, f2, or f3.

```
prompt> set_vt_filler_rule -lib_cell {f1 f2 f3} \
```

-threshold_voltage {vtTypeA vtTypeB}

SEE ALSO

insert_stdcell_filler(2)
set_cell_vt_type(2)
set_left_right_filler_rule(2)
remove_vt_filler_rule(2)
report_vt_filler_rule(2)

set_write_stream_options

Sets options for the **write_stream** command.

SYNTAX

```
status set_write_stream_options
[-reset]
[-map_layer layer_mapping_file_name]
[-rename_cell cell_renaming_file_name]
[-child_depth child_cell_depth]
[-skip_ref_lib_cells]
[-resize_text {width text_conversion_factor}]
[-flatten_via]
[-contact_prefix $$]
[-output_filling {list_of_types}]
[-output_outdated_fill]
[-output_instance_name_as_property inst_prop_value]
[-output_geometry_property]
[-remove_backslash_from_instance_net_names]
[-max_name_length max_name_length]
[-keep_data_type]
[-output_by_layer {layer_number_strings}]
[-oasis_compression_level compress_level_value]
[-compressed]
[-output_pin {output_pin_types}]
[-pin_name_mag pin_name_mag]
[-net_name_mag net_name_mag]
[-output_net {output_net_types}]
[-output_net_name_as_property net_prop_value]
[-output_polygon_pin]
[-output_design_intent]
[-design_intent_cell_name di_struct_name]
[-critical_object_names critical_inst_net_name_list]
[-map_design_intent_layers di_layer_mapping_file_name]
[-design_intent_only]
[-rotate_pin_text_by_access_dir]
[-set_hier_instance_name_long]
[-set_hier_net_name_long]
[-ignore_layer_mapping_for_imported_cells]
```

Data Types

<i>layer_mapping_file_name</i>	string
<i>cell_renaming_file_name</i>	string
<i>list_of_types</i>	list of strings
<i>di_struct_name</i>	string
<i>di_layer_mapping_file_name</i>	string

ARGUMENTS

-reset
Resets all options to their default value.

-map_layer *layer_mapping_file_name*

Specifies the file that defines the layer mapping from Milkyway to GDSII.
The layer mapping file format is

MWObjType[MWNetType][PinCode]

MWLayer[:MWDataType] GDSLayer[:GSDDataType]

Valid values for MWObjType are

* A for all types

* T for text

* D for data

Valid values for MWNetType are

* A for all types

* S for signal

* P for power

* G for ground

* C for clock

* U for up conduction layer (upper layer in a via)

* D for down conduction layer (lower layer in a via)

* X for power or ground wires or contacts with a "signal"
or "tie-off" routing type

Valid values for PinCode are

* P means that the pins on each CEL view cell being
processed, which might include child cells depending on
the value of the **-child_depth** option, are mapped
by this rule.

* T means that only the top-level pins (terminals) on the
current top-level cell are mapped by this rule.

* null

For example, to map the text associated with signal nets on Milkyway layer
10 with a data type of 20 to GDS layer 22 with a data type of 1, enter

TS 10:20 22:1

To map the geometry of METAL1 terminals to GDS layer 15:2 and map the geometry
of lower-level METAL1 pins (not terminals) to GDS layer 14:2, enter

DSP METAL1 14:2

DAT METAL1 15:2

By default, **write_stream** uses the layer information defined in the Milkyway
database.

-rename_cell *cell_renaming_file_name*

Specifies the file that contains the mapping from original cell names to new
cell names. This option is for GDS only.

The name mapping file format is

OldCellName NewCellName

For example, to map TOP.CEL to NEWTOP.CEL, enter

TOP NEWTOP

By default, the original cell names are output.

-child_depth *child_cell_depth*

Specifies the hierarchical level to output child cells. To export all child
cells, you should specify a large number, such as 20. Milkyway exports the
CEL view of each child cell to the specified level of hierarchy.

The default value is 0, meaning only the top-level cell is output.

-skip_ref_lib_cells

Skips the conversion of child cells in reference libraries. This option is
for GDS only.

set_write_stream_options

By default, **write_stream** converts all the referenced child cells from the reference libraries.

-resize_text {*width text_conversion_factor*}

Specifies the text width in user units and the conversion factor for the text height relative to its width in Milkyway.

The default value is 1.0 for *text_conversion_factor* and 0.0 for *width*.

-flatten_via

Breaks down contacts and contact arrays into component parts.

By default, **write_stream** does not flatten contacts and contact arrays.

Instead, it translates contacts as references, outputs contact geometry as cells, and creates cells with the following naming convention for a contact array: "prefix_contactName_xViaPitch_yViaPitch_xReps_yReps", where "prefix" is the prefix specified in the **-contact_prefix** option, "contactName" is the name assigned to the contact in the deviceTable section in the Milkyway technology file, "xViaPitch" is the horizontal distance in database units between centers of contacts in the array, "yViaPitch" is the vertical distance in database units between centers of contacts in the array, "xReps" is the number of columns (contacts in the horizontal direction) in the array, and "yReps" is the number of rows (contacts in the vertical direction) in the array. For example, \$\$VIA1_2000_3000_4_4.

-contact_prefix \$\$

Specifies the prefix added to contact or contact array names.

The default is "\$\$".

-output_filling {*list_of_types*}

Specifies the types of filling data to output. Valid values are

- * notch

Outputs notch data stored in the NOTC view.

- * gap

Outputs gap data stored in the GAP view.

- * fill

Outputs fill data stored in the FILL view.

You can specify multiple types.

By default, no filling data is output.

-output_outdated_fill

Forces the output of fill data even if the fill data is out-of-date.

By default, out-of-date fill data is not output.

-output_instance_name_as_property *inst_prop_value*

Specifies the instance property integer (1-127) to attach a property to each object associated with cell instances in the design.

By default, the instance property is not generated.

-output_geometry_property

Outputs a geometry's property.

By default, the property attached to a geometry is not written out.

-remove_backslash_from_instance_net_names

Removes backslashes from cell instance and net names where backslashes can be inserted. For example, a backslash can be inserted before a bus character or hierarchy character as an escape.

-max_name_length *max_name_length*

Specifies the name length limit for all output cells. If the cell name exceeds the length limit, **write_stream** truncates the cell name and resolves the new name conflict automatically. The default length is 32.

Note that this option is for library cell names only. To set a name length limit for instance names or net names, use the **-set_hier_instance_name_long** or **-set_hier_net_name_long** options.

-keep_data_type

Keeps each object's data type as defined in the Milkyway library. By default, **write_stream** converts the data type to 0.

-output_by_layer {*layer_number_strings*}

Specifies the layers for which to output the geometry and text. You specify the layers and data types to output in the following format:

{layer1[:datatype1] layer2[:datatype2] ...}

Separate different layers by a space. For example,
{3 4 5 11:10}

-oasis_compression_level *compress_level_value*

Specifies the compression level (1-9) to compress the resulting stream file. By default, **write_stream** generates the resulting stream file without compression. This option is for OASIS only.

-compressed

Compresses the resulting stream file in gzip format. By default, **write_stream** generates the resulting stream file without gzip compression. This option is for GDSII only.

-output_pin {*output_pin_types*}

Specifies the object types to output for each pin. Valid values are:

* **text**

Outputs the name of each pin as a TEXT object associated with the pin.

* **geometry**

Outputs the geometry of each pin as a POLYGON or BOUNDARY object associated with the pin.

-pin_name_mag *pin_name_mag*

Specifies the magnification for text when outputting a pin as a TEXT object. The default value is 1.0. This value takes effect only when **-output_pin {text}** is specified. This option is for GDS only.

-net_name_mag *net_name_mag*

Specifies the magnification for text when outputting a net as a TEXT object. The default value is 1.0. This value takes effect only when **-output_net {text}** is specified. This option is for GDS only.

-output_net {*output_net_types*}

Specifies the object types to output for each net. Valid values are:

* **text**

Outputs the name of each net as a TEXT object associated with the net.

* **plex**

Assigns a plex number to each net and generates a text record with the plex number. When you specify **plex**, it automatically selects **text**, as well. **plex** is GDS only.

set_write_stream_options

`-output_net_name_as_property net_prop_value`
 Specifies the net property integer (1-127) to attach a property to each object associated with cell nets in the design. By default, the net property is not generated.

`-output_polygon_pin`
 Outputs pin geometries that are not rectangles. By default, nonrectangular pin geometries are not output. This option is for GDS only.

`-output_design_intent`
 Outputs design intent data. By default, design intent data is not output, even if you specify design-intent-related options. This option is for GDS only.

`-design_intent_cell_name di_struct_name`
 Specifies the struct name to hold design intent data. Default value is "DesignIntent". This option is for GDS only.

`-critical_object_names critical_inst_net_name_list`
 Specifies the file name of the critical nets and cell instances. The critical name list is in ASCII format with each name entry on a separate line prefixed by N for net or I for instance. In each name list, net and instance names are grouped by level of criticality with the DITYPE keyword. For example,

 DITYPE CRITICAL_10_PERCENT
 N INST1/NET12
 N INST1/NET15/Q
 I INST2
 DITYPE EM_SENSITIVITY
 N INST1/NET12
 N INST2/net14/q
 I INST3

 This option is for GDS only.

`-map_design_intent_layers di_layer_mapping_file_name`
 Specifies the file name of the design intent layer mapping. The file contains the mapping between the original design layer-datatype pairs and design intent layer-datatype pairs. The layer mapping is grouped by level of criticality with DITYPE keyword. The design intent layer mapping format is N|I original_layer:original_datatype->DI_layer:DI_datatype
 For example,

 DITYPE CRITICAL_10_PERCENT
 N 31:0->31:0 ; Metal 1 critical net
 I 255:0->64:10 ; instance marking for critical design intent

 DITYPE EM_SENSITIVE_NETS
 N 31:0->31:15 ; Metal 1 EM design intent
 N 32:0->32:15 ; Metal 2 EM design Intent

 This option is for GDS only.

`-design_intent_only`
 Outputs only design intent data. By default, **write_stream** outputs original

design data along with the design intent data. This option is for GDS only.

-rotate_pin_text_by_access_dir

Rotates a pin's TEXT object according to the pin's access direction. By default, the TEXT object is not rotated. This option takes effect only when **-output_pin {text}** is specified. This option is for GDS only.

-set_hier_instance_name_long

Specifies the name length limit of the output instance name is longer than 126 characters.

-set_hier_net_name_long

Specifies the name length limit of the output net name is longer than 126 characters.

-ignore_layer_mapping_for_imported_cells

Turn this option on to ignore layer mapping for those cells created from GDSII/OASIS file. The layer number will be written exactly to GDSII/OASIS file without any conversion.

DESCRIPTION

The **set_write_stream_options** command sets the options for the **write_stream** command. The command is additive; you can run it multiple times to incrementally set the options. You can report the option values by using the **report_write_stream_options** command. The options set by **set_write_stream_options** are modal. Once set, the values are not changed until next set.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets a child cell depth of 1 and outputs gap data.

```
prompt> set_write_stream_options \  
-child_depth 1 \  
-output_filling {gap}
```

SEE ALSO

`write_stream(2)`
`report_write_stream_options(2)`

set_xtalk_route_options

Sets global route and track assignment crosstalk options.

SYNTAX

```
status set_xtalk_route_options
[-default]
[-groute_minimize_xtalk true | false]
[-groute_xtalk_weight number]
[-track_assign_minimize_xtalk true | false]
[-track_assign_noise_threshold value]
```

Data Types

<i>number</i>	integer
<i>value</i>	float

ARGUMENTS

-default
Set default options

-groute_minimize_xtalk true | false
Global route xtalk mode (default is false).

-groute_xtalk_weight *number*
Global route xtalk weight (default is 4). Range: 0 to 16.

-track_assign_minimize_xtalk true | false
Track assign xtalk mode (default is false).

-track_assign_noise_threshold *value*
Noise threshold ratio to supply voltage (default is 0.45). Range: 0 to 1.

DESCRIPTION

This command allows you to set global route and TrackAssignment xtalk options for use by subsequent router steps. This options can also be set by set_parameter commands.

Many of the options have a (true | false) syntax because it may be necessary to set the option true if it was originally set to false, or it may be necessary to set it to false if it was set previously to true.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

```
prompt> set_xtalk_route_options\  
-track_assign_minimize_xtalk true
```

SEE ALSO

`set_parameter(2)`

set_zero_interconnect_delay_mode

Forces the timer to ignore the contribution on a timing path from any wire capacitance in the design.

SYNTAX

```
status set_zero_interconnect_delay_mode  
[true | false]
```

ARGUMENTS

true | false

Enables the zero interconnect delay mode when set to **true**. Disables the zero interconnect delay mode when set to **false**. If no value is specified, the default is **true**.

DESCRIPTION

This command enables or disables the zero interconnect delay mode. The zero interconnect delay mode forces the timer to ignore contributions on a timing path from any wire capacitance in a design. It forces all wire capacitance to be as trivial as 0, regardless of the wire load model used in the design or any back-annotated capacitance on a wire. Disabling the mode allows the timer to consider the wire capacitance as it did before enabling the mode.

The command is used primarily in preplacement and postplacement steps to assess design and constraint feasibility. When the mode is enabled, the design is analyzed with only cell delay and the capacitance of the pin load on all wires in the design to determine if the design can meet timing goals. It also helps when debugging potential missing timing exceptions in the constraints. Zero interconnect delay mode must not be used in the final implementation step.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

SEE ALSO

```
get_zero_interconnect_delay_mode(2)  
report_timing(2)
```

set_zrt_net_properties

Sets Zroute properties for the specified nets.

SYNTAX

```
status set_zrt_net_properties  
[-ignore_voltage_areas true | false]  
-nets collection_of_nets  
-from_file file_name
```

Data Types

collection_of_nets collection

ARGUMENTS

`-ignore_voltage_areas true | false`
Controls whether the router should ignore voltage area constraints for the specified nets.
The default value is false.

`-nets collection_of_nets`
Specifies the list of nets to set the properties on.
You must specify one of the **-nets** or **-from_file** options.

`-from_file file_name`
Specifies the nets by name in the file.
You must specify one of the **-nets** or **-from_file** options.

DESCRIPTION

The **set_zrt_net_properties** command sets properties on the specified nets. Zroute honors the net properties as it performs routing.

EXAMPLES

The following example sets a list of nets that do not need to satisfy the voltage area constraints:

```
prompt> set_zrt_net_properties -nets {n1 n2 n3}  
          -ignore_voltage_areas true
```

SEE ALSO

get_zrt_net_properties(2)
report_zrt_net_properties(2)

setenv

Sets the value of a system environment variable.

SYNTAX

```
string setenv  
variable_name new_value
```

Data Types

<i>variable_name</i>	string
<i>new_value</i>	string

ARGUMENTS

variable_name
Names of the system environment variable to set.

new_value
Specifies the new value for the system environment variable.

DESCRIPTION

The **setenv** command sets the specified system environment *variable_name* to the *new_value* within the application. If the variable is not defined in the environment, the environment variable is created. The **setenv** command returns the new value of *variable_name*. To develop scripts that interact with the invoking shell, use **getenv** and **setenv**.

Environment variables are stored in the Tcl array variable **env**. The environment commands **getenv**, **setenv**, and **printenv** are convenience functions to interact with this array.

The **setenv** command sets the value of a variable only within the process of your current application. Child processes initiated from the application using the **exec** command after a usage of **setenv** inherit the new variable value. However, these new values are not exported to the parent process. Further, if you set an environment variable using the appropriate system command in a shell you invoke using the **exec** command, that value is not reflected in the current application.

EXAMPLES

The following example changes the default printer.

```
prompt> getenv PRINTER  
laser1  
prompt> setenv PRINTER "laser3"  
laser3  
prompt> getenv PRINTER  
laser3
```

SEE ALSO

`getenv(2)`
`printenv(2)`
`printvar(2)`

shape_fp_blocks

Automatically places and shapes plan group boundaries, black boxes, and other soft macros in a design core.

SYNTAX

```
status shape_fp_blocks
[-rectilinear]
[-incremental target_utilization_driven | congestion_driven]
[-channels]
[-refine_placement]
[-constraint_file file_name]
[-top_down]
[-sliver_threshold threshold]
[-place_submacros]
```

Data Types

<i>file_name</i>	string
<i>threshold</i>	float

ARGUMENTS

-rectilinear

If you do not select this option, the command creates only rectangular plan groups or soft macro boundaries unless they would overlap fixed objects, in which case the shape might need to be rectilinear if it needs to be cut out around the fixed objects.

If you select this option, then in cases where one small block and one large block need to be shaped close to each other, the small block is placed in one of the corners of the large block, creating a rectilinear L-shaped boundary. Therefore, L-shaped plan groups can be created even if no fixed objects are encountered on the floorplan.

The default is off.

-incremental target_utilization_driven | congestion_driven

Adjusts the floorplan to meet new target plan group and voltage area utilizations or to reduce congestion in channels between blocks or inside the blocks themselves. The command incrementally modifies the sizes, shapes, and locations of plan groups and voltage areas in the design.

If you specify the `target_utilization_driven` mode, the command tries to decrease the plan group and voltage area utilizations to be at most the sum of the corresponding target utilizations and the value of the option set with `set_fp_shaping_strategy -util_slack`. It first attempts to keep the utilizations at or below the target utilization, but will allow a slightly higher value as long as the difference is not more than the above mentioned value. Utilizations that are below the target ones are considered acceptable and therefore the command does not attempt to increase utilizations to match the target ones.

If you specify the `congestion_driven` mode, the command first attempts to use the congestion map created by global routing. If one is not available, it tries to use a placement congestion map. If neither congestion map is

available, an error message is generated. Once a congestion map exists, you can use it to estimate the required sizes of the channels and blocks that will be needed to help reduce congestion. The channels and blocks are resized and reshaped to reduce congestion in the channels between blocks and inside the blocks themselves so that the estimates based on the congestion map are satisfied. The objective is to reduce congestion by resizing the channels as close to an optimal width as possible based on the existing congestion map. The default is off.

-channels

Use this option if you want to create routing channels between plan groups, black boxes, and soft macros. The channel widths are based on pin counts with nonperpendicular connections to objects on the associated object edge. The default is off.

-refine_placement

If you select this option, the command will refine the placement after it performs placement and shaping of all unfixed plan groups, soft macros, and black boxes in the design. The effect is the same as running the `create_fp_placement` command without any arguments. The default is off.

-constraint_file *file_name*

If you select this option, you must enter the name of the file that contains the path and name of an ASCII file containing relative placement constraints to guide the placement and shaping of objects. The format of the constraint file is one constraint per line. The first word in the line is always the name of the constraint. For example,
`fplModuleDestRegion plan_group_name region_abbrev`
where `region_abbrev` is one of N, S, E, W, NE, NW, SE, SW; S stands for south, and so forth.

For a description of the relative placement constraints, see the section on "Using Relative Placement Constraints to Guide the Placement and Shaping of Objects" in Chapter 6 of the IC Compiler User Guide: Design Planning.

-top_down

If you select this this option, the command sets variables prior to running so that the placement and shaping of objects is constrained like top down placement (`fphBlockPlacment` command) in JupiterXT. Initial virtual flat placement is not assumed with this option. If the `-channels` option is on, the command tries to maintain the block area. After the command runs, any variables it set are restored to their previous settings. The default is off.

-sliver_threshold *threshold*

If you select this option, placement areas between blockages or fixed objects or both, which are smaller than the input value in microns are blocked out during shaping. This avoids "skinny fingers" that can occur when there are many blockages or when there are blockages with small channels between each other or the design core. By default, the value is -1, which means that the tool will automatically calculate a reasonable value for this option.

-place_submacros

If you select this option, the command places hard macros in the top-level cell and in all unfixed soft macros. The default is off.

DESCRIPTION

This command automatically places and shapes plan group boundaries, black boxes, and other soft macros in a design core. It assumes that an initial virtual flat placement has already been run on the design. You can also use this command to refine channels if a congestion map has been created. The command returns 0 if it fails, 1 if it succeeds in reshaping the blocks. If no action was necessary (this happens only with `-incremental` option), the command returns 2. Note that additional options can be set using `set_fp_shaping_strategy` command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example places and shapes objects with all default options.

```
prompt> shape_fp_blocks
```

The following example resizes channels between already placed plan groups/soft macros to match the existing congestion map.

```
prompt> shape_fp_blocks -incremental congestion_driven
```

The following example places and shapes plan groups/soft macros while obeying the constraints specified in the file named `myConstraints`.

```
prompt> shape_fp_blocks -constraint_file myConstraints
```

SEE ALSO

```
set_fp_shaping_strategy(2)  
report_fp_shaping_strategy(2)  
create_fp_placement(2)
```

shell_is_in_upf_mode

Determines if the shell is in UPF mode.

SYNTAX

status **shell_is_in_upf_mode**

ARGUMENTS

The **shell_is_in_upf_mode** command has no arguments.

DESCRIPTION

This command is used to check if the shell is in UPF mode. The command returns 1 if shell is in UPF mode, or 0 otherwise. In 2008.09, UPF mode is on by default. To use non-UPF mode, invoke dc_shell or icc_shell with -non_upf_mode.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows how to determine the mode in which the shell has been invoked:

```
prompt> if {[shell_is_in_upf_mode]} {\
    echo "You invoked dc_shell in UPF mode" \
} else {\
    echo "You invoked dc_shell in non-UPF mode" \
}
```

signoff_drc

Performs design rule checking on the current design using a foundry runset.

SYNTAX

```
status signoff_drc
[-error_view errview_name]
[-check_all_layers
  | -select_layers {collection_of_layers}]
[-read_cel_view]
[-ignore_child_cell_errors]
[-ignore_fill_view_time_stamp]
[-select_rule {list_of_rule_names}]
[-unselect_rule {list_of_rule_names}]
[-bounding_boxes {{llx1 lly1} {urx1 ury1} ...}]
[-excluded_bounding_boxes {{llx1 lly1} {urx1 ury1} ...}]
[-run_dir string]
[-num_cpus integer]
[-batch]
```

Data Types

<i>errview_name</i>	string
<i>collection_of_layers</i>	collection
<i>list_of_rule_names</i>	list
<i>llx1</i>	float
<i>lly1</i>	float
<i>urx1</i>	float
<i>ury1</i>	float
<i>string</i>	<i>errview_name</i>

ARGUMENTS

-error_view *errview_name*
Specifies the name of the error view.
If you do not specify this option, the error view is called *current_design_sdrc.err*.

-check_all_layers
Checks design rules on all layers defined in the runset, such as routing layers and device layers.
When you specify this option, the **-read_cel_view** option is automatically enabled. This **-check_all_layers** option requires CEL views for the child cells and does not work if the reference libraries have only FRAM views for the child cells. In addition, when you use this option, all lower-level errors are flattened to the top level.
If you do not specify this option, only routing layers (metal and via layers) are checked.
The **-check_all_layers** and **-select_layers** options are mutually exclusive; you can specify only one.

-select_layers {*collection_of_layers*}

Specifies a list of routing layers (metal or via layers) on which to perform design rule checking. The layer names must be the names defined in the technology file.

If you do not specify this option, all routing layers are checked.

The **-check_all_layers** and **-select_layers** options are mutually exclusive; you can specify only one.

-read_cel_view

Reads the cell instance data from the CEL view. If the CEL view of a cell instance cannot be found, the command stops.

If you do not specify this option, the cell instance data is read from the FRAM view. If the FRAM view of a cell instance cannot be found, the command ignores this cell instance for design rule checking and continues to check other cell instances. The command issues a warning message for the ignored cell instance in the error file.

-ignore_child_cell_errors

Skips the DRC violations inside all cell instances (reference libraries).

If you do not specify this option, DRC violations inside cell instances are reported.

-ignore_fill_view_time_stamp

Reads the FILL view even though its timestamp is older than the time stamp of the CEL view.

By default, the FILL view is read only when its timestamp is newer than the timestamp of the CEL view.

-select_rule {*list_of_rule_names*}

Specifies the rules to check. You can specify either the rule name or number. You can specify wildcards. The rule names are declared in the COMMENT option of the foundry runset.

By default, all design rules are checked.

If you specify both the **-select_rule** option and the **-unselect_rule** option, the **-unselect_rule** option unselects rules from the selected ones.

-unselect_rule {*list_of_rule_names*}

Specifies the rules that are not checked. You can specify either the rule name or number. You can specify wildcards. The rule names are declared in the COMMENT option of the foundry runset.

By default, all design rules are checked (no rules are excluded).

If you specify both the **-select_rule** option and the **-unselect_rule** option, the **-unselect_rule** option unselects rules from the selected ones.

-bounding_boxes {{*llx1 lly1*} {*urx1 ury1*} ...}

Specifies the areas in which to perform design rule checking. An area is a collection of two points that specify the lower-left and upper-right corners of the area. The units of the coordinates are the main library units.

By default, design rule checking is performed on the entire design.

If you specify both the **-bounding_boxes** option and the **-excluded_bounding_boxes** option, the **-excluded_bounding_boxes** option excludes areas from design rule checking based on the selected areas.

-excluded_bounding_boxes {{*llx1 lly1*} {*urx1 ury1*} ...}

Specifies the areas to exclude from design rule checking. An area is a

collection of two points that specify the lower-left and upper-right corners of the area. The units of the coordinates are the main library units. By default, design rule checking is performed on the entire design (no areas are excluded).

If you specify both the **-bounding_boxes** option and the **-excluded_bounding_boxes** option, the **-excluded_bounding_boxes** option excludes areas from design rule checking based on the selected areas.

-run_dir *string*

Specifies the path of working directory.

If you do not specify this option, the tool uses a directory called `signoff_drc_run` under the current working directory. The path can be either relative or absolute.

-num_cpus *integer*

Specifies the number of CPUs for distributed processing. The number of CPUs must be greater than or equal to 1.

The default value comes from the setting of the **set_physical_signoff_options** command.

-batch

Launches IC Validator or Hercules in the background so that you can use other tool functions without waiting for the IC Validator or Hercules run to finish.

DESCRIPTION

This command launches IC Validator or Hercules with a foundry runset to check for DRC violations in the current design. Use this command to perform design rule checking for designs that use a 65-nm or below process. When you use the **signoff_drc** command for DRC checking, you must provide reference libraries that contain the child cell data and do not have any DRC violations.

Before running this command, you must provide the environment settings for IC Validator or Hercules by using the **set_physical_signoff_options** command. For more details about IC Validator or Hercules, see the *IC Validator Reference Manual* or the *Hercules Reference Manual*.

After you complete design rule checking on the routing layers, you can perform a quick design rule check on the Milkyway database by rerunning the **signoff_drc** command with the **-check_all_layers** option.

Warning

IC Validator and Hercules run hierarchically and report errors in the top-level design by expanding the lower-level errors without merging them for the purpose of the IC Compiler error browser.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example runs design rule checking for the rules whose names match the M*.W.1 and M*.W.3 patterns:

```
prompt> signoff_drc -select_rule {M*.W.1 M*.W.3}
```

The following example runs design rule checking for all rules in the specified runset, except those whose names match the VIA*.S pattern.

```
prompt> signoff_drc -unselect_rule {VIA*.S}
```

The following example runs design rule checking within the (42.67 98.85) (125.39 157.16) bounding box.

```
prompt> signoff_drc -bounding_boxes {{42.67 98.85} {125.39 157.16}}
```

SEE ALSO

```
report_physical_signoff_options(2)  
set_physical_signoff_options(2)  
verify_drc(2)  
verify_route(2)
```

signoff_metal_fill

Invokes Hercules or IC Validator to perform metal fill on current cells to meet the metal density requirements.

SYNTAX

```
status signoff_metal_fill
[-output_view fillview_name]
[-purge]
[-eco]
[-append]
[-mode flat]
[-select_layers {layer_name}]
[-all_runset_layers]
[-bounding_boxes {rectangle_list}]
[-excluded_bounding_boxes {rectangle_list}]
[-run_dir dir_path]
[-timing_preserve_nets {selected_nets}]
[-timing_preserve_setup_slack_threshold slack]
[-timing_preserve_hold_slack_threshold slack]
[-space_to_critical_nets {layer_name space}]
[-fill_over_critical_nets]
[-remove_overlap_with_nets {selected_nets}]
```

Data Types

<i>fillview_name</i>	string
<i>layer_name</i>	string
<i>rectangle_list</i>	rectangle
<i>dir_path</i>	string
<i>selected_nets</i>	string
<i>slack</i>	float
<i>space</i>	float

ARGUMENTS

-output_view *fillview_name*

Specifies the name of the output fill view. The default value is *cell_name.FILL*. The specified file will overwrite the existing fill view file. This option is mutually exclusive with **-purge**, **-eco**, and **-append**.

-purge

Removes metal fill in the default fill view of the current cell. Only fill in selected areas on selected layers would be removed. If neither **-select_layers** nor **-bounding_boxes** is specified, the whole FILL view would be removed. This option is mutually exclusive with **-output_view**, **-eco**, **-append**, **-mode**, and **-excluded_bounding_boxes**. This option cannot be used when **-timing_preserve_*** is specified.

-eco

Specifies this option with the **-select_layers**, **-bounding_boxes**, **-excluded_bounding_boxes**, or **-timing_preserve_*** options for the purpose of

post-ECO layer-based fill removal and refill. The behavior changes according to different option combinations.

1. **-select_layers** and/or **-bounding_boxes/-excluded_bounding_boxes**:

First, all existing metal fill in selected areas on selected layers is removed. Next, Hercules or IC Validator is called to refill those selected areas only. Metal fill outside selected areas or on other metal layers remains unchanged.

2. **-timing_preserve_***:

First, ICC will do RC extraction including the existing fill and identify timing critical nets. Next, all existing metal fill near critical nets is removed. Finally, Hercules or IC Validator is called to refill the whole chip while blocking the regions near critical nets.

3. **-timing_preserve_*** and **-bounding_boxes/-excluded_bounding_boxes**:

First, ICC will do RC extraction including the existing fill and identify timing critical nets. Next, all existing metal fill in selected areas is removed. Finally, Hercules or IC Validator is called to refill the selected areas while blocking the regions near critical nets. Metal fill outside the selected areas remains unchanged.

Both removal and refill are operated on the default fill view of the current cell. This option is mutually exclusive with **-output_view**, **-purge**, and **-append**.

-append

Specifies not to modify existing metal fill in the output fill view. You can use this option to incrementally insert metal fill. By default, the new output fill view will overwrite the existing one. If you specify this option, the new fill would be appended to the existing output view without modifying the existing data. This option is mutually exclusive with **-output_view**, **-purge**, and **-eco**. This option cannot be used when **-timing_preserve_*** is specified.

-mode flat

Specifies the mode of fill that Hercules or IC Validator generates. There are two modes of fill: hierarchical fill and flattened fill. By default, the mode in the runset is used. If the runset writes hierarchical fill, you can use **flat** to convert them to flattened fill. This option is mutually exclusive with **-purge**.

-select_layers {layer_name}

Indicates a list of metal and via layers where the metal fill and via fill are inserted or removed. The names of layers must be strings, such as *m1*, *m2*, *v1*, *v2*, and so on. By default, **signoff_metal_fill** is performed on all metal routing layers that are defined by the **set_ignored_layers** command, but you can fill any specific metal and via layers defined in the technology file using this option.

Do not specify via layers alone, but always include the upper and lower metal layers with via layers. You should not use this option when **-timing_preserve_*** is specified. Via layers are not supported when **-remove_overlap_with_nets** is specified.

-all_runset_layers

Inserts fill for all output fill layers that are defined in the runset. Without this option (the default), **signoff_metal_fill** inserts fill only for metal fill layers and ignores other output fill layers in the runset. This option is mutually exclusive with **-eco**, **-purge**, **-select_layers**, **-bounding_boxes**, **-excluded_bounding_boxes**, **-timing_preserve_***, and **-**

remove_overlap_with_nets.

-bounding_boxes {rectangle_list}

Indicates a list of rectangle bounding boxes where metal fill is inserted or removed. Each rectangle is formed by two coordinates: **{{llx lly} {urx ury}}**, where *llx*, *lly*, *urx*, and *ury* are float values. The `unitLength` defined in the technology file is used. The selected regions are applied to all selected layers. By default, the working area is the entire chip. Note that when using this option, the flat mode will be automatically turned on.

-excluded_bounding_boxes {rectangle_list}

Indicates a list of rectangle bounding boxes where metal fill is not inserted or removed. Each rectangle is formed by two coordinates: **{{llx lly} {urx ury}}**, where *llx*, *lly*, *urx*, and *ury* are float values. The `unitLength` defined in the technology file is used. The excluded regions are applied to all selected layers. By default, the excluded area is none. This option is mutually exclusive with **-purge**.

-run_dir dir_path

Specifies the path of working directory. The default is the **signoff_fill_run** directory under the current working directory. The path can be either relative or complete.

-timing_preserve_nets {selected_nets}

Specifies a list of critical nets. The command preserves the regions near the critical nets on the same layer that is specified by **-space_to_critical_nets** and on the adjacent upper and lower layers that are specified by **-fill_over_critical_nets**. You should not use this option with **-purge**, **-append**, and **-select_layers**.

-timing_preserve_setup_slack_threshold slack

Specifies the threshold of setup slack. The command automatically searches critical nets based on the slack thresholds specified by this option and the **-timing_preserve_hold_slack_threshold** option. Nets with setup slack that is smaller than the given threshold are treated as critical nets. The command preserves the regions near the critical nets on the same layer specified by **-space_to_critical_nets** and on the adjacent upper and lower layers specified by **-fill_over_critical_nets**.

The default threshold is null, that is, no setup slack for all nets when searching critical nets. You should not use this option with **-purge**, **-append**, and **-select_layers**.

-timing_preserve_hold_slack_threshold slack

Specifies the threshold of hold slack. The command automatically searches critical nets based on the slack thresholds specified by this option and the **-timing_preserve_setup_slack_threshold** option. Nets with hold slack that is smaller than the given threshold are treated as critical nets. The command preserves the regions near the critical nets on the same layer specified by **-space_to_critical_nets** and on the adjacent upper and lower layers specified by **-fill_over_critical_nets**.

The default threshold is null, that is, no hold slack of all nets when searching critical nets. You should not use this option with **-purge**, **-append**, and **-select_layers**.

-space_to_critical_nets {*layer_name space*}

Specifies the space to keep away from the selected nets. You can specify a list of layers and spaces. The layer name must be a string, such as *m1*, *m2*, and so on. The unit is the *unitLength* defined in the technology file. By default, the command uses 2 times the *minSpacing* for each layer. This option must be used with **-timing_preserve_*** or **fb-remove_overlap_with_nets**.

-fill_over_critical_nets

Allows fill in the vertical extension of critical nets, the adjacent upper and lower layers. By default, the vertical extension regions of critical nets on the upper and lower layers are blocked with *minSpacing*. If this option is set, only the regions on the same layers are blocked. This option must be used with **-timing_preserve_***.

-remove_overlap_with_nets {*selected_nets*}

Removes the fill that is overlapping metals. You should use this option after you finish metal fill insertion and ECO routing. Use this option to specify nets and the **-space_to_critical_nets** option for spacing. The command removes the fill within the spacing from the specified nets. To remove all fill with overlapping nets, use **-remove_overlap_with_nets {*}**. You should not use the wildcard character "*" for power and ground nets. To include power and ground nets in the selection, specify the full names explicitly, such as {**VDD VSS**}. You can use this option only with the **-select_layers** and **-space_to_critical_nets** options.

DESCRIPTION

This command invokes Hercules or IC Validator to perform metal fill on current cell to meet the metal density requirements and to perform post-ECO removal and refill. The output of this step is a fill view that consists of the fill-track objects and is stored in the FILL directory of the design Milkyway library. You should use this command on a fully routed DRC-clean or near DRC-clean design. The settings of Hercules or IC Validator environment are required and can be specified by using the **set_physical_signoff_options** command before running the **signoff_metal_fill** command.

Multicorner-Multimode Support

This command uses information from active scenarios only.

EXAMPLES

The following example fills all empty regions with metal fill. The output library is the current design library, and the output fill view is *cell_name.FILL*. The output fill view will overwrite the original fill view.

```
prompt> signoff_metal_fill
```

The following example fills all empty regions with metal fill. The output fill view is "newFill.FILL".

```
prompt> signoff_metal_fill -output_view newFill
```

The following example fills all empty regions with flattened metal fill.

```
prompt> signoff_metal_fill -mode flat
```

The following example fills all empty regions on **metal1**, outside the region {{100 150} {300 200}}.

```
prompt> signoff_metal_fill -select_layers {m1} \  
-excluded_bounding_boxes {{100 150} {300 200}}
```

The following example removes the default fill view of the current cell.

```
prompt> signoff_metal_fill -purge
```

The following example removes fill on layers **metal1** and **metal3** in the default fill view of the current cell.

```
prompt> signoff_metal_fill -purge -select_layers {m1 m3}
```

The following example removes all metal fill on **metal1** and **metal3**, and then refills those two metal layers. The existing metal fills on other metal layers remain unchanged.

```
prompt> signoff_metal_fill -eco -select_layers {m1 m3}
```

The following example fills all empty regions in {{100 150} {300 200}}, without overwriting the existing metal fill. Note that the flat mode will be automatically turned on.

```
prompt> signoff_metal_fill -append -bounding_boxes {{100 150} {300 200}}
```

The following example fills all empty regions with metal fill, but not in the regions within 0.5um away from critical nets on metal1 and metal2, 2x minSpacing away from critical nets on other layers, and the regions within minSpacing away from the vertical extension of critical nets on the layers above and below. Those nets with negative setup slack are critical nets. The hold slack is not considered.

```
prompt> signoff_metal_fill -timing_preserve_setup_slack_threshold 0.0 \  
-space_to_critical_nets {m1 0.5 m2 0.5}
```

The following example fills all empty regions with metal fill, but not in the regions within 2x minSpacing away from critical nets on the same layer and the regions within minSpacing away from the vertical extension of critical nets on layers above or below. The nets with setup slack that is smaller than -0.5 and/or the hold slack that is smaller than -0.3 are critical nets.

```
prompt> signoff_metal_fill -timing_preserve_setup_slack_threshold -0.5 \  
-timing_preserve_hold_slack_threshold -0.3
```

The following example fills all empty regions with metal fill, but not in the regions within 2x minSpacing away from critical nets on the same layer. The critical nets are n1, n2, and n3.

```
prompt> signoff_metal_fill -timing_preserve_nets {n1 n2 n3} \  
-fill_over_critical_nets
```

The following example removes fill on layers m1, m2, and m3, which are overlapping with the existing nets.

```
prompt> signoff_metal_fill -remove_overlap_with_nets {*} \  
-select_layers {m1 m2 m3} -space_to_critical_nets {m1 0.2 m2 0.2 m3 0.2}
```

The following example removes fill in the regions within 2x minSpacing away from nets n1 and n2.

```
prompt> signoff_metal_fill -remove_overlap_with_nets {n1 n2}
```

SEE ALSO

```
insert_metal_filler(2)  
set_physical_signoff_options(2)  
report_physical_signoff_options(2)
```

signoff_opt

Performs sign-off ECO optimization.

SYNTAX

```
status signoff_opt
[-effort low | medium | high]
[-only_psyn]
[-no_design_rule | -only_design_rule | -only_hold_time]
[-power | -only_power_recovery]
[-xtalk_reduction | -only_xtalk_reduction]
[-full_extract]
[-full_analysis]
[-skip_initial_analysis]
[-num_iteration integer]
[-only_focal all | file_name]
[-star_only]
[-pt_only]
[-aocvm]
[-path_based_analysis]
[-update_rail_voltage]
[-variation]
[-ignore_design_readiness]
[-ignore_library_mismatch]
[-keep_license]
[-snapshot base_name]
```

ARGUMENTS

- effort low | medium | high**
Specifies the effort level for **signoff_opt**. The default effort level is **medium**. If you specify **high**, **signoff_opt** spends more time to further improve the quality of results (QoR). If you specify **low**, **signoff_opt** spends less time on improving the quality of results (QoR).
- only_psyn**
Performs optimization only. No incremental sign-off analysis is performed.
- no_design_rule | -only_design_rule | -only_hold_time**
Specifies whether or not the tool fixes design rule/hold time violations before exiting.
- power | -only_power_recovery**
Enables multi-voltage-threshold based leakage power optimization as part of the optimization loop in signoff_opt flow. By default, this option is off.
- xtalk_reduction | -only_xtalk_reduction**
Runs routing-based crosstalk reduction optimization in addition to signal integrity (SI) optimization. By default, this option is off.
- full_extract**
Performs full extraction with StarRC. You must use the **-full_extract** option

with the **-full_analysis** option. Otherwise, the tool runs StarRC with incremental extraction and creates a full netlist output.

-full_analysis

Performs full analysis with PrimeTime.

-skip_initial_analysis

Skips the initial sign-off analysis. This option is valid only if you have run the **run_signoff** command before running the **signoff_opt** command.

-num_iteration *integer*

Specifies the number of iterations to perform for optimization and incremental sign-off analysis. The default is 0, which lets the tool decide the appropriate number of iterations.

-only_focal all | file_name

Invokes only focal optimization during **signoff_opt**. When you specify **all**, all violations are considered during focal optimization. When you specify a file name, only the violations specified in the file are considered during focal optimization. The file format is the same as for the **focal_opt** command. By default, when **-only_focal** is used, only focal setup optimization is performed. When used with **-only_hold_time**, only focal hold optimization is performed. When used with **-only_design_rule**, only focal logic design rule optimization is performed. It is not allowed to mix setup, hold, or design rule optimizations during the same session of **-only_focal**.

-star_only

Invokes StarRC only during sign-off analysis and continues to use IC Compiler for other tasks, such as using the IC Compiler timer, not PrimeTime, for timing analysis.

-pt_only

Invokes PrimeTime only during sign-off analysis and continues to use IC Compiler for other tasks, such as using IC Compiler extraction, not StarRC, for extraction.

-aocvm

Indicates that the timing paths are to be adjusted using advanced on-chip variation (OCV) modeling information. Advanced OCV modeling computes the effects of OCV on timing by using variable derating factors that consider the location and the logic depth of each path being analyzed. This approach reduces pessimism compared to global derating, thereby minimizing unnecessary overdesign.

-path_based_analysis

Enables the path-based analysis capability in PrimeTime.

-update_rail_voltage

Estimates and updates rail voltages for the changed cells. This option is valid only with the IR analysis flow.

-variation

Invokes StarRC VX and PrimeTime VX to perform variation-aware analysis.

-ignore_design_readiness
Disables the design readiness check in **signoff_opt**.

-ignore_library_mismatch
Ignores library mismatches between IC Compiler and PrimeTime saved sessions.

-keep_license
Prevents the release of IC Compiler licenses while running sign-off tools and the release of PrimeTime licenses while running optimization.

-snapshot *base_name*
Enables snapshots. When snapshots are enabled, the **run_signoff** command saves the design after each optimization. The saved design is named by using the specified *base_name* appended with an index. The **signoff_opt** command also saves a session from PrimeTime. This saved session is also named by using the specified *base_name* appended with an index.

DESCRIPTION

This command performs optimization based on sign-off costing from sign-off tools: PrimeTime and StarRC. StarRC provides sign-off quality parasitic extraction, and PrimeTime provides sign-off quality timing and SI analysis. IC Compiler uses the sign-off costing from these tools to perform optimization on the design. Use the **set_primetime_options** and **set_starrcxt_options** commands to customize the PrimeTime and StarRC settings for the **signoff_opt** command.

In general, designs still need to go through regular timing and signal integrity closure flows within IC Compiler to reduce the number of violations seen by the sign-off tools as much as possible. This is to reduce the runtime of the **signoff_opt** command and to achieve the best quality of results (QoR).

The **signoff_opt** command has a built-in readiness checker that checks the following conditions:

- The design has no more than 1000 violating end points in either best-case or worst-case analysis.
- The design has no routing shorts.
- The number of routing design rule violations is no more than the larger of 0.1 percent of the number of nets or 1000.
- The number of logic design rule violations is no more than the larger of 0.1 percent of the number of nets or 1000.

If the design does not meet the design readiness check, you can still force **signoff_opt** to run sign-off driven optimization. However, the design might not converge to a solution with better quality of results (QoR).

To enable signal integrity fixing, use the **set_si_options** command. To enable hold time fixing, use the **set_fix_hold** command.

Multicorner-Multimode Support

This command uses information from active scenarios only.

EXAMPLES

The following example runs the **signoff_opt** command.

```
prompt> open_mw_cel -lib my_lib my_cel
prompt> set_primetype_options \
-exec_dir /PrimeTime/linux/syn/bin
prompt> set_starrcxt_options \
-exec_dir /StarRCXT/linux/bin \
-max_nxtgrd_file /design/max.nxtgrd \
-map_file /design/map_file
prompt> signoff_opt
prompt> save_mw_cel
```

SEE ALSO

```
clock_opt(2)
place_opt(2)
report_primetype_options(2)
report_starrcxt_options(2)
route_opt(2)
focal_opt(2)
set_fix_hold(2)
set_primetype_options(2)
set_si_options(2)
set_starrcxt_options(2)
```


size_cell

Relinks leaf cells to a new library cell that has the required drive strength (or other properties).

SYNTAX

```
collection size_cell  
cell_object  
lib_cell_object
```

Data Types

```
cell_object          list  
lib_cell_object      string
```

ARGUMENTS

cell_object
Specifies the leaf cell to be relinked. Each cell must be in scope (at or below the current instance).

lib_cell_object
Specifies the library cell objects to which the specified cell is to be linked. In this case, each object is either a named library cell or a library cell collection. The library cells specified must be logical equivalent to the library cell which will be relinked.

DESCRIPTION

The **size_cell** command changes the drive strength (or other properties) of a leaf cell by relinking it to a new library cell that has the required properties. Like all other netlist editing commands, for **size_cell** to succeed, all of its arguments must succeed. If any arguments fail, the netlist remains unchanged. **size_cell** returns the collection of cells sized if successful and a NULL string if unsuccessful. Each cell in *cell_list* must be in scope; that is, at or below the current instance.

The *lib_cell* that is being swapped in must conform to the following restrictions:

- *lib_cell* must be functionally compatible with the current library cell to which the cells in *cell_list* are linked.
- *lib_cell* cannot be the same as the current library cell of any of the cells in *cell_list*.
- *lib_cell* must have the same pin count and pin directions as the current library cell of the cells in *cell_list*.

You can get a list of library cells that are compatible with a given cell using the **get_alternative_lib_cells** command.

The **size_cell** command is for leaf cells only. **size_cell** is optimized for incremental timing and does not cause a full timing update.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

In the following example, an attempt to size a cell fails because the target is not functionally equivalent. After finding a compatible library cell, the second attempt succeeds.

```
prompt> size_cell o_reg1 class/NR4P  
Error: Could not size 'o_reg1' ('FD2') with 'NR4P'. (NLE-009)
```

```
prompt> get_alternative_lib_cells o_reg1  
{ "class/FD2P" }
```

```
prompt> size_cell o_reg1 class/FD2P  
Information: Sizing cell 'o_reg1' with lib cell 'class/FD2P'  
{o_req1}
```

SEE ALSO

```
get_alternative_lib_cells(2)  
get_libs(2)  
get_lib_cells(2)
```

sizeof_collection

Returns the number of objects in a collection.

SYNTAX

```
int sizeof_collection  
collection1
```

Data Types

collection1 collection

ARGUMENTS

collection1
Specifies the collection for which to get the number of objects. If the empty collection (empty string) is used for the *collection1* argument, the command returns 0.

DESCRIPTION

The **sizeof_collection** command is an efficient mechanism for determining the number of objects in a collection.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows a simple way to find out how many objects matched a particular pattern and filter in a **get_cells** command.

```
prompt> echo "Number of hierarchical cells: \  
[sizeof_collection [get_cells * -hier] \  
-filter "is_hierarchical == true"]]"
```

Number of hierarchical cells is: 10

The following example shows what happens when the argument to **sizeof_collection** results in an empty collection.

```
prompt> set s1 [get_cells *]  
{"u1", "u2", "u3"}  
prompt> set ssize [filter_collection $s1 "area < 0"]  
prompt> echo "Cells with area < 0: [sizeof_collection $ssize]"  
Cells with area < 0: 0  
prompt> unset s1
```

SEE ALSO

`collections(2)`
`filter_collection(2)`

skew_opt

Optimizes clock skews to increase timing slack and writes the solution to an output file. By default, the output file is named "skew_opt.tcl" and is automatically sourced.

SYNTAX

```
status skew_opt
[-output file_name]
[-clock_balancing_only]
[-no_optimization]
[-no_auto_source]
[-fix_boundary_pins]
[-ignore_boundary_paths]
[-pins pin_list]
[-clocks clock_list]
[-path_groups path_group_list]
[-enable_pins]
[-macro_pins]
[-halfcycle_path_pins]
[-setup]
[-hold]
[-setup_margin setup_margin_value]
[-hold_margin hold_margin_value]
[-guard_band guard_band_value]
[-adjustment_limit adjustment_limit_value]
[-decrease_factor decrease_factor_value]
[-improvement_threshold improvement_threshold_value]
[-resolution resolution_value]
```

Data Types

<i>file_name</i>	string
<i>pin_list</i>	list of pins
<i>clock_list</i>	list of clocks
<i>path_group_list</i>	list of path-groups
<i>setup_margin_value</i>	float
<i>hold_margin_value</i>	float
<i>guard_band_value</i>	float
<i>adjustment_limit_value</i>	float
<i>decrease_factor_value</i>	float
<i>improvement_threshold_value</i>	float
<i>resolution_value</i>	float

ARGUMENTS

This group of options controls what **skew_opt** does with the solution file:

```
-output file_name
    Specify the name of the solution file. By default the name of the solution
    file is "skew_opt.tcl".
```

-clock_balancing_only

Only configure inter-clock delay balancing groups; do not write any pin-specific annotations to the output file. This is useful to automatically configure inter-clock balancing without changing pin skews.

-no_optimization

Do not optimize; write the solution file using the present clock network latencies. This option uses **skew_opt** to translate skews set by some other means to the clock-tree exceptions that will implement those skews. By default **skew_opt** will optimize the skews itself before writing the solution file.

-no_auto_source

Do not automatically source the solution file. This option can be used to explore **skew_opt** improvements with different settings without applying the solution file. You can source the solution file manually when/if desired. By default **skew_opt** will source the solution file. See the OUTPUT section for more information about controlling how sections of the solution file are activated.

This group of options controls how **skew_opt** handles I/O paths:

-fix_boundary_pins

Do not adjust latencies at the clock pins of registers participating in paths with I/O ports. This option allows external blocks to be designed without having to know the latency at each individual boundary register. This option also activates the **-ignore_boundary_paths** option below.

-ignore_boundary_paths

Ignore paths to and from I/O ports. Allows latency adjustments on boundary registers to improve register-to-register paths without caring about the paths to and from I/O ports. This option can be useful if the external clock latencies for the I/O ports are unknown. To additionally prevent latency adjustments at boundary registers, use the **-fix_boundary_pins** option instead.

This group of options filters what **skew_opt** considers for optimization; when the options are used together, the intersection of their sets are used:

-pins *pin_list*

Specify a list of clock pins to optimize skews for.

-clocks *clock_list*

Specify a list of clocks to optimize skews for.

-path_groups *path_group_list*

Specify a list of path-groups to optimize skews for.

This group of options selects what **skew_opt** considers for optimization; when the options are used together, the union of their sets are used; when a pin is selected for optimization by the option(s) from this group, all the timing paths connected to the pin are used in optimization:

-enable_pins

Only adjust those clock pins that participate as startpoints in enable-timing paths to clock-gating endpoints.

`-macro_pins`
Only adjust clock pins on macro cells.

`-halfcycle_path_pins`
Only adjust clock pins that participate in paths where the startpoint and endpoint have opposing edge directions.

This group of options selects what type of timing constraints to consider, and how much margin to subtract from the slacks to coerce `skew_opt` to improve timing for paths that meet timing by a small amount:

`-setup`
Collect setup constraints. By default only setup constraints are collected, so specifying `-setup` by itself is unnecessary.

`-hold`
Collect hold constraints. By default this is not performed. This option can be specified with or without `-setup`. See the DETAILS section for more information about using hold constraints with **`skew_opt`**.

`-setup_margin setup_margin_value`
Subtract this value (in nanoseconds) from the collected setup slacks before launching skew optimization. A positive value can be used to coerce **`skew_opt`** to create additional setup slack for paths that are only slightly non-violating. The default value is 0.

`-hold_margin hold_margin_value`
Subtract this value (in nanoseconds) from the collected hold slacks before launching skew optimization. A positive value can be used to coerce **`skew_opt`** to create additional hold slack for paths that are only slightly non-violating. The default value is 0.

This group of options controls the adjustments **`skew_opt`** makes:

`-guard_band guard_band_value`
Specify a positive slack threshold (in nanoseconds) for clock pins above which can be consumed to improve the worst/total negative slack for the design. This can be used to reduce an increase in the number of violating paths at the expense of less timing improvement. The default value is 0, which means that a guard-band is not used (not that positive slacks are kept at or above zero). To keep positive slacks at or above zero, use a tiny positive value (e.g. 0.001).

`-adjustment_limit adjustment_limit_value`
Specify a limit on skew adjustments (in nanoseconds). Changes in clock network latency, either increases or decreases, will be constrained by this amount. Changes are also constrained by the clock period, and decreases have an additional constraint controlled by the `-decrease_factor` option. The default value of `1e+30` essentially shuts this feature off.

`-decrease_factor decrease_factor_value`
Specify a fractional limit on skew decreases using a value between zero and one. The minimum latency allowed will be the larger of: zero, the initial latency minus the adjustment limit, the initial latency minus the clock period, and the initial latency multiplied by one minus the decrease factor.

The default decrease factor is 0.5, meaning that latencies will be allowed to decrease up to 50% of their initial value. It is easier for clock tree synthesis to increase latencies than to decrease them, and a zero latency is not physically realizable. Some highly customized flows require an increase-only solution, in which case a decrease factor of 0 may be used. For designs with many clock-tree levels, a larger decrease factor may yield more slack improvement.

`-improvement_threshold improvement_threshold_value`

Specify the threshold (in nanoseconds) that **skew_opt** must improve the worst negative slack (WNS) by in order to create a solution file. If **skew_opt** cannot meet this threshold, an empty solution file will be created. If both `-setup` and `-hold` are specified, the combined WNS improvement will be compared against the improvement threshold. The default value is 0.010 (10ps).

`-resolution resolution_value`

Snap the clock-tree exception values to an integer multiple of this value (in nanoseconds). The default value is 1e-3 (1ps), and the minimum value allowed is 1e-4. Some highly customized flows require the clock-tree exceptions to be an integer multiple of some specific delay value.

DESCRIPTION

The **skew_opt** command does the following by default:

- Collects the worst timing path between every startpoint/endpoint pair.
- Collects the clock latencies at every startpoint/endpoint.
- Classifies each pin as adjustable, fragile, or fixed.
- Optimizes the latencies at adjustable pins to improve timing slacks.
- Converts the latencies to skews for clock-tree synthesis to implement.
- Configures inter-clock balancing for clock-tree synthesis.
- Writes a solution file.
- Sources the solution file.

The following options control what happens with the solution file:

`-output -clock_balancing_only -no_optimization -no_auto_source`

The following options control how I/O paths and pins are handled:

`-fix_boundary_pins -ignore_boundary_paths`

The following options control which pins and paths are considered:

`-pins -clocks -path_groups -enable_pins -macro_pins -halfcycle_path_pins`

The following options control how types and values of timing constraints are considered:

`-setup -hold -setup_margin -hold_margin`

The following options control the skew adjustments:

`-guard_band -adjustment_limit -decrease_factor -improvement_threshold -resolution`

Each of the above options is described in detail at the top of this man page in the ARGUMENTS section.

FLOW

The **skew_opt** command is intended to be used within a two-pass clock tree synthesis flow. The first pass produces realistic skews and slacks for **skew_opt** to improve upon, and the second pass implements the **skew_opt** solution. By using two passes, reliable improvements can be made to I/O paths and paths to clock-gating cells. Additionally, most of the changes **skew_opt** makes can be implemented in the second pass with new clustering and leveling, instead of adding many clock-tree buffers.

Here is the basic recommended flow:

```
place_opt save_mw_cel -as placed close_mw_cel
```

```
open_mw_cel placed skew_opt -clock_balancing_only clock_opt -inter_clock_balance  
skew_opt close_mw_cel
```

```
open_mw_cel placed source skew_opt.tcl clock_opt -inter_clock_balance route_opt
```

If `update_clock_latency` is to be used to change the ideal latencies for I/O ports, this is the recommended flow:

```
place_opt save_mw_cel -as placed close_mw_cel
```

```
open_mw_cel placed set_latency_adjustment_options ... skew_opt -clock_balancing_only  
clock_opt -inter_clock_balance -update_clock_latency -no_clock_route route_group -  
all_clock_nets extract_rc skew_opt close_mw_cel
```

```
open_mw_cel placed source skew_opt.tcl clock_opt -inter_clock_balance -  
no_clock_route route_group -all_clock_nets extract_rc route_opt
```

The above example separates clock-routing from **clock_opt** in order to avoid having a second `update_clock_latency` called after the internal clock-routing. This ensures **skew_opt** will see the same I/O latencies as **psynopt** does, which makes the flow more reliable.

PLEASE NOTE: It is imperative that **update_clock_latency** not be run after the

solution file is sourced, otherwise the solution will be unrealized.

OUTPUT

The changes are written to an output file, also referred to as the "solution file", which contains three types of settings:

```
command group controlling variable -----  
----- set_clock_latency skew_opt_skip_ideal_clocks set_clock_tree_exceptions  
skew_opt_skip_propagated_clocks set_inter_clock_delay_options  
skew_opt_skip_clock_balancing
```

Each type of command is grouped together and is applied only when the associated controlling variable is either *false* or undefined. Note that the controlling variables only affect which pieces of the solution file are active, not any behavior of the **skew_opt** command itself. The command types are used as follows.

A first set of **set_clock_latency** commands is used to capture the ideal latencies on clock objects used to clock ports at the time **skew_opt** is run; these commonly come from the **update_clock_latency command**, but they can also come from a SDC file loaded earlier in the design flow.

A second set of **set_clock_latency** commands is used to capture the ideal latencies on pins in order to allow users to see an estimate of the timing improvement before the second pass through **clock_opt**. Note that this estimate is only accurate immediately after **skew_opt** is run after the first pass through **clock_opt**; this is because there will be realistic latencies at clock-gating cells and realistic slews to select setup and hold times.

PLEASE NOTE: Timing reports may not be accurate after sourcing the solution file ahead of the second pass through **clock_opt**. This is because the **skew_opt** solution is in response to the propagated clock tree state.

PLEASE NOTE: The variable **skew_opt_skip_ideal_clocks** should not be set true before sourcing the solution file ahead of the second pass through **clock_opt**; otherwise, the I/O latencies will likely be very different between the passes, and timing improvement will likely not occur.

Next a set of **set_clock_tree_exceptions** commands are provided to configure clock tree synthesis to implement the optimized skews. These are rise/fall specific float-pin exceptions.

Lastly, a set of **set_inter_clock_delay_options** commands is provided to configure clock tree synthesis to balance clocks that relate through timing paths.

DETAILS

Hold constraints are not collected by default because we assume **skew_opt** will be used primarily in a pre-route flow. Before steps like scan-chain ordering, CTS, and routing are performed, hold slacks are not known with enough predictability to be useful to guide the skew optimization. To minimize the impact on hold violations, however, **skew_opt** will post-process the solution found for the best setup WNS so that it is most favorable for hold WNS without worsening the setup WNS. If it is

desired to test what would happen if hold constraints are considered, we provide the -hold option to **skew_opt**.

During optimization **skew_opt** reports the worst negative slack (WNS) and cumulative negative slack (CNS) for the pins being adjusted. CNS is different from total negative slack (TNS); TNS is the sum of the WNS at every endpoint, whereas CNS is the sum of the WNS between every startpoint/endpoint pair.

After optimization **skew_opt** may show one of the following messages:

"There are dangling paths in this design." "There are floating clock-pins in this design." "There are clocks shared by data ports and sink pins in this design."

These messages are for informational purposes only and require no action. Dangling paths involve registers that are exclusively either a startpoint or an endpoint; designs with dangling paths require more processing than designs without them. Floating clock-pins occur at registers that only have paths to and from themselves; these cannot be optimized. When clocks are shared by data ports and sink pins in a design, the ports are clocked by an ideal clock latency value, not a propagated one; virtual-clocks can be used to make this more clear if desired.

Pre-existing exclude exceptions set with the **set_clock_tree_exceptions** command are honored by **skew_opt**; no adjustments are made to those pins.

The conversion of latencies to skews is different depending upon whether a subset option (i.e. -pins, -clocks, -path_groups, etc.) is in effect. When the default behavior is used to write clock tree exceptions for all sinks, this formula is used:

$$\text{float\_pin\_value} = \text{minimum\_desired\_latency} - \text{desired\_latency}$$

where minimum_desired_latency is the minimum desired_latency over all pins. This has the effect of creating only non-positive float-pin values, which yields the best skew correlation from clock tree synthesis. If a master clock domain has source latency, it will be included in the above conversion.

If a subset option is in use, however, this formula is used:

$$\text{float\_pin\_value} = \text{initial\_latency} - \text{desired\_latency}$$

where initial_latency is the value before **skew_opt** is run.

LIMITATIONS

Note that the following types of paths fundamentally cannot be improved by adjusting skew:

-- feedthrough paths between ports -- looping paths from a register to itself

Additionally, clock tree synthesis has the limitation that paths to or from pins in the middle of clock trees (e.g. clock-gating cells) cannot be adjusted by **set_clock_tree_exceptions** commands. This is because placing a float-pin exception on a nonstop-pin will convert that pin to a stop-pin. Hence **skew_opt** will not optimize latencies at clock-gating cells.

Similarly, clock tree synthesis cannot separately shift the phase of clock pins within an interface logic model (ILM), so **skew_opt** will not optimize latencies within ILM's.

Since the slack at level-sensitive latches is not a simple function of the clock network latency, **skew_opt** presently does not optimize latencies at level-sensitive latches, nor the edge-triggered registers that connect directly to and from them.

The following "fragile" clock pins are written to the solution file even though **skew_opt** did not optimize them; this is done to preserve their skew in the context of those pins **skew_opt** did optimize:

```
-- ILM clock pins -- unconstrained clock pins -- level-sensitive latch clock pins --  
clock pins at registers connecting to/from latches
```

The inclusion of the above pins is affected by the various subset options (e.g. -pins, -clocks, -path_groups, etc.). Unclocked sink pins get exclude exceptions instead of float-pin exceptions.

skew_opt will improve the slack for the paths it can adjust skews for, but if those paths do not include the critical paths, the **report_qor** command may not show the WNS improvement claimed by **skew_opt**.

Users can create a separate path-group for feedthrough paths:

```
group_path -from [all_inputs] -to [all_outputs] -name FEEDTHROUGH
```

skew_opt does not adjust skews to satisfy **set_min_delay** or **set_max_delay** constraints; the **report_qor** command may show negative slack due to those constraints even though the skew-dependent slack has improved.

Typically the set of pins affected by the **skew_opt** output file should match that shown by the **report_clock_timing** latency report; however, **report_clock_timing** also shows clock pins on unconstrained paths which **skew_opt** ignores.

Presently **skew_opt** does not consider clock-tree exceptions placed on combinational pins. It is possible that manually placed clock-tree exceptions will interfere with those created by **skew_opt**; the **check_clock_tree** command can be used to check for such issues.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

If **skew_opt** is run with no options, skews on all clock pins are adjusted to maximize setup slack, and the output file *skew_opt.tcl* is created and sourced:

```
prompt> skew_opt
```

A user can restrict the pins adjusted by **skew_opt** by using the **-pins** option:

```
prompt> skew_opt -pins [get_pins "prefix/*/clk"]
```

A user can restrict the path-groups adjusted by **skew_opt** by using the **-path_groups** option:

```
prompt> skew_opt -path_groups [get_path_groups "prefix/*"]
```

A user can restrict the sourcing of the output file to not apply the **set_inter_clock_delay_options** commands by doing this:

```
prompt> set skew_opt_skip_clock_balancing true
prompt> skew_opt
```

SEE ALSO

```
set_clock_latency(2)
set_clock_tree_exceptions(2)
set_inter_clock_delay_options(2)
set_latency_adjustment_options(2)
balance_inter_clock_delay(2)
update_clock_latency(2)
report_qor(2)
report_clock_timing(2)
remove_ideal_latency(2)
remove_propagated_clock(2)
check_clock_tree(2)
```

slot_wire

Slots wide wires and contact arrays on selected nets by replacing them with thinner wires and smaller arrays.

SYNTAX

```
integer slot_wire
-nets {collection_of_nets}
[-signal_net]
[-cutwidth {layer distance list}]
[-cutlength {layer distance list}]
[-width {layer distance list}]
[-length {layer distance list}]
[-sidespace {layer distance list}]
[-endspace {layer distance list}]
[-sideclearance {layer distance list}]
[-endclearance {layer distance list}]
[-no_stagger {layer list}]
[-dmode {layer list}]
[-treat_width_as_max]
[-treat_length_as_min]
[-treat_space_clearance_min]
[-min_via_removal]
[-recreate_vias]
[-report]
[-report_wire_relationship file_name]
[-undo]
```

Data Types

<i>collection_of_nets</i>	collection
<i>file_name</i>	string

ARGUMENTS

- nets {*collection_of_nets*}
- Specifies the nets to slot wires on. If you specify more than one net in a collection they must be separated by a space. This option is required.
- signal_net
- Allows the slotter to work on signal nets as well as preroute wires. By default, the command considers preroute wires only.
- cutwidth {*layer distance list*}
- Specifies wire width threshold to slot. The parameter is a list of layer and wire width threshold values. The <layer> can be specified as either the layerNumber or maskName from the technology file. There is one predefined name to indicate that the following value will apply to all layers, rather than a particular layer. It is *all_layers*. Names and values in a list should be separated by space. The default is set to the value of the SlotRule widthThreshold, when SlotRule is defined for a layer in the technology file, or 99, if not defined.

`-cutlength {layer distance list}`

Specifies wire length threshold to slot. The parameter is a list of layer and wire length threshold values. The `<layer>` can be specified as either the `layerNumber` or `maskName` from the technology file. There is one predefined name to indicate that the following value will apply to all layers, rather than a particular layer. It is `all_layers`. Names and values in a list should be separated by space. The default is set to the value of the `SlotRule lengthThreshold`, when `SlotRule` is defined for a layer in the technology file, or 99, if not defined.

`-width {layer distance list}`

Specifies open slot width. The parameter is a list of layer and open slot width values. The `<layer>` can be specified as either the `layerNumber` or `maskName` from the technology file. There is one predefined name to indicate that the following value will apply to all layers, rather than a particular layer. It is `all_layers`. Names and values in a list should be separated by space. `<distance>` must be within the range of the `minSlotWidth` and `maxSlotWidth`, when the `SlotRule` is defined for a layer in the technology file. The default is set to the value of the `SlotRule minSlotWidth`, when `SlotRule` is defined for a layer in the technology file, or `minSpace`, if not defined.

`-length {layer distance list}`

Specifies the open slot length. The parameter is a list of layer and open slot length values. The `<layer>` can be specified as either the `layerNumber` or `maskName` from the technology file. There is one predefined name to indicate that the following value will apply to all layers, rather than a particular layer. It is `all_layers`. Names and values in a list should be separated by space. `<distance>` must be within the range of the `minSlotLength` and `maxSlotLength`, when the `SlotRule` is defined for a layer in the technology file. The default is set to the value of the `SlotRule maxSlotLength`, when `SlotRule` is defined for a layer in the technology file, or `10*minSpace`, if not defined.

`-sidespace {layer distance list}`

Specifies the spacing between the sides of two adjacent parallel open slots. The parameter is a list of layer and sidespace values. The `<layer>` can be specified as either the `layerNumber` or `maskName` from the technology file. There is one predefined name to indicate that the following value will apply to all layers, rather than a particular layer. It is `all_layers`. Names and values in a list should be separated by space. `<distance>` must be within the range of the `minSideSpace` and `maxSideSpace`, when the `SlotRule` is defined for a layer in the technology file. The default is set to the value of the `SlotRule maxSideSpace`, when `SlotRule` is defined for a layer in the technology file, or `10*minWidth`, if not defined.

`-endspace {layer distance list}`

Specifies the spacing between the ends of two consecutive open slots. The parameter is a list of layer and endspace values. The `<layer>` can be specified as either the `layerNumber` or `maskName` from the technology file. There is one predefined name to indicate that the following value will apply to all layers, rather than a particular layer. It is `all_layers`. Names and values in a list should be separated by space. `<distance>` must be within the range of the `minEndSpace` and `maxEndSpace`, when the `SlotRule` is defined for a layer in the technology file. The default is set to the value of the `SlotRule maxEndSpace`, when `SlotRule` is defined for a layer in the technology file, or `10*minWidth`,

if not defined.

`-sideclearance {layer distance list}`

Specifies the spacing from the out-most slot to the side-edge of the slotting wire. The parameter is a list of layer and sideclearance values. The `<layer>` can be specified as either the `layerNumber` or `maskName` from the technology file. There is one predefined name to indicate that the following value will apply to all layers, rather than a particular layer. It is `all_layers`. Names and values in a list should be separated by space. `<distance>` must be within the range of the `minSideClearance` and `maxSideClearance`, when the `SlotRule` is defined for a layer in the technology file. The default is set to the value of the `SlotRule maxSideClearance`, when `SlotRule` is defined for a layer in the technology file, or `10*minWidth`, if not defined.

`-endclearance {layer distance list}`

Specifies the space from the out-most slot to the end-edge of the slotting wire. The parameter is a list of layer and endclearance values. The `<layer>` can be specified as either the `layerNumber` or `maskName` from the technology file. There is one predefined name to indicate that the following value will apply to all layers, rather than a particular layer. It is `all_layers`. Names and values in a list should be separated by space. `<distance>` must be within the range of the `minEndClearance` and `maxEndClearance`, when the `SlotRule` is defined for a layer in the technology file. The default is set to the value of the `SlotRule maxEndClearance`, when `SlotRule` is defined for a layer in the technology file, or `10*minWidth`, if not defined.

`-no_stagger {layer list}`

Specifies not to stagger slots. The parameter is a list of layer(s). The `<layer>` can be specified as either the `layerNumber` or `maskName` from the technology file. There is one predefined name to indicate that the option will apply to all layers, rather than a particular layer. It is `all_layers`. Names in a list should be separated by space. The default is set to allow slot staggering.

`-dmode {layer list}`

Enables the dimension rule mode. In this mode, slotting must meet the following dimension rule constraints: It must create the number of slot rows as $n = \text{wire\_width} / \text{cutWidth}$. It must center the rows at multiples of $\text{wire\_width} / (n+1)$. The command issues a warning if, using these constraints, it cannot find a valid slotting solution for a particular wire. The parameter is a list of layer(s). The `<layer>` can be specified as either the `layerNumber` or `maskName` from the technology file. There is one predefined name to indicate that the option will apply to all layers, rather than a particular layer. It is `all_layers`. Names in a list should be separated by space. The default is a disabled dimension rule mode.

`-treat_width_as_max`

Sets the specified slot width to the maximum. This means that the open slot width might be adjusted to smaller values to fit some wide wires when necessary, but no less than the value of the `minSlotWidth`, when defined in the technology file. The default is to set the specified slot width to the minimum, allowing open slot width to be adjusted to larger values to fit some wide wires when necessary, but no larger than the value of the `maxSlotWidth`, when defined in the technology file.

-treat_length_as_min
 Sets the specified slot length to the minimum. This means that the open slot length might be adjusted to larger values to fit some wide wires when necessary, but no larger than the value of the `maxSlotLength`, when defined in the technology file. The default is to set the specified slot length to the maximum, allowing open slot length to be adjusted to smaller values to fit some wide wires when necessary, but no less than the value of the `minSlotLength`, when defined in the technology file.

-treat_space_clearance_min
 Treats `sideSpace`, `endSpace`, `sideClearance` and `endClearance` as minimum. The actual parameters might be adjusted to larger values, but should be less than the respective maximum values when defined in the technology file. The default is to treat them as maximum. In this case the actual parameters might be adjusted to smaller values, but should be no less than the respective minimum values when defined in the technology file.

-min_via_removal
 Minimizes the removal of existing vias on a slotted wire. The default is not to minimize.

-recreate_vias
 Recreates vias on slotted wires in the original via area only. When not selected, vias can be recreated in the entire intersection area of the connecting wires.

-report
 Prints out more slotting information for each wire. The default is not to print.

-report_wire_relationship *file_name*
 Prints out the bounding box of each slotted parent wire and its child wires to a named output file. The default is not to output.

-undo
 Restores slotted wires to the shape before last **slot_wire** run.

DESCRIPTION

Slots wide wires and contact arrays on selected nets by replacing them with thinner wires and smaller arrays.

You can change the internal application of DRC rules by using the **set_preroute_drc_strategy** command.

Slotting rules might be defined in a technology file, where each layer has its own rule in the format, shown in the following example:

```
SlotRule "" {layerNumber = 18 lengthThreshold = 20 widthThreshold = 10 minSlotWidth = 1 maxSlotWidth = 2 minSlotLength = 8 maxSlotLength = 16 minSideSpace = 1 maxSideSpace = 5 minSideClearance = 1 maxSideClearance = 5 minEndSpace = 1 maxEndSpace = 2 minEndClearance = 1 maxEndClearance = 2 }
```

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example slots wires of nets VDD and VSS, with wire width threshold of 10.0, length threshold of 20.0, open slot width of 5.0, open slot length of 10.0, applied to all layers:

```
prompt> slot_wire -nets {VDD VSS} \  
-cutwidth {all_layers 10} \  
-cutlength {all_layers 20} \  
-width {all_layers 5} \  
-length {all_layers 10}
```

The following example slots wires of nets VDD and VSS, with wire width threshold of 10.0, length threshold of 20.0, open slot width of 5.0, open slot length of 10.0, applied to METAL2 layer; width threshold of 8.0, length threshold of 30.0, open slot width of 2.0, open slot length of 5.0, applied to METAL3 layer:

```
prompt> slot_wire -nets {VDD VSS} \  
-cutwidth {METAL2 10 METAL3 8} \  
-cutlength {METAL2 20 METAL3 30} \  
-width {METAL2 5 METAL3 2} \  
-length {METAL2 10 METAL3 5}
```

SEE ALSO

set_preroute_drc_strategy(2)

snap_objects

Snaps objects to the default snap type. An object is snapped onto the nearest position that satisfies the snapping constraint.

SYNTAX

```
status snap_objects
[-snap_pin_to_edge]
objects
```

ARGUMENTS

-snap_pin_to_edge

Snaps terminals and soft macro pins to the closest relevant boundary.

This option causes a terminal to snap to the closest cell boundary, if it is not already on a cell boundary. A soft macro pin will be also snapped to the closest edge of the soft macro to which it belongs, if it is not already on a valid soft macro edge.

This option does not do anything if no terminals or soft macro pins are among the arguments.

objects

Argument is a collection of movable objects.

DESCRIPTION

Snap objects to the default snap types according to the object types of individual arguments.

The default snap type can be found with `get_object_snap_type`. It can also be set with `set_object_snap_type`. Except for snapping pins (soft macro pins or top-level pins) onto wire tracks, there is no built-in interaction with any part of the legalization code. When soft macro or top-level pins are snapped to wire tracks, care is taken to avoid shorts with other pins or wires, if at all possible. Furthermore, pins will not be snapped to wire tracks beyond the end point of a soft macro (or cell) edge.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example snaps pin "abc" to the nearest wire track if the pins are not already on a wire track. The pin will also be centered on the wire track.

```
prompt> get_object_snap_type -type pin
wiretrack
prompt> snap_objects [get_pins "abc"]
```

In following example the standard cell "abc" is first snapped to the minimum grid and then to the nearest cell row.

```
prompt> get_object_snap_type -type cell  
litho  
prompt> snap_objects [get_cells "abc"]  
prompt> set_object_snap_type -type cell -snap row  
litho  
prompt> get_object_snap_type -type cell  
row  
prompt> snap_objects [get_cells "abc"]
```

SEE ALSO

```
set_object_snap_type(2)  
get_object_snap_type(2)  
set_user_grid(2)
```

sort_collection

Sorts a collection based on one or more attributes, resulting in a new, sorted collection. The sort is ascending by default.

SYNTAX

```
collection sort_collection  
[-descending] collection1 criteria
```

Data Types

<i>collection1</i>	collection
<i>criteria</i>	list

ARGUMENTS

-descending
Indicates that the collection is to be sorted in reverse order. By default, the sort proceeds in ascending order.

collection1
Specifies the collection to be sorted.

criteria
Specifies a list of one or more application or user-defined attributes to use as sort keys.

DESCRIPTION

You can use **sort_collection** to order the objects in a collection based on one or more attributes. For example, to get a collection of leaf cells increasing alphabetically, followed by hierarchical cells increasing alphabetically, sort the collection of cells using the **is_hierarchical** and **full_name** attributes as *criteria*.

In an ascending sort, Boolean attributes are sorted with those objects first that have the attribute set to *false*, followed by the objects that have the attribute set to *true*. In the case of a sparse attribute, objects that have the attribute come first, followed by the objects that do not have the attribute.

Sorts are ascending by default. The **-descending** option reverses the order of the objects.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sorts a collection of cells based on hierarchy, and adds a

second key to list them alphabetically. In this example, cells i1 and i2 are hierarchical, and u1 and u2 are leaf cells. Because **is_hierarchical** is a Boolean attribute, those objects with the attribute set to **false** are listed first in the sorted collection.

```
prompt> set zcells [get_cells {o2 i2 o1 i1}]
{"o1", "i2", "o1", "i1"}
prompt> set zsort [sort_collection $zc {is_hierarchical full_name}]
{"o1", "o2", "i1", "i2"}
```

SEE ALSO

`collections(2)`

sort_fp_pins

Sorts the specified collection of pins into alphabetic order by name or in reverse alphabetic order if `-reverse` is specified.

SYNTAX

```
status sort_fp_pins
[-reverse]
pins
```

ARGUMENTS

```
-reverse
    reverse sort

pins
    collection of pins
```

DESCRIPTION

Sorts the specified collection of pins into alphabetic order by name or in reverse alphabetic order if `-reverse` is specified.

Note: supplied pins must belong to a single soft macro

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

Sort matching pins in reverse order:

```
> sort_fp_pins [get_pins -filter "is_hierarchical==true" *clk*]
```

SEE ALSO

`create_fp_pins(2)`

source

Read a file and evaluate it as a Tcl script.

SYNTAX

```
string source  
[-echo] [-verbose] [-continue_on_error] file
```

Data Types

file string

ARGUMENTS

-echo

Echoes each command as it is executed. Note that this option is a non-standard extension to Tcl.

-verbose

Displays the result of each command executed. Note that error messages are displayed regardless. Also note that this option is a non-standard extension to Tcl.

-continue_on_error

Don't stop script on errors. Similar to setting the shell variable `sh_continue_on_error` to true, but only applies to this particular script.

file

Script file to read.

DESCRIPTION

The **source** command takes the contents of the specified *file* and passes it to the command interpreter as a text script. The result of the source command is the result of the last command executed from the file. If an error occurs in evaluating the contents of the script, then the **source** command returns that error. If a return command is invoked from within the file, the remainder of the file is skipped and the source command returns normally with the result from the return command.

By default, source works quietly, like UNIX. It is possible to get various other intermediate information from the source command using the **-echo** and **-verbose** options. The **-echo** option echoes each command as it appears in the script. The **-verbose** option echoes the result of each command after execution.

NOTE: To emulate the behavior of the dc_shell **include** command, use both of these options.

The file name can be a fully expanded file name and can begin with a tilde. Under normal circumstances, the file is searched for based only on what you typed. However, if the system variable `sh_source_uses_search_path` is set to "true", the file is searched for based on the path established with the `search_path` variable.

The **source** command supports several file formats. The *file* can be a simple ascii script file, an ascii script file compressed with gzip, or a Tcl bytecode script, created by TclPro Compiler. The **-echo** and **-verbose** options are ignored for gzip formatted files.

EXAMPLES

This example reads in a script of aliases:

```
prompt> source -echo aliases.tcl
alias q quit
alias hv {help -verbose}
alias include {source -echo -verbose}
prompt>
```

SEE ALSO

```
search_path(3)
sh_source_uses_search_path(3)
sh_continue_on_error(3)
```

split_clock_gates

Replicates (splits) integrated clock-gating cells that have timing violations on the enable pin.

SYNTAX

status **split_clock_gates**

ARGUMENTS

The **split_clock_gates** command has no arguments.

DESCRIPTION

For a placed design, **split_clock_gates** automatically replicates (splits) certain integrated clock-gating cells (ICGs). When running the command on a design that the clock trees have already been synthesized, the tool exits with the CTS-680 error message.

The ICGs to be replicated must meet the following two requirements:

1. The ICG has a timing violation on the enable pin.
2. The ICG drives a large enough fanout that it requires at least one level of clock buffers between the ICG and the registers that are driven by the ICG. This is determined by internal synthesis of the clock trees in the design.

When the requirements are met, the command replicates the ICGs by invoking the **split_clock_net** command automatically. At the end of **split_clock_net**, all clock trees will be removed.

ICGs that have the `is_fixed` attribute will not be replicated. If you want to replicate such ICGs that have timing violations, you should remove the `is_fixed` attribute before running **split_clock_gates**.

This command determines the enable pin of the ICGs based on the library attribute **clock_gate_enable_pin**. If the **split_based_on_test_pin** variable is set to true, the tool checks the timing on the test pin, instead of the enable pin, of the ICGs.

EXAMPLES

The following example replicates the violating clock-gating cells that do not have the **size_only** attribute.

```
prompt> set_split_clock_gates_options -honor_size_only
prompt> split_clock_gates
```

SEE ALSO

`split_clock_net(2)`

```
set_split_clock_gates_options(2)
reset_split_clock_gates_options(2)
report_split_clock_gates_options(2)
optimize_pre_cts_power(2)
```

split_clock_net

Duplicates the gates on the clock nets.

SYNTAX

```
status split_clock_net
[-objects net_or_gate_list]
[-gate_sizing]
[-gate_relocation]
[-split_any_cell_type]
[-split_intermediate_level_clock_gates]
[-operating_condition min | max | min_max]
[-isolate_float_pins]
[-drive_ungated_registers]
[-estimated_early_delay float]
```

Data Types

net_or_gate_list list

ARGUMENTS

-objects *net_or_gate_list*
Duplicates the clock gates that you specify in the list. When you specify a net name, all the clock gates on that net are duplicated. Integrated clock-gating cells are duplicated by default.

-gate_sizing
Sizes clock gates.

-gate_relocation
Switches off gate relocation.

-split_any_cell_type
Splits any gates of cell type.

-split_intermediate_level_clock_gates
Splits intermediate-level clock gates.

-operating_condition [min | max | min_max]
Instructs the clock tree synthesis engine to use the specified operating condition. The default is max.

-isolate_float_pins
Isolates float pins from the other pins while duplicating the clock gates.

-drive_ungated_registers
Adds a buffer to drive ungated registers.

DESCRIPTION

This command duplicates the clock gates you specify. If a list of nets is provided, all the clock gates on the given nets are duplicated. The command supports the duplication of integrated clock-gating cells by default. If the **cts_split_any_gate** variable is set to true, all clock gate duplication is enabled. Using the variable is equivalent to using the **-split_any_cell_type** option. If the **cts_split_intermediate_level_clock_gates** variable is set to true, the intermediate level of clock gates duplication is enabled. Using the variable is equivalent to using the **-split_intermediate_level_clock_gates** option. You can use the **-isolate_float_pins** option to separate float pins from the other pins.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example duplicates and resizes all the clock gates on net1 and net2.

```
prompt> split_clock_net -objects {net1 net2} -gate_sizing
```

SEE ALSO

```
compile_clock_tree(2)
```

split_mw_lib

Splits the top-level Milkyway design into different libraries. Splitting the library hierarchically facilitates concurrent design.

SYNTAX

```
status split_mw_lib
[-check_only]
[-to_dir path]
[-lib_prefix prefix]
-from_library mw_lib
mw_top_design
```

Data Types

<i>path</i>	string
<i>prefix</i>	string
<i>mw_lib</i>	string

ARGUMENTS

-check_only
Prints out the list of Milkyway designs, views, and versions but does not copy information to the destination library.

-to_dir *path*
The destination path of the new libraries. This path can be given as an absolute path or as a relative path. Relative paths will be resolved relative to the location of the library specified by **-from_library**. If this option is not given, the new libraries will be created at the same level as the given design library.

-lib_prefix *prefix*
This sets the prefix of the new libraries that are created. The name of the new libraries being created is determined by appending the *prefix* with *design_name*. If this option is not used, the default prefix "lib_" is used.

-from_library *mw_lib*
Specifies the name of the design library that contains the top-level Milkyway design that you want to split.

mw_top_design
Specifies the name of the top-level Milkyway design that you want to split.

DESCRIPTION

This command creates new design libraries in the directory specified by the **-to_dir** option. Each new library contains a single Milkyway design. The name of the new design libraries are determined by appending the *prefix* with the *design_name*. If the **-lib_prefix** option is not specified, the default prefix "lib_" is used. If the top-level Milkyway design contains a subdesign that is not a soft macro, such as a hard

macro or a standard design that is located in the design library, these subdesigns are copied to the destination design library containing the top-level design to maintain the original top-level design structure. No other Milkyway designs from the reference library are copied to the top-level design library.

All new libraries inherit the library information from the original library, including the technology file and reference library path.

After splitting, the new Milkyway designs include all information from the original Milkyway designs. Therefore, after splitting, you can perform block-level design concurrently on the resultant libraries.

After splitting, the new Milkyway designs have their versions set to 1.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example splits the chip_top design and its subdesigns from the myDesign design library to new design libraries.

```
prompt> split_mw_lib -check_only -from_library myDesign \  
        chip_top
```

Library Files will be copied:

```
    From: /Training/myDesign  
    To: /Training/lib_chip_top
```

Cells will be copied:

```
    From: /Training/myDesign  
    To: /Training/lib_chip_top
```

```
cell: chip_top.CEL;2
```

Library Files will be copied

```
    From: /Training/myDesign  
    To: /Training/lib_nand_macro
```

Cells will be copied:

```
    From: /Training/myDesign  
    To: /Training/lib_nand_macro
```

```
cell: nand_macro.FRAME;2
```

```
cell: nand_macro.CEL;1
```

```
1
```

SEE ALSO

copy_mw_cel(2)

save_mw_cel(2)

split_net

Splits routing objects on one net into many other nets.

SYNTAX

```
status split_net  
-start | list_of_nets | -batch  
[-cells cell_list]
```

Data Types

list_of_nets list

ARGUMENTS

-start

Clears the generated files `split_net_batch.tcl` and `split_net_batch`. The **-batch**, **list_of_nets**, and **-start** options are mutually exclusive: you can use only one.

list_of_nets

Specifies that the routing on the first specified net should be split over the remaining nets in the list. This option is used in the generated `split_net_batch.tcl` file. The **-batch**, **list_of_nets**, and **-start** options are mutually exclusive: you can use only one. This option can only be used in conjunction with **-cells** option.

-batch

Splits the routing on the original nets when you use it after a sequence of **insert_buffer -on_route -repeater_distance** commands. The **-batch**, **list_of_nets**, and **-start** options are mutually exclusive: you can use only one.

-cells *cell_list*

Specifies the cells where the routes needs to be broken which is now connected to the nets for which the routes are distributed This option is used in the generated `split_net_batch.tcl` file. This option can only be used in conjunction with **list_of_nets** option.

DESCRIPTION

This command re-uses the routing that was on a net before it was buffered by the execution of the **insert_buffer -on_route -repeater_distance** or **insert_buffer -on_route -divide_load_by** command.

The design must be routed and extracted in order to use the **split_net -batch** command.

The **insert_buffer -on_route -repeater_distance**, **legalize_placement**, and **split_net** commands are used together to split nets with routing into many nets using the same routing. The `split_net_batch.tcl` file is generated to hold the sequence of nets, so

that the entire set of nets can be split in a batch after legalization. By splitting the nets in a batch, the router is started only once and the log file is much shorter. The `split_net_batch.tcl` file (and another temporary file named `split_net_batch`) are automatically deleted.

If the nets that you are buffering have wires or vias typed "BUS", then before **`split_net -batch`** you need to allow the router to reroute those nets with this command:

```
set_parameter -name rerouteBus -value 1 -module droute
```

If the design is only partially routed and you want to work on only those nets mentioned by **`split_net`**, then you may need this command, to prevent routing fixes on other (unrouted) nets:

```
set_parameter -name splitNetConnBrokenNet -value 0 -module droute
```

Helpful hint: You might want to make a snapshot of the design before inserting buffers on nets because the process of legalization might move some buffers off the original route and cause excess routing during the execution of the **`split_net -batch`** command. If you do not like the result of splitting, you can rerun the script on the snapshot and then edit the `split_net_batch.tcl` file before calling **`split_net -batch`**. An entire **`split_net`** command can be deleted or an individual net (not the first) can be deleted from a list of nets. The routing for the deleted nets must then be completed by using the **`route_eco`** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows how two **`insert_buffer -on_route -repeater_distance`** commands generate a file called `split_net_batch.tcl` in the current directory. This file contains **`split_net`** commands specifying which nets should have their routing split at a later time.

The **`legalize_placement`** command is called to put the new buffers (or inverters) in legal locations.

The **`split_net -batch`** command reads the `split_net_batch.tcl` file and reuses the routing from the original nets and adds the routing necessary to reach the legalized buffers.

```
prompt> insert_buffer [get_net long_net] slow/BUFX4 -on_route -repeater_distance 50  
-new_net_names ibn -new_cell_names ibc  
prompt> insert_buffer [get_net longer_net] slow/BUFX4 -on_route -  
repeater_distance 80  
-new_net_names ibn -new_cell_names ibc  
prompt> legalize_placement -cells [get_cells -hier ibc*]
```

```
prompt> split_net -batch
```

In the following example, the `split_net_batch.tcl` file has the following format:

```
prompt> split_net {long_net ibn ibn_1 ibn_2} -cells {ibc ibc_1 ibc_2}  
prompt> split_net {longer_net ibn_3 ibn_4 ibn_5 ibn_6}  
-cells {ibc_3 ibc_4 ibc_5 ibc_6}
```

You can delete commands or nets from this file before applying the **split_net -batch** command, to avoid generating undesirable results.

If you do not want to use the commands in `split_net_batch.tcl`, you can use the **split_net -start** command to start over with a new sequence of **insert_buffer** commands, as shown below:

```
prompt> split_net -start  
prompt> insert_buffer [get_net fanout_net] slow/BUFX4 -on_route -divide_load_by 2  
prompt> insert_buffer [get_net signal_net] slow/BUFX4 -on_route -divide_load_by 3  
prompt> legalize_placement -cells [get_cells -hier eco*]  
prompt> set_parameter -name rerouteBus -value 1 -module droute  
prompt> split_net -batch
```

SEE ALSO

`insert_buffer(2)`

split_objects

Splits one or more objects at the specified x or y coordinate or by a specified line.

SYNTAX

```
new_objects split_objects
{-x float | -y float | -line line}
[-gap float]
objects
```

Data Types

objects collection

ARGUMENTS

-x float

Specifies the x-coordinate at which to split the objects.
This is essentially the same as splitting by a vertical line from (x,-infinity) to (x,+infinity).

-y float

Specifies the y-coordinate at which to split the objects.
This is essentially the same as splitting by a horizontal line from (-infinity, y) to (+infinity, y).

-line line}

Specifies the line to use to split the objects.
If the specified line is not horizontal or vertical, the longest direction of the line is used to specify the direction and the average of the points is used for the position. For example, a line from (10,10) to (20,30) becomes a vertical line at the x value 15 ((10 + 20)/2).

Note: Only objects whose bounding box intersect the line are split.

-gap float

Specifies the gap to leave between the cut objects.
The gap is centered at the cut point and is either horizontal or vertical depending on the cut direction.

objects

Specifies the objects to be split, which can be net shapes, wiring keepouts, placement keepouts or voltage areas.

RETURNS

new_objects

Contains a list of the new split objects.

Note: The command is guaranteed to return the new list of objects after the split. Objects that are not split are returned unchanged and objects that are split have the new split objects returned. For example, if you split two wires

(W1 and W2) by a line that causes W2 to be replaced by two new wires (W2_1 and W2_2), the command returns three wires (W1, W2_1 and W2_2).

DESCRIPTION

This command splits one or more objects at the specified x or y coordinate or the specified line.

See the man page for the `get_edit_property` command for details of which objects can be split in x and y directions.

Note: Snapping is done automatically using the global snap settings.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example splits the currently selected objects at an x-coordinate of 100 and selects the new objects.

```
prompt> change_selection [split_objects -x 100 [get_selection]]
```

The following example splits the currently selected objects using a vertical line.

```
prompt> change_selection [split_objects -line {100 0 100 100} \  
[get_selection]]
```

SEE ALSO

`move_objects(2)`
`remove_objects(2)`
`resize_objects(2)`
`rotate_objects(2)`
`align_objects(2)`
`distribute_objects(2)`
`expand_objects(2)`
`flip_objects(2)`
`set_object_snap_type(2)`
`get_object_snap_type(2)`

spread_spare_cells

Places the specified cells evenly throughout a rectilinear region.

SYNTAX

```
status spread_spare_cells  
cells  
[-bbox spare_cells_region  
  | -poly {{x1 y1} {x2 y2} ...}]
```

Data Types

<i>cells</i>	collection
<i>spare_cells_region</i>	rectangle
<i>x1</i>	float
<i>y1</i>	float
<i>x2</i>	float
<i>y2</i>	float

ARGUMENTS

cells
Specifies a collection of spare cells to spread.

-bbox *spare_cells_region*
Specifies the rectangular region in which to spread spare cells.

-poly {{x1 y1} {x2 y2} ...}
Specifies the rectilinear region in which to spread spare cells, given as a list of x,y coordinates for the lower-left and upper-right corners.

DESCRIPTION

This command spreads a collection of spare cell instances over a given rectilinear region. The cells in the collection must be marked as spare cell by using the **set_attribute** command or created by using the **insert_spare_cells** command. The command ignores a cell that is not properly marked.

The placement of cells is not guaranteed to be legal. You should run the **legalize_placement** command to get a legal placement, if necessary.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example put cells of pattern *spare* inside a rectangular region.

```
prompt> spread_spare_cells\  
-bbox {{10.8 20.3} {40.4 50.1}} [get_cells spare*]
```

The following example put cells *spareA* and *spareB* inside a polygonal region.

```
prompt> spread_spare_cells\  
-poly {{10.8 20.3} {40.4 20.3} {40.4 50.1} \  
{25.0 50.1} {25.0 30.3} {10.8 30.3} {10.8 20.3}} \  
"spareA spareB"
```

SEE ALSO

```
insert_spare_cells(2)  
set_attribute(2)
```

spread_zrt_wires

Spreads the wires in the current design to prevent shorted nets.

SYNTAX

```
status spread_zrt_wires
[-min_jog_length min_ratio]
[-pitch number_of_pitches]
[-timing_preserve_nets {collection_of_nets}]
[-timing_preserve_setup_slack_threshold slack_value]
[-timing_preserve_hold_slack_threshold slack_value]
```

Data Types

<i>min_ratio</i>	integer
<i>number_of_pitches</i>	float
<i>collection_of_nets</i>	collection
<i>slack_value</i>	float

ARGUMENTS

-min_jog_length *min_ratio*
Sets the minimum value of the ratio of jog length to layer pitch. By default, the *min_ratio* value is 2.

-pitch *number_of_pitches*
Sets the spread value in terms of number of pitches. By default, the *num_of_pitches* value is 0.5.

-timing_preserve_nets {*collection_of_nets*}
Specifies the nets to preserve timing on.
If you specify this option, the command considers the specified nets as timing critical and skips wire spreading on these nets and the neighboring nets.

-timing_preserve_setup_slack_threshold *slack_value*
Specifies the setup slack threshold for timing critical nets.
If you specify this option, the command skips wire spreading on all nets with a worst setup slack less than or equal to this value and the neighboring nets.

-timing_preserve_hold_slack_threshold *slack_value*
Specifies the hold slack threshold for timing critical nets.
If you specify this option, the command skips wire spreading on all nets with a worst hold slack less than or equal to this value and the neighboring nets.

DESCRIPTION

This command spreads the wires in the current design to prevent shorted nets. When spreading, a piece of the original wire is broken and pushed away and jog wires are created at both ends to maintain a connection.

Multicorner-Multimode Support

This command uses information from active scenarios only.

EXAMPLES

The following example performs wire spreading using the **-min_jog_length** and **-pitch** options.

```
prompt> spread_zrt_wires -pitch 0.5
```

SEE ALSO

```
route_zrt_detail(2)  
widen_zrt_wires(2)
```


stretch_wire

Stretches given wires or contacts horizontally or vertically without loosing connections.

SYNTAX

```
status stretch_wire  
objects  
{-x distance | -y distance}  
[-snap_to_track yes | no | halftrack]  
[-jog_layer same | up | down]  
[-auto_repair_net]  
[-end_point_range distance]  
[-undo]
```

ARGUMENTS

objects

Collection of wires or contacts.

-x *distance*

-y *distance*}

Select the delta x or y by which to move given wires or contacts. All distances are in unit of micron.

-snap_to_track yes | no | halftrack

Select snap to track: *yes* | *no* | *halftrack*. If this option is skipped, *yes* is default.

-jog_layer same | up | down

Selects the jog layer for jogs in the wire.

same means same layer as the given wires.

up means layer one level higher than the layer of the given wires.

down means layer one level below the layer of the given wires.

If this option is skipped, **same** is default.

-auto_repair_net

Select trim redundant net objects or antenna and reconnect them in a symbolic connection model.

-end_point_range *distance*

Specifies an endpoint extension range. If a wire extends past a contact but is less than this range, the connection is treated as an endpoint connection. If the contact is stretched shorter, the wire stretches with the contact. If the extension past the contact is greater than the endpoint value, the connection is treated as a T-connection and the wire is not trimmed. Default is 0.0

-undo

Restores the wires to the state that was before the last **stretch_wire** run.

DESCRIPTION

Stretches given wires horizontally or vertically without losing connections. You can change more than one wire at a time by providing collection of several wires. Signal wires are stretched according to the center-line.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

To move selected wires in x direction with auto repair and end point ranges options
`.nf stretch_wire -x 10.0 -jog_layer same -snap_to_track halftrack -auto_repair_net -
end_point_range 0.1 [get_selection]`

SEE ALSO

`get_selection(2)`
`get_net_shapes(2)`
`get_vias(2)`

sub_designs_of

Gets the subdesigns according to the options.

SYNTAX

```
collection sub_designs_of  
[-hierarchy]  
[-in_partition | -partition_only]  
[-dt_only | -ndt_only]  
[-multiple_instances | -single_instances]  
[-names_only]  
design
```

ARGUMENTS

-hierarchy

Returns all subdesigns in the hierarchy of the specified design. By default, only the children of the specified design are returned.

-in_partition

Returns only subdesigns that are part of the partition containing the specified design.

This option and the **-partition_only** option are mutually exclusive.

-partition_only

Looks only at subdesigns that are compile partitions (subdesigns that have the **in_partition** attribute set). Specifically, it returns the direct child compile partitions of the specified design (even if they are more than one level away). With the **-hierarchy** option, it returns all compile partitions in the hierarchy below the specified design.

This option and the **-in_partition** option are mutually exclusive.

-dt_only

Returns only subdesigns that are at least instantiated once with the **dont_touch** attribute set.

This option and the **-ndt_only** option are mutually exclusive.

-ndt_only

Returns only subdesigns that are at least instantiated once with the **dont_touch** attribute not set.

This option and the **-dt_only** option are mutually exclusive.

-multiple_instances

Returns only subdesigns that are instantiated multiple times.

This option and the **-single_instances** option are mutually exclusive.

-single_instances

Returns only subdesigns that are instantiated once.

This option and the **-multiple_instances** option are mutually exclusive.

-names_only

Returns a list of design names, instead of a collection.

design

Specifies the design whose subdesigns will be returned.

DESCRIPTION

The **sub_designs_of** command returns a collection of subdesigns of the specified design (or a Tcl list of design names if the **-names_only** option is specified). By default, the command returns the hierarchical children of the specified design. You can return all subdesigns in the hierarchy by specifying the **-hierarchy** option. In addition, you can filter the results in the following ways:

- subdesigns that are the top-level of a partition (**-partition_only**)
- subdesigns within the partition that contains the specified design (**-in_partition**)
- subdesigns that are at least instantiated once with the **dont_touch** attribute set (**-dt_only**)
- subdesigns that are instantiated at least once with the **dont_touch** attribute not set (**-ndt_only**)
- subdesigns that have multiple instances (**-multiple_instances**)
- subdesigns that have a single instance (**-single_instances**)

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

SEE ALSO

get_attribute(2)
set_attribute(2)
set_dont_touch(2)
sub_instances_of(2)

sub_instances_of

Gets the subinstances according to the options.

SYNTAX

```
collection sub_instances_of
[-hierarchy]
[-in_partition] [-partition_only]
[-dt_only] [-ndt_only]
[-of_references reference_list]
[-master_instance]
[-names_only]
design
```

ARGUMENTS

-hierarchy

Returns all subinstances in the hierarchy of the specified design.
By default, only the children of the specified design are returned.

-in_partition

Returns only subinstances that are part of the partition that contains the specified design.
This option and the **-partition_only** option are mutually exclusive.

-partition_only

Looks only at subinstances that are the top-level designs of compile partitions (subdesigns that have the **in_partition** attribute set). Specifically, it returns the instances that reference the next direct child compile partitions below the specified design (even if the instances are more than one level of hierarchy away). With the **-hierarchy** option, it returns all compile partition instances in the hierarchy below the specified design. This option and the **-in_partition** option are mutually exclusive.

-dt_only

Returns only subinstances that have the **dont_touch** attribute set.
This option and the **-ndt_only** option are mutually exclusive.

-ndt_only

Returns only subinstances that do not have the **dont_touch** attribute set.
This option and the **-dt_only** option are mutually exclusive.

-of_references *reference_list*

Specifies the reference designs whose instances will be returned.
By default, the **sub_instances_of** command returns instances for all reference designs.

-master_instance

Returns only the master instance when a design has multiple instances.
By default, the **sub_instances_of** command returns all instances of a multiply instantiated design.

-names_only
Returns a list of instance names, instead of a collection.

design
Specifies the design whose subinstances will be returned.

DESCRIPTION

This command returns a collection of instances of the specified design (or a Tcl list of instance names, if the **-names_only** option is specified). By default, it returns only the direct children of the design, if the **-hierarchy** option is specified, all instances of the entire design subtree are returned. If the **-master_only** option is specified, it will return only one instance for multiply instantiated designs. It will pick the master instance (i.e. the one that has the **MasterInstance** attribute set). If none of the qualifying instances (according to the other options) is the master instance, any instance of that design will be returned.

In addition, you can filter the results in the following ways:

- subinstances that are the top-level of a partition (**-partition_only**)
- subinstances within the partition that contains the specified design (**-in_partition**)
- subinstances that have the **dont_touch** attribute set (**-dt_only**)
- subinstances that do not have the **dont_touch** attribute set (**-ndt_only**)
- instances of certain designs (**-of_references**)

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

SEE ALSO

`get_attribute(2)`
`set_attribute(2)`
`set_dont_touch(2)`
`sub_designs_of(2)`

suppress_message

Disables printing of one or more informational or warning messages.

SYNTAX

```
string suppress_message  
[message_list]
```

Data Types

message_list list

ARGUMENTS

message_list
 A list of messages to suppress.

DESCRIPTION

The **suppress_message** command provides a mechanism to disable the printing of messages. You can suppress only informational and warning messages. The result of **suppress_message** is always the empty string.

A given message can be suppressed more than once. So, a message must be unsuppressed (using **unsuppress_message**) as many times as it was suppressed in order for it to be enabled. The **print_suppressed_messages** command displays the currently suppressed messages.

EXAMPLES

When the argument to the **unalias** command does not match any existing aliases, the CMD-029 warning message displays. This example shows how to suppress the CMD-029 message:

```
prompt> unalias q*  
Warning: no aliases matched 'q*' (CMD-029)  
prompt> suppress_message CMD-029  
prompt> unalias q*  
prompt>
```

SEE ALSO

```
print_suppressed_messages(2)  
unsuppress_message(2)
```

swap_cell_locations

Swaps the locations of two cells.

SYNTAX

```
int swap_cell_locations
cell1
[-cell1_orient N | W | S | E | FN | FE | FS | FW]
[-cell2_orient N | W | S | E | FN | FE | FS | FW]
```

Data Types

cell1 collection

ARGUMENTS

cell1
Specify the first cell to be swapped. The collection should only have one cell.

-cell1_orient N | W | S | E | FN | FE | FS | FW
Specifies the orientation of first cell after swapping. By default, the other cells' orientation will be assumed.

-cell2_orient N | W | S | E | FN | FE | FS | FW
Specifies the orientation of second cell after swapping. By default, the other cells' orientation will be assumed.

DESCRIPTION

This command swaps the locations of two cells. Following Milkyway convention, the location of a cell refers to its lower left corner regardless of its orientation. The lower left coordinates of the two cells are swapped.

By default, the orientation of the cell after swap assumes the orientation of the other cell.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example swaps the locations of the two cells A and B:

```
prompt> swap_cell_locations A B
1
```

```
swap_cell_locations
2922
```


SEE ALSO

`move_objects(2)`

syntax_check

Enables or disables the Syntax Checker's `syntax_check` mode which checks commands for syntax errors.

SYNTAX

integer **syntax_check**
true | false

ARGUMENTS

true | false
Indicates whether to enable or disable the `syntax_check` mode. A value of **true** enables the mode and **false** disables the mode. Set the related command **context_check** to **false** before setting **syntax_check** to **true**.

DESCRIPTION

This command, with **context_check**, comprises a pair of commands that enable or disable the `syntax_check` or `context_check` modes of the `dc_shell` Syntax Checker. In the `syntax_check` mode, the command interpreter checks each command for the correct syntax. To find out whether either mode is enabled, retrieve the value of the corresponding status variable (**context_check_status** or **syntax_check_status**), by running either the **printvar syntax_check_status** or the **printvar context_check_status** command.

Using `syntax_check` mode, you can identify and correct syntax errors in a script file that you want to include, before you actually execute the file. The command interpreter checks for the existence of the included file and parses the file line by line, to check its syntax. As errors (spelling, typos, invalid or missing arguments) are found, the normal error messages are issued but syntax checking continues to the end of the file.

Note that the `syntax_check` mode examines each line only once; it does not iterate through loops. Thus, there is a possibility of receiving false error messages if names are not found that would be generated during normal execution. For example, the `syntax_check` mode might make variable assignments as they occur line by line in the script. In that case, the value of the variable might be the one that appears in the variable assignment that physically appears last in the code. This value might be different from the value that the variable would have if the code were executed.

For details about the features of the `syntax_check` and the `context_check` modes, see the *Design Compiler Command-Line Interface Guide*.

EXAMPLES

The following example enables the `syntax_check` mode, checks the file named `myfile.tcl` for syntax errors, and redirects the output to `myfile.out`.

```
prompt> syntax_check true
```

```
syntax_check  
2924
```

Syntax checker on.

```
prompt> source myfile.tcl > myfile.out
```

In the following example, you determine whether **context_check** is enabled before attempting to enable **syntax_check**. The status of the variable shows that **context_check** is currently enabled, probably by a previous operation. Disable **context_check** and then enable **syntax_check**.

```
prompt> printvar context_check_status  
true
```

```
prompt> context_check false  
Context checker off.
```

```
prompt> syntax_check true  
Syntax checker on.
```

SEE ALSO

`context_check_status(3)`

`syntax_check_status(3)`

synthesize_fp_rail

Synthesizes power networks or power switch arrays based on user-specified constraints. In a single run, you can synthesize either a power network for a single voltage design, power networks for a multivoltage design, or a power switch array.

SYNTAX for Single Voltage Power Network Synthesis

status **synthesize_fp_rail**

```
[-synthesize_power_plan]
[-synthesize_power_pads]
-nets nets
[-power_budget power]
[-analyze_power]
[-read_default_power_file power_file_name]
[-read_power_compiler_file power_compiler_report_file]
[-read_prime_power_file prime_power_report_file]
[-target_voltage_drop target_voltage]
[-lowest_voltage_drop]
[-pad_masters pad_reference_cells]
[-read_pad_instance_file pad_instance_file_name]
[-read_pad_master_file pad_reference_cell_file_name]
[-use_pins_as_pads]
[-use_strap_ends_as_pads]
[-top_level_only]
[-create_virtual_rails layer]
[-ignore_blockages]
[-honor_conn_view_layers conn_layer]
[-output_directory directory_name]
```

Data Types for Single Voltage Power Network Synthesis

<i>nets</i>	list
<i>power</i>	float
<i>power_file_name</i>	string
<i>power_compiler_report_file</i>	string
<i>prime_power_report_file</i>	string
<i>target_voltage</i>	float
<i>pad_reference_cells</i>	list
<i>pad_instance_file_name</i>	string
<i>pad_reference_cell_file_name</i>	string
<i>layer</i>	string
<i>conn_layer</i>	string
<i>directory_name</i>	string

SYNTAX for Multivoltage Power Network Synthesis

status **synthesize_fp_rail**

```
-synthesize_voltage_areas
| -synthesize_power_switch_array
```

```

[-voltage_areas voltage_areas]
[-power_budget power]
[-output_directory directory_name]
[-analyze_power]
[-pad_masters pad_reference_cells]
[-read_pad_instance_file pad_instance_file_name]
[-read_pad_master_file pad_reference_cell_file_name]
[-use_pins_as_pads]
[-use_strap_ends_as_pads]
[-top_level_only]
[-honor_conn_view_layers conn_layer]
[-read_power_compiler_file power_compiler_report_file]
[-read_prime_power_file prime_power_report_file]
[-read_default_power_file power_file_name]
[-output_directory directory_name]
[-voltage_supply float]
[-strap_ends_direction string]

```

Data Types for Multivoltage Power Network Synthesis

voltage_areas list

ARGUMENTS

-synthesize_power_plan

Performs normal power network synthesis.

This is the default behavior if you do not specify any of the following options: **-synthesize_power_pads**, **-synthesize_voltage_areas**, and **-synthesize_power_switch_array**.

This option can be specified together with the **-synthesize_power_pads** option to perform simultaneous power network and power pad synthesis.

This option is mutually exclusive with the **-synthesize_voltage_areas** and **-synthesize_power_switch_array** options.

-synthesize_power_pads

Performs power pad synthesis.

This option can be specified together with the **-synthesize_power_plan** option to perform simultaneous power network and power pad synthesis.

This option is mutually exclusive with the **-synthesize_voltage_areas** and **-synthesize_power_switch_array** options.

If you do not specify any of the following options, the tool performs normal power network synthesis: **-synthesize_power_plan**, **-synthesize_power_pads**, **-synthesize_voltage_areas**, and **-synthesize_power_switch_array**.

-nets *nets*

Specifies the nets on which to perform power network synthesis or power pad synthesis. This is a required option for power network synthesis (**-synthesize_power_plan**) and power pad synthesis (**-synthesize_power_pads**).

This option is not supported for multivoltage power network synthesis (**-synthesize_voltage_areas**) or power switch synthesis (**-synthesize_power_switch_array**).

-power_budget *power*

Specifies a total power budget for the design. The unit is milliwatts.

This power budget is allocated to each instance in the design based on the area of the instance area, unless the power in that instance is already defined. For a hierarchical block, the power budget is computed based on the sum of all cells and blocks inside it.

The power for an instance can be defined in the following ways: by performing power analysis (**-analyze_power**), by reading a default power file (**-read_default_power_file**), by reading a Power Compiler file (**-read_power_compiler_file**), or by reading a PrimePower file (**-read_prime_power_file**).

If the power is defined for one or more instances, the tool computes the sum of the defined instance power. If the sum of the defined instance power is greater than the total power budget, the tool ignores the total power budget. Otherwise, the tool allocates the remaining power budget among the instances without defined power.

For multivoltage power synthesis, this option is ignored if a power budget has already been specified for all voltage areas by the **set_fp_rail_voltage_area_constraints** command.

By default, the total power budget is 1000 mW unless you specify **-analyze_power**, in which case the default is 0.

-analyze_power

Perform power analysis on each instance by using the IC Compiler power analysis engine (the same engine that is used by the **report_power** command). Before using this option for power analysis, you can specify the toggle rate and static probability in the following ways:

- * Specify the default toggle rate by setting the **power_default_toggle_rate** variable. The default value is 0.1.
- * Specify the default static probability for primary inputs and black box outputs by setting the **power_default_static_probability** variable. The default value is 0.5.
- * Set the toggle rate and static probability for a specific object, such as a net, pin, or port, by using the **set_switching_activity** command.
- * Read the switching activity from a switching activity interchange format (SAIF) file by using the **read_saif** command.
- * Set the power value of a pin per toggle by using the **set_cell_internal_power** command.

Before performing power network synthesis (PNS) with **-analyze_power**, you should run the **report_power -cell -flat -nosplit** command. For more information, see the **report_power** man page.

You can use this option together with the **-read_default_power_file** option. The power specified in the default power file has a higher priority than that calculated by the **-analyze_power** option. If the power value for a given instance is both calculated and read from the default power file, the instance is assigned the power value from the default power file.

This option is mutually exclusive with the **-read_power_compiler_file** and **-read_prime_power_file** options.

-read_default_power_file *power_file_name*

Specifies the default power report file to use to assign power to each instance. The format of this file is as follows:

instance|master name power_value

The unit for the power value is mW.

For example, to specify a power value of 0.2 mW for instance X1 and a power value of 0.1 mW for reference cell abc.FRAME, the file would contain the following lines:

```
instance X1 0.2
master abc.FRAME 0.1
```

You can generate a default power file by running the **synthesize_fp_rail** or **analyze_fp_rail** command. These commands generate a file called

You can use this option with the **-power_budget** and **-analyze_power** options. This option is mutually exclusive with the **-read_default_power_file** and **-read_prime_power_file** options.

-read_power_compiler_file *power_compiler_report_file*

Specifies the Power Compiler report file to use to assign power to each instance.

You can use this option with the **-power_budget** option.

This option is mutually exclusive with the **-analyze_power**, **-read_default_power_file** and **-read_prime_power_file** options.

-read_prime_power_file *prime_power_report_file*

Specifies the PrimePower or PrimeTime PX report file to use to assign power to each instance.

You can use this option with the **-power_budget** option.

This option is mutually exclusive with the **-analyze_power**, **-read_default_power_file** and **-read_power_compiler_file** options.

-target_voltage_drop *target_voltage*

Specifies a target voltage (IR) drop value for single-voltage power network synthesis to succeed. The unit is millivolts.

By default, 10 percent of the supply voltage value is used as the target voltage drop.

This option is ignored for multivoltage power network synthesis and power switch synthesis. You must specify the target voltage drop for each voltage area by using the **set_fp_rail_voltage_area_constraints** command.

-lowest_voltage_drop

Allows single-voltage power network synthesis to generate the power network that has the lowest possible voltage (IR) drop based on user-specified power network constraints.

This option is ignored for multivoltage power network synthesis and power switch synthesis. To specify this option for a voltage area, you must use the **set_fp_rail_voltage_area_constraints** command.

-pad_masters *pad_reference_cells*

Specifies the pad reference cells and their associated nets as follows:

[*net_name*:]*pad_reference_cell*

If the net name is not specified, the pad reference cell is used for both power and ground nets.

By default, PNS assumes that all pads logically connected to signal 1 or 0 are power pads.

You can use this option together with **-read_pad_instance_file** or **-read_pad_master_file**.

-read_pad_instance_file *pad_instance_file_name*

Specifies the power and ground (PG) pad instances by using an input file. The instances specified in the file are used as PG pads. The format is:

instance_name [*net_name*]

For example, to specify that the VDD1 pad instance is used for the VDD net, enter

VDD1 VDD

If you do not specify a net name, the command assumes that the specified pad instance is used for both power and ground nets.

This option and **-read_pad_master_file** are mutually exclusive; you can specify only one. You can use this option together with **-pad_masters**.

-read_pad_master_file *pad_reference_cell_file_name*

Specifies the PG pad reference cells by using an input file. The reference cells specified in the file are used as PG pad reference cells. The format is:

reference_cell_name *net_name*

For example, to specify that the VDD.FRAME reference cell is used for the VDD net, enter

VDD.FRAME VDD

If you do not specify a net name, the command assumes that the specified pad reference cell is used for both power and ground nets.

This option and **-read_pad_instance_file** are mutually exclusive; you can specify only one. You can use this option together with **-pad_masters**.

-use_pins_as_pads

Allows power network synthesis to use PG pins on the design boundary as PG pads. This option is useful in block-level power network synthesis.

-use_strap_ends_as_pads

Allows power network synthesis to use the ends of straps at the design boundary or core ring as power pads. This option is useful at the design planning stage when PG pads have not yet been created.

-top_level_only

Ignores cell instances inside soft macros during power network synthesis. By default, power network synthesis flattens all cell instances in each soft macro.

-create_virtual_rails *layer*

Creates pseudo-straps on the specified metal layer to connect standard cell pins in the row grid. These virtual straps are used only for IR drop analysis in power network synthesis. They will not be created after commit.

By default, metall is used to generate the pseudo-straps.

-ignore_blockages

Ignores any blockage for virtual pads to make a pseudo-connection to the power network.

By default, the tool honors blockages for virtual pads.

-honor_conn_view_layers *conn_layer*

Specifies the metal layers in which the connectivity (CONN) views are

honored. To honor the CONN view in all metal layers, use the keyword **all** as the layer name.

By default, CONN views are not honored in any metal layer.

-output_directory *directory_name*

Specifies the directory in which to store the voltage drop and electromigration results.

The default is `./pna_output`.

-synthesize_voltage_areas

Performs multivoltage power network synthesis on voltage areas that have been specified by using the **set_fp_rail_voltage_area_constraints** command.

This option is mutually exclusive with the **-synthesize_power_plan**, **-synthesize_power_pads**, and **-synthesize_power_switch_array** options.

If you do not specify any of the following options, the tool performs normal power network synthesis: **-synthesize_power_plan**, **-synthesize_power_pads**, **-synthesize_voltage_areas**, and **-synthesize_power_switch_array**.

-synthesize_power_switch_array

Performs power switch synthesis on the specified powered-down voltage area.

This option is mutually exclusive with the **-synthesize_power_plan**, **-synthesize_power_pads**, and **-synthesize_voltage_areas** options.

If you do not specify any of the following options, the tool performs normal power network synthesis: **-synthesize_power_plan**, **-synthesize_power_pads**, **-synthesize_voltage_areas**, and **-synthesize_power_switch_array**.

-voltage_areas *voltage_areas*

Specifies the voltage areas in which to perform multivoltage power network synthesis or power switch synthesis.

For multivoltage power network synthesis, you can specify multiple voltage areas. If you do not specify this option, power networks are synthesized in all voltage areas that have been specified by the **set_fp_rail_voltage_area_constraints** command.

For power switch synthesis, this is a required option and you can specify only a single voltage area.

-voltage_supply *float*

Specifies the supply voltage for the synthesized power nets in a single voltage design. The unit is volts.

By default, the supply voltage is 1.5 V.

This option is ignored for multivoltage power network synthesis and power switch synthesis. You must specify the supply voltage for each voltage area by using the **set_fp_rail_voltage_area_constraints** command.

DESCRIPTION

This command allows you to synthesize a power network with a specified IR (voltage) drop and electromigration (EM) constraints, power budgets, and customized design style (for example, with or without a core ring, specific metal layers, stacked via, and so on). It can perform single-voltage power network synthesis, single-voltage power pad synthesis, multivoltage power network synthesis, or power switch synthesis.

Power network synthesis (PNS) enables you to preview an early power network at the

design planning stage.

After PNS, if the IR drop and electromigration meet the requirements, you can commit the synthesized power network into a real power network by using the **commit_fp_rail** command. To commit the synthesized power switches, you must run the `create_power_switch.tcl` script that is written to the `pna_output` directory when you perform power switch synthesis.

Multicorner-Multimode Support

This command is not supported in a multi-scenario design flow.

EXAMPLES

The following example shows how to synthesize a power network for the VDD and VSS nets with a power budget of 1000 mW and a supply voltage of 1.2 V. The pad reference cells are `pvdi.FRAM` for the VDD net and `pvdo.FRAM` for the VSS net.

```
prompt> synthesize_fp_rail -nets {VDD VSS} \  
-power_budget 1000 -voltage_supply 1.2 \  
-pad_masters "VDD:pvdi.FRAM VSS:pvdo.FRAM"
```

The following example shows how to synthesize a power network for the VDD and VSS nets. The power is analyzed by IC Compiler and the supply voltage 1.2 V.

```
prompt> synthesize_fp_rail -nets {VDD VSS} \  
-analyze_power -voltage_supply 1.2
```

The following example shows how to synthesize a power network in multiple voltage areas with different power supplies.

```
prompt> synthesize_fp_rail \  
-synthesize_voltage_area -voltage_area {MULTIPLIER DEFAULT_VA}
```

The following example shows how to synthesize power switches in the powered-down voltage area named `PWR_DN`.

```
prompt> synthesize_fp_rail \  
-synthesize_power_switch_array -voltage_area PWR_DN
```

SEE ALSO

```
commit_fp_rail(2)  
read_saif(2)  
report_power(2)  
set_cell_internal_power(2)  
set_fp_block_ring_constraints(2)  
set_fp_power_pad_constraints(2)  
set_fp_rail_constraints(2)
```

```
set_fp_rail_region_constraints(2)
set_fp_rail_voltage_area_constraints(2)
set_switching_activity(2)
power_default_static_probability(3)
power_default_toggle_rate(3)
```

trace_scan_chain

Traces scan chain in the current design.

SYNTAX

```
status trace_scan_chain  
[-from scan_input] [-to scan_output]
```

Data Types

```
scan_input      string  
scan_output    string
```

ARGUMENTS

```
-from scan_input  
    Specify the start point of the scan chain.  
  
-to scan_output  
    Specify the stop point of the scan chain.
```

DESCRIPTION

Traces the scan chain in the current design backwards from the user- specified scan-out pin to the scan-in pin. A correctly traced scan chain is updated in the Milkyway. Tracing through scan cells and single-input cells is automatic. Any other leaf cell requires the user to specify the scan-in and scan-out pins by using *set_scan_pin_type*. Arguments for the -from and -to options are instance pins or ports. After *trace_scan_chain*, *check_scan_chain* is recommended to do the verification.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows a new chain is generated.

```
prompt> trace_scan_chain -from TESTSI1 -to u0/u0/U2/A  
  
===== Scan Trace =====  
INFO: reading in existing scan chain group...  
Total number of edges in scan chain group is 1730  
Number of scan pins processed is 1725  
start tracing scan chain backward ...  
WARNING : Existing scan chain group ScanGroup_physopt_TESTSI1 will be replaced.
```

```
trace_scan_chain  
2934
```

```
INFO: creating new scan chain group <ScanGroup_physopt_TESTSI1>
Traced scan chain has 1730 edges
=== END Scan Trace ===
```

1

SEE ALSO

```
set_scan_pin_type(2)
check_scan_chain(2)
report_scan_chain(2)
```

translate_zrt_parameters

Translates the routing parameters to Zroute.

SYNTAX

```
status translate_zrt_parameters  
[-output filename]
```

Data Types

filename string

ARGUMENTS

-output *filename*
Writes the Tcl translation script to the specified file name.
By default, a Tcl script is not generated.

DESCRIPTION

The **translate_zrt_parameters** command does translation between Zroute options and routing parameters. A parameter will only be translated if it has been set by the user explicitly before the translation command. A parameter will also be translated if its value is different than the default value (e.g. the value has been saved with the cell), but only if the corresponding option to be translated to has not been set by the user explicitly.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example translates routing parameters to Zroute options.

```
prompt> translate_zrt_parameters -output translate_script.tcl
```

SEE ALSO

trim_fill_eco

Trims the metal fill. Use this command after you finish inserting metal fill and running ECO routing.

SYNTAX

```
status trim_fill_eco  
[-input fill_view_name]  
[-output fill_view_name]  
[-spacing_to_routing number]  
[-remove_vio_fill]  
[-from_metal metal_layer_name]  
[-to_metal metal_layer_name]
```

Data Types

<i>fill_view_name</i>	string
<i>number</i>	float
<i>metal_layer_name</i>	string

ARGUMENTS

-input *fill_view_name*
specify the fill view in which the fill will be trimmed, without specifying this option means the tool will read fills from the editing cell itself

-output *fill_view_name*
specify the fill view in which the trimmed fills will be saved, without specifying this option means the tool will save the fills to the editing cell itself

-spacing_to_routing *number*
the float value will be multiplied by the minimum spacing, and the result will be used to control the spacing between fill to routing, the fill within this spacing will be trimmed off. Default : 1.0

-remove_vio_fill
remove whole fill segment with DRC violation

-from_metal *metal_layer_name*
specify the name of the bottom layer to start the trimming of fill. The *metal_layer_name* must match a layer name in the database. If no *bottom_layer_name* is specified, *metal1* is the default bottom layer.

-to_metal *metal_layer_name*
specify the name of the top layer to end the trimming of fill. The *metal_layer_name* must match a layer name in the database. The corresponding layer should be the same as or above the bottom layer. If no *top_layer_name* is specified, the topmost metal layer of the design is the default top metal layer.

DESCRIPTION

When you have finished metal filling and the engineering change order has changed the routing, you can use this command to trim the metal fill and avoid design rule constraint violations.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example will read the fills from the editing cell and trim the fill and output to cell "aaa.fill" :

```
prompt> trim_fill_eco -output aaa.fill
```

The following example will read the fills from an existing fill view "bbb.fill" and trim the fill metal from layer METAL2 to METAL4 and output to cell "ccc.fill" :

```
prompt> trim_fill_eco -input bbb.fill -output aaa.fill -from_metal METAL2 -  
to_metal METAL4
```

SEE ALSO

`insert_metal_filler(2)`

unalias

Removes one or more aliases.

SYNTAX

```
string unalias  
pattern
```

ARGUMENTS

DESCRIPTION

The **unalias** command removes aliases created using the alias command.

EXAMPLES

The following command removes all aliases.

```
prompt> unalias *
```

The following command removes all aliases beginning with f, and the alias rt100.

```
prompt> unalias f* rt100
```

SEE ALSO

alias(2)

uncommit_fp_soft_macros

Transforms soft macros into plan groups.

SYNTAX

```
status uncommit_fp_soft_macros  
[-push_up_preroutes pg | none]  
[-remove_feedthroughs]  
[objects]
```

ARGUMENTS

-push_up_preroutes pg | none

Specifies command will push up just pg routing or no routing at all. If this option is not specified, the default behavior of uncommit_fp_soft_macros is to push up ALL routing, including both PG and detail routing. This option allows the user to choose JUST PG, or no routing at all. The user should be cautioned that the use of the "none" value along with this option will result in the loss of any existing child routing during the uncommit process, and the use of the "pg" option will result in the loss of any detail routing in the child cell. This option used with a route_type of pg will push up all P/G routing, regardless of whether it was originally a part of the plan group or whether it was a global type P/G net.

-remove_feedthroughs

Removes feedthrough ports on soft macros

objects

Collection of soft macros. If no collection is specified, then all soft macros are processed.

DESCRIPTION

This command transforms soft macros into plan groups. For each soft macro processed: the soft macro is disconnected and deleted, a plan group with the soft macro's boundary is created, and all of the plan group's standard cells are created and connected to their corresponding nets.

The command by default creates child level power meshes and detailed routing at its relative place in the flat design. The command can also has the capability to limit routing that is pushed up to just PG routing, or no routing at all. Unlike commit_fp_plan_groups (which automatically pushes down internal net routes but requires the -push_down_power_and_ground_straps option in order to include global P/G nets), uncommit_fp_soft_macros makes no distinction about what to push up (it pushes up both internal net routes and global net routes automatically), unless the -push_up_preroutes option is used to specify either just P/G type routes (including internal P/G net routes), or no routes. The user may wish to run push_up_fp_objects using route types prior to running uncommit_fp_soft_macros to push up just the global-type power and ground nets that may have been pushed down previously by either employing the -push_down_power_and_ground_straps during commit, or by running push_down_fp_objects after commit. The distinction is that if a port was added to

the soft macro as a result of a push-down routing operation (either `push_down_fp_objects` with routing of commit with the `-push_down_power_and_ground_straps` option), an explicit push-up using `push_up_fp_objects` will remove that port and then uncommit will produce a plan group without that port on the plan group. If it is left to uncommit to push up the global power or ground net routing, such an added port will remain on the plan group's hierarchical cell instance and an internal hierarchical power or ground net will be retained by the plan group. If that is not desired, then run `push_up_fp_objects` prior to uncommit so that global-type routes can be distinguished from internal type net routes. The command also allows the user to remove feedthrough ports on a given soft macro.

The command will push up everything in the soft macro, including hard move bounds and relative placement (RP) groups.

The command takes a collection of soft macros as an argument. The command will process the soft macros in the collection. If no collection is given, then all soft macros are processed.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

Uncommit entire design and push up all routing

```
uncommit_fp_soft_macros
```

Uncommit entire design and push up just PG routing

```
uncommit_fp_soft_macros -push_up_preroutes pg
```

Uncommit selected soft macros, remove feed throughs, and push up no routing

```
uncommit_fp_soft_macros -push_up_preroutes none -remove_feedthroughs $SM_List
```

SEE ALSO

`commit_fp_plan_groups(2)`

`push_up_fp_objects(2)`

undefine_bus

Removes bus property from a port or net bus and makes its elements bit-blasted.

SYNTAX

```
status undefine_bus  
-type type_name  
-name full_bus_name
```

Data Types

```
type_name          string  
full_bus_name      string
```

ARGUMENTS

```
-type type_name  
    Specifies the type of object whose bus attribute is to be defined. Valid types  
    include port and net.  
    This argument is required.  
  
-name full_bus_name  
    Specifies the full path and the name of the bus. Note that the specified name  
    cannot be the same as any other bus. For example, port bus names must be  
    different from the names of ports, and net bus names must be different from  
    the names of nets.  
    This argument is required.
```

DESCRIPTION

The **undefine_bus** command collects a list of ports or nets and removes their bus property. Buses appear as multibit ports on a design or as multiwire nets in a design. Individual members of buses remain but become bit-blasted.

There is no need to specify the start and end index of the bus. The command will collect all port or net members based on the given bus name.

EXAMPLES

This example removes bus property from every member of port bus A.

```
prompt> report_bus [get_ports A]  
prompt> undefine_bus -type port -name A  
prompt> report_bus [get_ports A]
```

This example removes bus property from every member of net bus B.

```
prompt> report_bus [get_nets B]
```

```
prompt> undefine_bus -type net -name B  
prompt> report_bus [get_nets B]
```

This example removes bus property from every member of port bus U1/A in the MID subdesign. U1 is a unique instantiation of MID.

```
prompt> report_bus [get_pins U1/A]  
prompt> undefine_bus -type port -name U1/A  
prompt> report_bus [get_pins U1/A]
```

This example removes bus property from every member of net bus U1/NET_BUS in the MID subdesign. U1 is a unique instantiation of MID.

```
prompt> report_bus [get_nets U1/NET_BUS]  
prompt> undefine_bus -type net U1/NET_BUS  
prompt> report_bus [get_nets U1/NET_BUS]
```

SEE ALSO

```
get_nets(2)  
get_ports(2)  
get_pins(2)  
define_bus(2)  
report_bus(2)
```

undo

Undoes the last operation.

SYNTAX

```
status undo  
[-all]  
[-mark string]
```

ARGUMENTS

-all
Undo all operations in undo stack.

-mark *string*
Undo all operations in undo stack to the named mark.

DESCRIPTION

This command undoes one or more undoable operations.

The undone commands can be redone using the **redo** command.

EXAMPLES

The following example undoes the result of the **move_objects** command.

```
prompt> move_objects -to {1400 1600} [get_route_guides *]  
prompt> undo
```

SEE ALSO

```
redo(2)  
undo_mark(2)  
undo_config(2)
```

undo_config

Configures the undo stack.

SYNTAX

```
status undo_config
{-max_depth int | -max_memory int
 | -depth int
 | -memory int
 | -enable
 | -disable}
-enabled sbool
```

ARGUMENTS

- max_depth *int***
Set the maximum depth for the undo stack
Note: If the value is set to zero then maximum depth checking is disabled.
- max_memory *int***
Set the maximum memory usage for the undo stack
Note: If the value is set to zero then maximum memory checking is disabled.
- depth *int***
Set the current depth of the undo stack
- memory *int***
Set the current memory usage for the undo stack
- enable**
Re-enables the undo stack after it is disabled. See *-disable* option for more details.
- disable**
Disables the undo stack. The undo is cleared and no new commands are added to the stack until it is enabled.
This option is useful if the user is about to make a large number of changes which he/she knows that they are not going to undo. Disabling in this case will not waste memory and also speed up the command processing.

DESCRIPTION

This command configures the undo stack.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example clears the undo stack by setting the depth to zero:

```
> undo_config -depth 0
```

The following example disables undo, runs a complex command and re-enables undo:

```
> undo_config -disable  
> remove_objects [get_plan_groups *]  
> undo_config -enable
```

SEE ALSO

```
undo(2)  
redo(2)  
undo_mark(2)
```


undo_mark

Marks a position in the undo stack to undo back to.

SYNTAX

```
status undo_mark  
mark
```

ARGUMENTS

```
mark  
    Name of the mark to undo back to
```

DESCRIPTION

This command marks a position in the undo stack that can be undone back to with a single undo command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example adds a mark, does two moves, and then undoes them as a single operation.

```
prompt> undo_mark -mark move_mark  
prompt> move_objects -to {1400 1600} [get_cell alu1]  
prompt> move_objects -to {1600 1600} [get_cell mem1]  
prompt> undo move_mark
```

SEE ALSO

```
undo(2)  
redo(2)  
undo_config(2)
```

ungroup

Removes a level of hierarchy.

SYNTAX

```
status ungroup
cell_list | -all
[-prefix prefix_name]
[-flatten]
[-simple_names]
[-soft]
[-small n]
[-force]
[-start_level n]
[-all_instances]
```

Data Types

<i>cell_list</i>	list
<i>prefix_name</i>	string
<i>n</i>	integer

ARGUMENTS

cell_list
Specifies a list of cells in the current design that are to be ungrouped. The contents of these cells are brought up to the same level as the cells. You must specify either *cell_list* or **-all**, but not both.

-all
Indicates that all cells in the current design or current instance are to be ungrouped. You must specify either **-all** or *cell_list*, but not both.

-prefix *prefix_name*
Specifies the prefix to use in naming ungrouped cells. The default naming style is as follows:

cell_being_ungrouped/old_cell_name{number}
To provide a vestige of the former hierarchy, omit this option when using the **-flatten** option

-flatten
Indicates that the specified cell and all of its subcells are to be exploded recursively until all levels of hierarchy are removed.

-simple_names
Indicates that simple, non-hierarchical names are to be used for cells that are ungrouped. Unless this option is used, cells are given default hierarchical names. With this option, cells maintain their original names.

-soft
Indicates that the **group_name** attribute is to be removed on the specified

cells. With the **-soft** and **-flatten** options, the **group_name** attribute is removed recursively down the hierarchy of the selected cells. The **group_name** attribute specifies cell grouping constraints for layout placement tools.

-small *n*

Causes all subdesigns with less than the number of leaf cells specified by *n* to be ungrouped. This option implies the **-all** option.

-force

Causes all **dont_touch** hierarchical cells to be flattened. The **dont-touch** attribute is inherited by their leaf cells. This option can only be used when either **-all** or *cell_list* is specified.

-start_level *n*

Causes all hierarchical cells which are at the level specified by *n* to be flattened. This option implies **-flatten** option. The value of level can be 1, 2, 3, and so forth. Specifying a value of 1 for **-start_level** flattens the cells from the current design. Specifying a value of 2 causes the top-level blocks to be maintained and all cells in each of the top-level blocks be flattened. The current instance cannot be set when this option is used. This option cannot be used with **-small** option.

-all_instances

Enables the command to accept instance cells that are not unique in the object list.

DESCRIPTION

Removes a single level of hierarchy from the current design by exploding the contents of the specified cell or cell instance in the current design.

Any cells that are marked **dont_touch** are left undisturbed. Black boxes are also not collapsed. Power Compiler-inserted clock gates are not collapsed by default. To do that, set the *power_cg_flatten* variable to TRUE before running **ungroup**.

New cell names are created by concatenating *prefix_name* with original cell names. If the resulting name is not unique, the new unique name is similar to the following:

<prefix_name>cell<number>

If the design contains instances (leaf cells) and only part of design hierarchy needs to be ungrouped, use **current_instance** to set the instance before ungrouping the cells within that design instance. Alternatively, you can use the full hierarchical name of the cell instance. When **current_instance** is defined, it is to be unique and when cell instance is used, the parent must be unique. This restriction to prevent design inconsistency.

If you do not specify a prefix, the default is

cell_being_ungrouped/old_cell_name{number}

If there are dangling nets in the design, they are removed by **ungroup**.

If timing derate is defined for the cell being ungrouped, the derates are pushed down to the lower-level cells if they or their library cells do not have the corresponding derates. This may cause timing inconsistency in the timing report before and after the ungrouping.

Use the **-all_instances** option when any cell in the object list is an instance in the lower hierarchy and its parent cell is not unique. All similar instance cells under the same parent design are ungrouped. The current design does not have to be changed in order to ungroup the instance cells in the lower design hierarchy. If none of the cells in the object list is an instance in the lower hierarchy, or if its parent cell is unique, the **-all_instances** option is not required.

Multicorner-Multimode Support

This command is not supported in a multi-scenario design flow.

EXAMPLES

The following example ungroups a specified list of cells:

```
prompt> ungroup {u1 u2 u3}
```

The following example ungroups a specific cell and specifies the prefix to be used:

```
prompt> ungroup {u1} -prefix "U1:"
```

The following example completely removes the hierarchy of two cells:

```
prompt> ungroup {u1 u2} -flatten
```

The following example completely collapses the hierarchy of the current design:

```
prompt> ungroup -all -flatten
```

The following example ungroups all cells in the design except u1:

```
prompt> set_dont_touch {u1}
```

```
prompt> ungroup -all
```

The following example hierarchically removes cell grouping constraints destined to drive layout placement tools, without modifying the design hierarchy:

```
prompt> ungroup -soft -flatten
```

The following example ungroups all cells under the instance A/B/C:

```
prompt> current_instance A/B
```

```
prompt> ungroup C
```

The following example flattens all cells starting at level 2:

```
prompt> ungroup -start_level 2
```

The following example ungroups the cell instance MID1/BOT1/FOO1 and MID1/BOT1/FOO2. MID1/BOT1 is a unique instantiation of design BOT.

```
prompt> ungroup {MID1/BOT1/FOO1 MID1/BOT1/FOO2}
```

SEE ALSO

```
current_design(2)  
current_instance(2)  
group(2)  
remove_attribute(2)  
set_dont_touch(2)  
power_cg_flatten(3)  
ungroup_keep_original_design(3)
```

uniquify

Removes multiply-instantiated hierarchy in the current design by creating a unique design for each cell instance.

SYNTAX

```
int uniquify  
[-force]  
[-base_name base_name]  
[-cell cell_list]  
[-reference design_name]  
[-new_name new_design_name]  
[-dont_skip_empty_designs]
```

Data Types

<i>base_name</i>	string
<i>cell_list</i>	list
<i>design_name</i>	string
<i>new_design_name</i>	string

ARGUMENTS

-force

Indicates that instances are to be renamed even if they are already unique or are assigned the **dont_touch** attribute. The top-level design is not renamed.

-base_name *base_name*

Specifies the base name to be used instead of the original design name for new designs.

-cell *cell_list*

Specifies a list of cells for which unique designs are to be generated. Unique designs are generated even if the referenced design is already unique, or if the **dont_touch** attribute is set on the cell or its reference. The cells specified in *cell_list* must be in the current design. You cannot specify cells at other levels of hierarchy.

-reference *design_name*

Specifies the name of a design that is referenced by cells for which unique designs are to be created. Unique designs are created even if the referenced design is already unique, or if the **dont_touch** attribute is set on the cells or reference. Only cells in the current design that reference the given design are considered.

-new_name *new_design_name*

Used with the **-cell** option to specify the name for a single cell.

-dont_skip_empty_designs

Indicates that unique designs need to be created even for black box designs. Without this option, **uniquify** will not modify black box designs.

DESCRIPTION

The **uniquify** command removes multiply-instantiated hierarchy in the current design by creating a unique design for each cell instance. The command initiates a search for designs referenced in the hierarchy of the current design and generates new, uniquely-named designs for all instances that have the same name. If the **dont_touch** attribute is set on a cell, reference, or design, a new design is not created for that cell or any cell with that reference or design.

The **uniquify** command links the current design before it executes. If the link fails, **uniquify** aborts and issues an error message.

If a new design is generated, it is copied from the original design, and placed in the same file as the current design. The new design is named according to the **uniquify_naming_style** variable. The default value of this variable is %s_%d, where %s is replaced by the original design name unless **-base_name** is specified, and %d is replaced by the smallest integer that forms a name not used by any other design in memory. The cell reference is updated to use the new design name.

After **uniquify** executes, all designs in the hierarchy of the current design have unique names. However, some names might conflict with designs on disk if the current design is a submodule of another design. In this case, the user must resolve these conflicts before integrating the designs. One way to resolve conflicts is to use the **uniquify** command with different values of the **uniquify_naming_style** variable.

When a cell instance is used with **-cell** option, then the complete path to that instance is uniquified. ie. All the parent instances are uniquified if they are not already unique.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates uniquely-named designs for all cells and designs in the hierarchy of the current design that have non-unique names, unless a cell or reference has the **dont_touch** attribute set.

```
prompt> uniquify
```

In the following example, a new design "ADDER1" is created for cell "U4" in the current design.

```
prompt> uniquify -cell U4 -new_name ADDER1
```

In the following example, new designs are created with the base name "X" for all cells starting with "U" in the current design.

```
prompt> uniquify -cell [get_cells U*] -base_name X
```

The following example forces all designs in the hierarchy of the current design to be renamed using the **uniquify_naming_style** variable.

```
prompt> set uniquify_naming_style "joe_%s_%d"  
prompt> uniquify -force
```

The following example uniquifies the instance mid1 and mid1/bot1. Though the argument to uniquify command is mid1/bot1 since the design mid corresponding to mid1 is not unique, mid1 is also uniquified.

```
prompt> uniquify -cell mid1/bot1
```

SEE ALSO

```
current_design(2)  
set_dont_touch(2)  
set_dont_use(2)  
uniquify_naming_style(3)  
uniquify_keep_original_design(3)
```


uniquify_fp_mw_cel

Removes multiple-instantiated hierarchy lib cell in the current design or a specified design by creating a unique hierarchy lib cell for each hierarchy cell instance.

SYNTAX

```
status uniquify_fp_mw_cel  
[-verbose]  
[-store_mim_property cell_instances]  
[design_name]
```

Data Types

<i>cell_instances</i>	collection
<i>design_name</i>	string

ARGUMENTS

-verbose

Indicates to print more information, like the newly created hierarchy lib cell's name.

-store_mim_property *cell_instances*

Indicates to store a "multi instantiated module" (MIM) property on the user specified cell instances during uniquify. Cells with this property will be treated as MIM in Virtual Flat (VF) MIM Design Planning (DP) Flow while actually it has been uniquified.

design_name

Specifies the name of the design in which uniquification is to be performed. If no design is specified, this command tries to find the current design and performs. If specified, this command opens the design first.

DESCRIPTION

uniquify_fp_mw_cel removes multiply-instantiated hierarchy lib cell in the current design or the specified design by creating a unique hierarchy lib cell for each hierarchy cell instance.

If a new hierarchy lib cell is generated, it is copied from the original one, and placed in the same file as the current design. The new hierarchy lib cell is named in a simple %s_%d format, where %s is replaced by the original hierarchy lib cell name and %d is replaced by the smallest integer that forms a name not used by any other hierarchy lib cell in memory. The cell reference is updated to use the new lib cell name.

After **uniquify_fp_mw_cel** executes, all hierarchy lib cell in the hierarchy of the current design have unique names.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates uniquely-named hierarchy lib cell for all hierarchy cell instances in the hierarchy of the newly created design from read_verilog command.

```
prompt> link
Error: Design is not uniquified. (MWDC-101)
Error: Multiply-
instantiated Hierarchical Cell Master 'sub1' (one instance is
'C1'). (MWDC-043)
    Designs should be uniquified before loading them in IC Compiler.
0
prompt> uniquify_fp_mw_cel CORE
1
prompt> link
1
```

In the following example, an existing design is opened for the uniquify_fp_mw_cel command.

```
prompt> open_mw_cel top -lib design
{top}
prompt> uniquify_fp_mw_cel
1
prompt> save_mw_cel
Information: Saved design named top. (UIG-5)
1
```

The following example uniquified the current design and store MIM property for instance A and B.

```
prompt> report_mim
{A B C D E}
prompt> uniquify_fp_mw_cel -store_mim_property {A B}
1
prompt> report_mim
{A B}
```

SEE ALSO

```
open_mw_lib(2)
open_mw_cel(2)
save_mw_cel(2)
current_mw_cel(2)
report_mim(2)
```

unset_hierarchy_color

Removes colors set on leaf cells.

SYNTAX

```
status unset_hierarchy_color  
[collection1]
```

Data Types

collection1 collection

ARGUMENTS

collection1
Specifies a collection of objects. Colors set on all leaf cells descended from a hierarchical cell in the collection will be removed

DESCRIPTION

If *collection1* is specified, this command removes the colors set on all leaf cells descended from a hierarchical cell in the collection. Otherwise the commands will remove colors on all leaf cells in the current design.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

SEE ALSO

set_hierarchy_color(2)

unset_power_guide

Unsets an existing power guide to be just like an exclusive movebound.

SYNTAX

```
unset_power_guide  
[power_guide_list]
```

Data Types

power_guide_list collection

ARGUMENTS

power_guide_list
Unsets the selected power guides.

DESCRIPTION

This command unsets the given power guides. If no specific name is given, all existing power guides will be unset. They will be in design just like an exclusive movebound.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows the flow of the **unset_power_guide** command. In it, if the given power guide AO_WELL exists in the design it will be unset.

```
prompt> unset_power_guide AO_WELL
```

SEE ALSO

```
create_bounds(2)  
create_voltage_area(2)  
report_power_guide(2)  
set_power_guide(2)
```

unsuppress_message

Enables printing of one or more suppressed informational or suppressed warning messages.

SYNTAX

```
string unsuppress_message  
[messages]
```

Data Types

messages list

ARGUMENTS

messages
 A list of messages to enable.

DESCRIPTION

The **unsuppress_message** command provides a mechanism to re-enable the printing of messages which have been suppressed using **suppress_message**. You can suppress only informational and warning messages, so the **unsuppress_message** command is only useful for informational and warning messages. The result of **unsuppress_message** is always the empty string.

You can suppress a given message more than once. So, you must unsuppress a message as many times as it was suppressed in order to enable it. The **print_suppressed_messages** command displays currently suppressed messages.

EXAMPLES

When the argument to the **unalias** command does not match any existing aliases, the CMD-029 warning message displays. This example shows how to re-enable the suppressed CMD-029 message. Assume that there are no aliases beginning with 'q'.

```
prompt> unalias q*  
prompt> unsuppress_message CMD-029  
prompt> unalias q*  
Warning: no aliases matched 'q*' (CMD-029)
```

SEE ALSO

```
print_suppressed_messages(2)  
suppress_message(2)
```

update_bounds

Updates an existing bound by adding or removing objects. The bound should be of type move bound.

SYNTAX

```
int update_bounds  
[-name bound_name]  
[-bound bound_object]  
[-add]  
[-remove]  
cell_list
```

ARGUMENTS

-name bound_name
Specifies name of the bound. The bound should be a move bound.

-bound bound_object
Specifies the bound object. The bound object can be specified using the *get_bounds* command. You cannot specify more than one bound object.

-add
Specifies that the bound will be updated by adding cells.

-remove
Specifies that the bound will be updated by removing cells.

cell_list
Specifies list of cells to be attached to the bound. If a cell is a hierarchical cell, the bound is also applied to all cells in the subdesigns.

DESCRIPTION

The *update_bounds* command enables you to add or remove cells from an existing move bound. This command does not work with group bounds. You can change the floorplan by adding cells within the move bound or readjust the utilization by removing cells from the move bound. If you add new objects to an existing move bound, the placer honors them during coarse placement. When objects are removed from a move bound, the placer assumes that the objects are no longer constrained by placement bounds. When adding objects, an error occurs if the specified objects are already in a move bound. Similarly for removing objects from move bound, an error occurs if the objects do not belong to the same move bound.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates a bound of name "foo" (using the `create_bounds` command), then adds some cells to the bound "foo", and then removes a cell from the same bound object.

```
prompt> create_bounds -name "foo" -coord {100 100 200 200} INSTN_1fp
```

```
prompt> update_bounds -name "foo" -add {INSTN_2 INSTN_3}
```

```
prompt> update_bounds -bound [get_bounds foo] -remove INSTN_3
```

SEE ALSO

`create_bounds(2)`

`remove_bounds(2)`

`report_bounds(2)`

`get_bounds(2)`

update_clock_latency

Updates the latencies of real and virtual clock objects after clock tree synthesis.

SYNTAX

```
status update_clock_latency
```

ARGUMENTS

This command has no arguments.

DESCRIPTION

This command updates the latencies of real and virtual clock objects after clock tree synthesis. If the **set_latency_adjustment_options** command is used before using the **clock_opt** or before clock tree synthesis, then those directives are used during the update. If no directives are given, then only the latencies of real clock objects are updated. The update mechanism uses the insertion delay (median of the clock arrival times at the clock sinks) of the constructed clock tree to execute a **set_clock_latency** command under the hood on the clock object.

Multicorner-Multimode Support

This command uses information from active scenarios only.

EXAMPLES

The following example shows the use of the command.

```
prompt> update_clock_latency
```

SEE ALSO

```
set_latency_adjustment_options(2)
```


update_flip_chip_pin_locations

Updates the flip chip bump I/O pin locations for timing purposes.

SYNTAX

```
status_value update_flip_chip_pin_locations
```

ARGUMENTS

No arguments needed, the command operates on the current design.

DESCRIPTION

This command updates the I/O pin location to follow the location of the corresponding flip chip bump. This is needed when flip chip bump cells have changed locations and their associated I/O pin locations need to follow for proper timing analysis. If bump locations are changed by `place_flip_chip_array` or `place_flip_chip_ring` commands, the I/O pin locations will be automatically adjusted and there is no need to use this command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command updates the I/O pin locations based on the current flip chip bump location:

```
prompt> update_flip_chip_pin_locations  
1
```

SEE ALSO

```
place_flip_chip_array(2)  
place_flip_chip_ring(2)
```

update_lib

Reads in a specified library file and uses it to update an existing technology, synthetic, or symbol library.

SYNTAX

```
int update_lib  
[-overwrite] [-permanent] library_name file_name [-no_warnings]
```

Data Types

<i>library_name</i>	string
<i>file_name</i>	string

ARGUMENTS

-overwrite

Indicates that a group declared in *file_name* is to overwrite any group in *library_name* that has the same name. Only groups added by **update_lib** or **model** can be overwritten. If **-force** is specified, some library-level groups (wire_load, power_supply and operating_conditions) included in the original library can be modified. The other groups included in the original library and groups added with **-permanent** cannot be modified.

-permanent

Indicates that groups declared in *file_name* are to be stored in the library in such a way that they cannot be overwritten.

library_name

Specifies the name of the library to be updated. The library must be resident in memory.

file_name

Specifies the name of the file in which one or more new groups are listed. The syntax of this file is the same as that expected by **read_lib**, except that the file contains only library level groups without the library() {} group definition.

-no_warnings

Indicates that all messages of severity 'warning' are to be suppressed. The default is to issue all warning messages. Warning messages can identify serious library problems; use this flag sparingly.

DESCRIPTION

Reads in a library file and adds the groups declared in that file to the library specified. Groups are added to the library as if entered at the end of the original text of the library.

Only library-level group statements are added to a library. Library level functions or attributes are not accepted (some library-level attributes can be an exception, as

described in the "-force" section).

The library can be a technology, symbol, or synthetic library. Only groups normally found in a library of that type are accepted. For example, you cannot add a symbol group to a technology library. For technology libraries, the delay model and technology associated with the original library must be the same used by new group(s).

If Library Compiler finds any errors while processing a group, it does not add that group to the library. Legal groups are added. The update operation does not fail if there are illegal groups intermixed with legal groups.

If you update a library with a wire_load group describing a wire load model in use by the current design, the wire load model of the current design is reset.

library_name can have a simple or a complex filename. A simple filename has no directory specification, which in UNIX means that it does not contain a "/" (slash). Simple filenames such as **test.lib** or **library.db** must be found in a directory listed in the **search_path**.

A complex filename has a directory specification, which in UNIX means that it contains a slash. Relative or complex filenames such as **./test.lib**, or **~synopsys/dc/test.lib** are not included in the **search_path**.

For details on library file syntax, refer to the *Library Compiler Reference Manual*. If you do not have a Library Compiler License, function statements and state information are ignored when you attempt to update technology library files. You do not need a Library Compiler License to update symbol library files.

Do not use **update_lib** on the synthetic library **standard.sldb**, because its parts are licensed differently from user parts.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

In the following example, the library file **add_ms.lib**, which contains technology information in Synopsys technology library entry format, is added to the library **cmos**:

```
prompt> update_lib cmos add_ms.lib
```

In the following example, the library file **new_wire_load.lib**, which contains a new **wire_load** group for a technology library, is added to the library **cmos**. Any **wire_load(s)** in the library that have identical names are overwritten if they are not permanent groups. In addition, all warning messages are suppressed with the **-no_warnings** flag.

```
prompt> update_lib -overwrite cmos new_wire_load.lib -no_warnings
```

In the following example, the library file **ramgen.lib**, which contains a black-box cell description of a RAM cell, is added to the library **cmos**. The **-permanent** flag indicates that the new cell cannot be overwritten.

```
prompt> update_lib -permanent cmos ramgen.lib
```

SEE ALSO

`read_lib(2)`
`write_lib(2)`

update_lib_model

Update the library into PG library, either take PG info from reference library FRAM view, or take the PG info definition from TCL setting commands. The library voltage, cell PG pin, cell pin related PG pin will be updated.

SYNTAX

Boolean **update_lib_model**
[-reference_mode FRAM | TCL]
[library_name]

ARGUMENTS

-reference_mode FRAM | TCL
Sets the reference mode where power and ground pin information come from. If FRAM mode is set, MW library opened should have reference library set, or mw_reference_library should be set. PG pin reference will be taken from FRAM views of each library cell. If TCL mode is set, the voltage and PG pin reference will be taken from previous TCL commands: **update_lib_voltage_model**, **update_lib_pg_pin_model** and **update_lib_pin_model**.
If the library to be updated is legacy old PG syntax library, the voltage value will be converted from power_supply attachment. And rail_connection will be converted into related PG pins.

library_name
Specify the library to be converted. If not specified, the libraries in link_library or target_library set will be updated by default.

DESCRIPTION

The **update_lib_model** command update the library into Liberty PG pin library. The library can be non-PG library, legacy old PG syntax library, the reference base for PG pin definition and association can be FRAM reference library or TCL setting. TCL setting can be done by **update_lib_voltage_model**, **update_lib_pg_pin_model** and **update_lib_pin_model** command combinations.

EXAMPLES

following example shows the setting of voltage map and update_lib_model with single library
prompt> update_lib_voltage_model standard_lib -voltage {VDD 1.2} -voltage {VSS 0.0}
prompt> update_lib_model -reference_mode TCL standard_lib

following example shows the updating base on reference library, and it applies to all libraries in link_library/target_library set.
prompt> update_lib_model -reference_mode FRAM

SEE ALSO

`update_lib_voltage_model(2)`
`update_lib_pg_pin_model(2)`
`update_lib_pin_model(2)`

update_lib_pg_pin_model

Set pg pin model for the specified library cell.

SYNTAX

```
Boolean update_lib_pg_pin_model  
cell_name  
[-pg_pin_name {pin_name}]  
[-pg_voltage_name {voltage_name}]  
[-pg_pin_type {pin_type}]  
[-pg_pin_direction {pin_direction}]  
[-pg_physical_connection {physical_connection}]  
[-pg_related_bias_pin {related_pin_name}]
```

Data Types

<i>cell_name</i>	string
<i>pin_name</i>	string
<i>voltage_name</i>	string
<i>pin_type</i>	string [primary_power primary_ground
<i>pin_direction</i>	string [input output inout internal]
<i>physical_connection</i>	string [device_layer routing_pin]

ARGUMENTS

cell_name
Specify the library cell name.

-pg_pin_name {*pin_name*}
Specify name of the library cell's pg pin.

-pg_voltage_name {*voltage_name*}
Specify voltage name of the library cell's pg pin. voltage will be defined in library voltage map by update_lib_voltage_model command.

-pg_pin_type {*pin_type*}
Specify type of the library cell's pg pin.

-pg_pin_direction {*pin_direction*}
Specify direction of the library cell's pg pin.

-pg_physical_connection {*physical_connection*}
Specify physical connection of the library cell's pg pin.

-pg_related_bias_pin {*related_pin_name*}
Specify related bias pin of the library cell's pg pin.

DESCRIPTION

The **update_lib_pg_pin_model** command sets the pg pin map for the specified library cell. The map will be used later during **update_lib_model**.

cell_name is in the form of "library_name/cell_name". If library_name is omitted, the pg pin map will apply to all libraries.

If the pg pin map already exists, all the new specifications will override the old ones. If not, the new pg pin will be created for the cell.

If a list of pins is specified, then other options must be paired each other.

EXAMPLES

```
prompt> update_lib_pg_pin_model LIB1/CELL1 -pg_pin_name {VDD VSS} -  
pg_voltage_name {VDDL VSS} -pg_pin_type {primary_power primary_ground}
```

SEE ALSO

```
update_lib_voltage_model(2)  
update_lib_pin_model(2)  
update_lib_model(2)
```


update_lib_pin_model

Set pin model for the specified library cell.

SYNTAX

```
Boolean update_lib_pin_model  
cell_name  
[-related_power_pin {pin_name}]  
[-related_ground_pin {pin_name}]  
[-related_bias_pin {related_bias_pin}]  
[-power_down_function {power_down_func}]  
[-pins {cell pin name list}]
```

Data Types

<i>cell_name</i>	string
<i>pin_name</i>	string
<i>related_bias_pin</i>	string
<i>power_down_func</i>	string

ARGUMENTS

cell_name
Specify the library cell name.

-related_power_pin {*pin_name*}
Specify related power pin of the library cell's pin.

-related_ground_pin {*pin_name*}
Specify related ground pin of the library cell's pin.

-related_bias_pin {*related_bias_pin*}
Specify related bias pin of the library cell's pin.

-power_down_function {*power_down_func*}
Specify power down function of the library cell's pin.

DESCRIPTION

The **update_lib_pin_model** command sets the pin map for the specified library cell. The map will be used later during **update_lib_model**.

cell_name is in the form of "library_name/cell_name". If library_name is omitted, the pin map will apply to all libraries.

If the pin map already exists, all the new specifications will override the old ones. If not, the new pin map will be created for the cell.

If a list of pins is specified, then other options must also be paired each other.

EXAMPLES

```
prompt> update_lib_pin_model LIB1/CELL1 -pin_name {A B} -  
related_power_pin {VDD VDD} -related_group_pin {VSS VSS}
```

SEE ALSO

```
update_lib_voltage_model(2)  
update_lib_pg_pin_model(2)  
update_lib_model(2)
```

update_lib_voltage_model

Set voltage model for the specified library.

SYNTAX

```
Boolean update_lib_voltage_model  
library_name  
[-voltage {voltage_name voltage_value}]
```

Data Types

<i>library_name</i>	string
<i>voltage_name</i>	string
<i>voltage_value</i>	float

ARGUMENTS

library_name
Specify the library name for the voltage definition.

-voltage {*voltage_name* *voltage_value*}
Set the voltage name and value pair for the library.

DESCRIPTION

The **update_lib_voltage_model** command sets the voltage map for the specified library. The map will be used later during **update_lib_model**.

EXAMPLES

following example shows the setting for single rail library

```
prompt> update_lib_voltage_model standard_lib -voltage {VDD 1.2} -voltage {VSS 0.0}
```

following example shows the setting for multiple rail library

```
prompt> update_lib_voltage_model standard_lib -voltage {VDD 1.2} -  
voltage {VSS 0.0} \  
prompt>                               -voltage {VDD_BACK 0.7}
```

SEE ALSO

```
update_lib_model(2)  
update_lib_pg_pin_model(2)  
update_lib_pin_model(2)
```

update_physical_bus

Updates an existing physical bus by adding or removing objects.

SYNTAX

```
status update_physical_bus
physical_bus
-add net_list | -remove net_list
[-position position]
[-sort {ascending | descending | none}]
[-delimiter delimiter]
[-right_precedence]
[-quiet]
```

Data Types

<i>physical_bus</i>	string
<i>net_list</i>	list
<i>position</i>	integer
<i>delimiter</i>	string

ARGUMENTS

physical_bus
Specifies a physical bus object. You can specify name of a physical bus object or a collection containing a physical bus.

-add net_list
Specifies a list of nets to be added into the specified physical bus.
The **-add** and **-remove** options are mutually exclusive. You must specify one, but not both.

-remove net_list
Specifies a list of nets to be removed from the specified physical bus.
The **-add** and **-remove** options are mutually exclusive. You must specify one, but not both.

-position position
Specifies the position after which the specified nets will be added.
This option is effective with the option **-add** specified.
If you don't specify *position*, nets will be appended to the end.

-sort {ascending | descending | none}
Specifies the sort mode. The valid values are: **ascending**, **descending**, and **none**.
This option is effective with the option **-add** specified.
If you specify *-sort ascending*, nets will be sorted by their names alphanumerically, from lowest to highest. If you specify *-sort descending*, nets will be sorted by their names alphanumerically, from highest to lowest. If you specify *-sort none*, no sort will be applied.
By default *sort_mode* is *none*.

-delimiter *delimiter*

Specifies the delimiter characters to use for the names of nets.
This option is effective with the option **-add** specified.
The valid values of *delimiter* are: **[]**, **{}**, **<>**, **()**, **_** and **:**. By default *delimiter* is **[]**.
When nets are sorted, characters inside *delimiter* will be regarded as index.
For example, Two nets Net[2] and Net[10] with *-sort ascending*, Net[2] will be in front of Net[10].

-right_precedence

Indicates that sorting of index inside *delimiter* is from right to left. By default the order is from left to right.
This option is effective with the option **-add** specified.
For example, there are six nets with *-sort ascending* and no option **-right_precedence**, the ordered list will be:
Net[0][1], Net[0][2], Net[1][1], Net[1][2], Net[2][1], Net[2][2],
While with *-sort ascending* and option **-right_precedence**, the ordered list will be:
Net[0][1], Net[1][1], Net[2][1], Net[0][2], Net[1][2], Net[2][2],

-quiet

Suppresses warning and error messages. Syntax error messages are not suppressed.

DESCRIPTION

The **update_physical_bus** command enables you to add or remove nets from an existing physical bus.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates a physical bus of name "bus1", removes two nets from it, adds other two nets to it, and then use command **get_nets -of_object \$bus1** to see the elements in the physical bus.

```
prompt> set bus1 [create_physical_bus bus1 -sort ascending \  
-nets [get_nets UPC_DATA*]] fp  
{bus1}
```

```
prompt> update_physical_bus $bus1 -remove \  
[get_nets {UPC_DATA[2] UPC_DATA[5]}]  
1
```

```
prompt> update_physical_bus bus1 -position 0 -add \  
[get_nets {Y[0] Y[3]}] -sort descending
```

1

```
prompt> get_nets -of_object $bus1
```

```
{Y[3] Y[0] UPC_DATA[0] UPC_DATA[1] UPC_DATA[3] UPC_DATA[4] UPC_DATA[6] UPC_DATA[7]  
UPC_DATA[8] UPC_DATA[9] UPC_DATA[10] UPC_DATA[11]}
```

SEE ALSO

```
create_physical_bus(2)  
get_nets(2)  
get_physical_buses(2)  
remove_physical_bus(2)  
report_physical_bus(2)
```

update_timing

Updates timing information on the current design.

SYNTAX

```
int update_timing
```

ARGUMENTS

None.

DESCRIPTION

Updates timing for the current design. Timing information is obsoleted when environmental information, such as arrival times or loads, is changed. The **update_timing** command is used to update this information after these changes are made.

Timing information is automatically updated in the current design when **compile** or **translate** is used or when reports that display timing information are used. So, for most cases, the **update_timing** command is not needed. However, there are instances when timing might not be current (such as if **set_attribute** is incorrectly used instead of **set_drive** to set a drive value).

Note that this command does not automatically identify and define clocks in the design. These should be defined by using the **create_clock** command.

Multicorner-Multimode Support

This command uses information from active scenarios only.

EXAMPLES

This example updates timing for the current design.

```
prompt> update_timing
```

SEE ALSO

```
create_clock(2)  
set_drive(2)  
current_design(3)
```

update_voltage_area

Updates an existing voltage area by adding or removing logical hierarchies.

SYNTAX

```
int update_voltage_area  
-voltage_area list  
[-add]  
[-remove]  
[-guard_band_x int]  
[-guard_band_y int]  
[modules]
```

ARGUMENTS

-voltage_area *list*
Specifies the voltage areas. The patterns can be a collection handle of voltages or name patterns:
If more than one voltage area object is specified, this command will report error. This option must be specified.

-add
Specifies that the voltage area will be updated by adding hierarchical cells. This option is exclusive with *-remove*. This option must be used with *modules*.

-remove
Specifies that the voltage area will be updated by removing hierarchical cells.
This option is exclusive with *-add*. This option must be used with *modules*.

-guard_band_x *int*
Specifies the horizontal margin or the left margin and the right margin of the voltage area where you want to change the guard band.

-guard_band_y *int*
Specifies the vertical margin or the top margin and the bottom margin of the voltage area where you want to change the guard band.

modules
Specifies list of cells to be attached with the voltage area.
This option must be used with *-add* or *-remove*.

DESCRIPTION

update_voltage_area command allows the user to keep adding or removing logical hierarchies in the existing voltage area.

If you try to add a hierarchical cell on which another voltage area is defined, the command will fail.

If you try to remove a hierarchical cell on which current voltage area isn't

defined, the command will fail.

When a hierarchical cell is removed from specified voltage area, its all leaf child cells will be reassocated with the its nearest ascent's voltage area, if any. If none of its ascents is associated with any voltage area, or it doesn't have any ascents, its all leaf child cells will be free.

You can update the value of the guard band around the voltage area using the **-guard_band_x** option and the **-guard_band_y** option. The tool considers the guard bands as placement keepouts. Use guard bands as a method to prevent substrate leakage between voltage areas.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example updates the voltage area named "foo" by first adding cells and then removing a cell from the same voltage area.

```
prompt> create_voltage_area -coord {10 10 100 80} -name foo \  
[get_cells INSTN_1]  
{foo}  
prompt> update_voltage_area -voltage_area foo -add INSTN_2  
1  
prompt> update_voltage_area -voltage_area [get_voltage_area foo] \  
-remove INSTN_2  
1
```

The following example shows error case by adding a hierarchical cell on which another voltage area is defined, and by removing a hierarchical cell on which the current voltage area isn't defined.

```
prompt> create_voltage_area -coord {10 10 50 50} -name VA1 \  
[get_cells INSTN_1]  
{VA1}  
prompt> create_voltage_area -coord {100 100 180 160} -name VA2 \  
[get_cells INSTN_2]  
{VA2}  
prompt> update_voltage_area -voltage_area VA1 -add INSTN_2  
ERROR : hierarchical cell 'INSTN_2' already has voltage area.  
prompt> update_voltage_area -voltage_area VA1 -remove INSTN_1/A1  
ERROR : cell 'INSTN_1/A1' doesn't belong to the current voltage area.
```

SEE ALSO

```
create_voltage_area(2)  
remove_voltage_area(2)  
report_voltage_area(2)  
get_voltage_areas(2)
```

verify_drc

Detects DRC violations.

SYNTAX

```
integer verify_drc
[-error_cell cell_name]
[-dir runset_directory]
[-ignore_width]
[-ignore_spacing]
[-ignore_area]
[-ignore_enclosed_area]
[-ignore_density]
[-ignore_min_edge_length]
[-check_via_size]
[-ignore_via_spacing]
[-ignore_adjacent_via]
[-ignore_min_via_number]
[-ignore_stack_level]
[-ignore_stackable]
[-check_enclosure]
[-check_end_of_line]
[-check_via_farm]
[-check_fat_poly_contact]
[-check_blockage]
[-ignore_child_cell]
[-read_cell_view]
[-check_cross_hier_short_only]
[-write_hercules_runset_only]
[-selected_area {{llx1 lly1} {urx1 ury1} ...}]
[-excluded_area {{llx1 lly1} {urx1 ury1} ...}]
[-exclude_by_cell_name {collection_of_master_cells}]
[-offset_distance value]
```

Data Types

<i>cell_name</i>	string
<i>runset_directory</i>	string
<i>llx1</i>	float
<i>lly1</i>	float
<i>urx1</i>	float
<i>ury1</i>	float
<i>collection_of_master_cells</i>	collection
<i>value</i>	float

ARGUMENTS

-error_cell *cell_name*
Name to assign to the error cell. If the option is missing,
<current_cell_name>_adrc.err is used by default.

`-dir runset_directory`
 Name of the Hercules runset directory. Default: adrc.

`-ignore_width`
 Set this option to skip checking the minimum and maximum width rules. These rules are checked by default.

`-ignore_spacing`
 Set this option to skip checking all rules related to spacing, including basic spacing, same-net spacing (notch), basic fat spacing, fat table spacing, the fat table parallel length rule (mode 0/1), and the fat table extension range rule (mode 0/1). These rules are checked by default.

`-ignore_area`
 Set this option to skip checking the minimum area rule. This rule is checked by default.

`-ignore_enclosed_area`
 Set this option to skip checking the basic minimum enclosed area rule and the fat table minimum enclosed area rule. These rules are checked by default.

`-ignore_density`
 Set this option to skip checking the metal density rule. This rule is checked by default.

`-ignore_min_edge_length`
 Set this option to skip checking the minimum-number-of-consecutive-short-edge rule. This rule is checked by default.

`-check_via_size`
 Set this option to check the via size rule. Usually, via sizes on the same layer are uniform, so it is not necessary to activate this option in most cases. These rules are not checked by default.

`-ignore_via_spacing`
 Set this option to skip checking the via spacing rules, including the basic via spacing rule, the basic fat contact rule, the fat contact table spacing rule, and the fat contact table extension range rule. These rules are checked by default.

`-ignore_adjacent_via`
 Set this option to skip checking the maximum-number-of-adjacent-via rule and the enclosed via rule. These rules are checked by default.

`-ignore_min_via_number`
 Set this option to skip checking the minimum-number-of-via rule defined in the fat contact table (including the extension range). These rules are checked by default.

`-ignore_stack_level`
 Set this option to skip checking the maximum-number-of-stack-layer rule. This rule is checked by default.

`-ignore_stackable`
 Set this option to skip checking the stack via rule. This rule is checked by

default.

-check_enclosure

Set this option to check the via enclosure rule. Usually, this rule is guaranteed by the definition of vias, so it is not necessary to activate this option in most cases. These rules are not checked by default.

-check_end_of_line

Set this option to check the end-of-line rule. Usually, this rule is guaranteed by the via enclosure rule, so it is not necessary to activate this option in most cases. These rules are not checked by default.

-check_via_farm

Set this option to check the via farm (via array) rule. This rule specifies the via array size and the spacing between two via arrays. These rules are not checked by default.

-check_fat_poly_contact

Set this option to check the fat poly contact rule. This rule specifies the thresholds that the router uses to determine whether a metal wire is fat. The tool avoids creating new fat wires of width greater than the threshold when connecting to a pin. These rules are not checked by default.

-check_blockage

Set this option to check the blockages on the metal layers. When this option is off, all blockages are ignored. When it is on, the blockages on the metal layers are treated as regular metal wires, and the blockages on the metal_blockage layer are treated as thin blockages. The option is off (blockages are ignored) by default.

-ignore_child_cell

Set this option to skip the violations inside or among all child cells. This option is off by default.

-read_cell_view

Set this option to read child cells from the cell view. If the cell view for a child cannot be found, the command stops. When this option is off (default), the child cells are read from the FRAM view. If the FRAM view for a child cell cannot be found, the command ignores it and continues.

-check_cross_hier_short_only

Set this option to check only overlapping errors between top level wires and metal objects, like wires, vias, blockages, routeguides and pins inside a FRAM view, including both Macros and Standard Cells. This option is useful for a quick violation check after manual editing. This option is off by default.

-write_hercules_runset_only

Set this option to dump the runset to the directory, specified in "dir" without running Hercules. You can modify and run the runset ev_engine adrc.ev This option is off by default.

-selected_area {{llx1 lly1} {urx1 ury1} ...}

Specify either an area or areas to perform design rule checking. An area is a collection of two points that specify the lower-left and upper-right

corners.

`-excluded_area {{llx1 lly1} {urx1 ury1} ...}`

Specify either an area or areas to exclude the design rule checking.

`-exclude_by_cell_name {collection_of_master_cells}`

Specify the cell master name(s) to exclude the design rule checking.

`-offset_distance value`

Specify a value to extend or shrink the checking areas (or cell boundary), the value should be more than 0. The value can be provided by user or automatically calculated (if no value is specified). For the automatic calculated case, it must get the largest possible checking value defined in the technology file. For the selected_area case, it will extend a value outside the checking area or cell boundary. On the other hand, for excluded_area or exclude_by_cell_name case, it will shrink a value inside the checking area or cell boundary.

DESCRIPTION

The **verify_drc** command detects design rule violations. It automatically generates design rule checking (DRC) runsets for the Hercules tool, then calls Hercules to check the violations. A Hercules license is required to run the **verify_drc** command. For a more detailed description about Hercules, please refer to the *Hercules Reference Manual*.

For information about the design rules and the technology file attributes that define them, see the *Milkyway Data Preparation User Guide*.

All options are optional.

NOTES

For the options `-exclude_by_cell_name` and `-offset_distance`, any data from cells at higher levels in the hierarchy that are within the selected boundary of the cell will be excluded as well.

The **verify_drc** command does not support macro metal density from LEF or PLIB when the option `-read_cell_view` is used.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example is to run command **verify_drc** with customized error cell name.

```
prompt> verify_drc -error_cell my_err_cell
```

The following example is to run command **verify_drc** with two selected areas (50.17 103.85)(105.29 147.16), (144.3 49.09)(184.02 97.05), with the offset distance 0.6.

```
prompt> verify_drc -selected_area {{50.17 103.85} {105.29 147.16} \  
{144.3 49.09} {184.02 97.05}} -offset_distance 0.6
```

The following example is to run command **verify_drc** with a excluded area (50.17 103.85) (105.29 147.16), and the offset distance is automatically calculated.

```
prompt> verify_drc -excluded_area {{50.17 103.85} {105.29 147.16}}
```

The following example is to run command **verify_drc** with the excluding master cells *A* and *B* that do not need to be verified, with the offset distance 0.

```
prompt> verify_drc -exclude_by_cell_name {A B} -offset_distance 0
```

SEE ALSO

```
verify_lvs(2)  
verify_route(2)  
count_drc_violations(2)
```

verify_lvs

Checks for inconsistencies between the schematic and physical layout of the current design. Violations discovered by the check are written to a design error view (error cell). You can browse the error view with the IC Compiler Error Browser.

SYNTAX

```
status verify_lvs
[-error_cell cell_name]
[-ignore_floating_port]
[-ignore_floating_net]
[-ignore_short]
[-ignore_open]
[-ignore_eeq_pin]
[-ignore_min_area]
[-use_notch_gap_fill_cell]
[-ignore_blockage_overlap]
[-check_single_pin_net_for_floating_port]
[-check_floating_port_on_null_net]
[-ignore_floating_metal_fill_net]
[-max_error ]
[-check_single_pin_net_for_floating_net]
[-check_short_locator]
[-check_open_locator]
```

ARGUMENTS

-error_cell *cell_name*
Specifies the name to assign to the design error view.
By default, the name is *current_cell_name_adrc.err*.

-ignore_floating_port
Do not check for ports that are not connected to a net.

-ignore_floating_net
Do not check for nets that are not connected to anything.

-ignore_short
Do not check for shorts between nets.

-ignore_open
Do not check for open nets.

-ignore_eeq_pin
Do not check for pins of different electrical equivalent classes that are connected together.

-ignore_min_area
Do not check for metal that is less than the minimum area.

-use_notch_gap_fill_cell
Include notch gap filler cells in the checking. By default, these cells are

not checked.

`-ignore_blockage_overlap`

Do not check for cell overlaps.

`-check_single_pin_net_for_floating_port`

Check for pins that are not connected to top-level metal. By default, this rule is not checked.

`-ignore_floating_metal_fill_net`

Do not check the floating metal created by metal fill.

`-max_error`

Specifies the maximum number of errors reported in this LVS run.
By default, there is no limit to the number of reported errors.

`-check_single_pin_net_for_floating_net`

Check for top-level metal that connect to only one pin. By default, this rule is not checked.

`-check_short_locator`

Report the detailed short locations.

By default, **verify_lvs** reports only which nets are shorted with each others and the reported short locations are the shapes for the nets containing the short.

When you specify this option, **verify_lvs** reports the exact short locations. A short type error is written for each short in a net and the reported short locations are the bounding boxes around the shorted areas. There is an additional runtime cost for computing the detailed short locations.

`-check_open_locator`

Report the detailed open pair locations.

By default, **verify_lvs** reports the nets with any number of opens, but not the potential connection endpoints. The open nets are shown as open type errors. When you specify this option, **verify_lvs** reports each open in a net as a pair of potential connection endpoints. The opens are shown as open locator type errors. There is an additional runtime cost for computing the detailed open pair locations.

DESCRIPTION

The **verify_lvs** command detects schematic versus physical layout errors.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example runs the **verify_lvs** command and generates an error cell named `my_err_cell`.


```
prompt> verify_lvs -error_cell my_err_cell
```

SEE ALSO

verify_pg_nets

Checks whether or not all power and ground pins of standard cells, macro cells and pad cells are connected to the corresponding power and ground nets.

SYNTAX

```
integer verify_pg_nets  
[-error_cell cell_name]  
[-std_cell_pin_connection ( check | ignore)]  
[-macro_pin_connection ( at_least_one | all | ignore)]  
[-pad_pin_connection ( at_least_one | all | ignore)]
```

ARGUMENTS

-error_cell *cell_name*
Specifies the name of the error cell. If this option is skipped, `<top_cell_name>.err` is used as default.

-std_cell_pin_connection (*check* | *ignore*)
Specifies whether to *check* (which is a default) or *ignore* checking if each standard cell connects to power and ground.

-macro_pin_connection (*at_least_one* | *all* | *ignore*)
Specifies to check whether and how macro cells connect to power and ground. *at_least_one* (which is a default) checks whether each macro cell has at least one pin connected to power and ground. *all* checks whether each macro cell has all its power and ground pins connected to power and ground. *ignore* does not check whether macro cells connect to power and ground.

-pad_pin_connection (*at_least_one* | *all* | *ignore*)
Specifies to check whether and how pads connect to power and ground. *at_least_one* (which is a default) checks whether each pad has at least one pin connected to power and ground. *all* checks whether each pad has all its power and ground pins connected to power and ground. *ignore* does not check whether pads connect to power and ground, and does a check for floating shapes on all power and ground nets, and flags the violations you have reported.

DESCRIPTION

The **verify_pg_nets** command checks whether or not all power and ground pins of standard cells, macro cells and pad cells are connected to the corresponding power and ground nets. The `verify_pg_nets` command also checks each power and ground net for floating shapes, and creates an error cell that you can view the errors. Checks power and ground pins only. Tie-up and tie-down pins, which are signal pins, are not checked. Use the `connect_pg_nets` command to specify port-to-net connections for power and ground nets if they are not part of the netlist.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

To use default <top_cell_name>.err cell, not check standard cell connections, check all macro pin connections, and check at least one pin of pad cells connection to power and ground (used as default here), enter

```
verify_pg_nets -std_cell_pin_connection ignore -macro_pin_connection all
```

SEE ALSO

```
create_rectangular_rings(2)  
create_power_straps(2)  
create_preroute_vias(2)  
preroute_instances(2)  
preroute_standard_cells(2)
```

verify_route

Verifies and reports DRC violations and opens

SYNTAX

```
status verify_route
[-nets {collection_of_nets}]
[-no_drc]
[-no_opens]
[-antenna]
[-top_layer_probe_constraints]
[-num_cpu int]
[-bounding_box {{llx lly} {urx ury}}]
[-output file_name]
```

Data Types

<i>collection_of_nets</i>	collection
<i>int</i>	integer
<i>llx</i>	float
<i>lly</i>	float
<i>urx</i>	float
<i>ury</i>	float
<i>file_name</i>	string

ARGUMENTS

-nets {*collection_of_nets*}
Specifies the nets to report the opens only violations. If you specify more than one net in a collection they must be separated by a space. This option is off by default.

-no_drc
Ignored drc violations in detail routed design. This option is off by default.

-no_opens
Ignored opens in detail routed design. This option is off by default.

-antenna
Check and report for Charge-Collecting Antenna in a routed design. This option is off by default.

-top_layer_probe_constraints
Check and report for Top-Layer Probe Constraints in a routed design. This option is off by default.

-num_cpu *int*
Selected this option to use the distributed routing feature, that is, to run the command parallel on different CPUs and machines. Enter the number of CPUs that the command uses. If it is less than 1, this distributed routing feature is disabled and uses the single CPU, which is default value. If it is ≥ 1 , the network has to be setup properly to be able to run the command parallel

before selecting this option. The upper limit is 63.

`-bounding_box {{llx lly} {urx ury}}`

Specify an area to perform design rule checking. An area is a collection of two points that specify the lower-left and upper-right corners. This option is off by default.

`-output file_name`

Output drc violations into a file. This option is off by default.

DESCRIPTION

Verifies, checks and reports the following about a routed design: DRC Violations, Opens, Charge-Collecting Antenna violations, and Top-Layer Probe Constraints.

If `-nets` option is selected, **verify_route** checks only Opens of the specified nets, and all other arguments are ignored.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports drc violations and opens by running the command on one CPU.

```
prompt> verify_route
```

The following example ignored drc violations in report. The job is distributed on two CPUs and run in parallel.

```
prompt> verify_route -no_drc -num_cpu 2
```

The following example reports opens for nets N1, N2, and N3.

```
prompt> verify_route -nets {N1 N2 N3}
```

The following example checks DRC violations within the bounding box (40.27 98.85) (135.39 167.16).

```
prompt> verify_route -bounding_box {{40.27 98.85} {135.39 167.16}}
```

The following example dumps drc violations to the file drc.sum.

```
prompt> verify_route -output drc.sum
```

SEE ALSO

`count_drc_violations(2)`

verify_zrt_route

Verifies and reports design rule constraint (DRC) violations, net opens, antenna rule violations, and voltage area rule violations.

SYNTAX

```
status verify_zrt_route  
[-nets {collection_of_nets}]  
[-open_net true | false]  
[-report_all_open_nets true | false]  
[-drc true | false]  
[-antenna true | false]  
[-voltage_area true | false]  
[-check_from_user_shapes true | false]
```

Data Types

collection_of_nets collection or list

ARGUMENTS

-nets {collection_of_nets}
Specifies the nets to verify. If you specify this option, the command does not check for DRC violations, regardless of the setting of the **-drc** option. By default, all nets are verified.

-open_net true | false
If true (the default), the command checks for open nets in the design.

-report_all_open_nets true | false
If false (the default), the command reports a maximum of 200 open nets when checking for open nets.

-drc true | false
If true (the default), the command checks for design rule violations on the specified nets.

-antenna true | false
If true (the default), the command checks for charge-collecting antenna rule violations on the specified nets.

-voltage_area true | false
If true (the default), the command checks for voltage-area rule violations on the specified nets.

-check_from_user_shapes true | false
If true, the command checks DRC violations for user-created shapes, as well as routing shapes. The default is false.

DESCRIPTION

Verifies, checks, and reports the following rules for a routed design: DRC violations, open nets, charge-collecting antenna violations, and voltage-area violations.

If you specify the **-nets** option, **verify_zrt_route** checks only for open nets, antenna violations, and voltage-area violations of the specified nets. All other arguments are ignored.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports DRC violations, open nets, antenna violations, and voltage-area violations for all nets in the design.

```
prompt> verify_zrt_route
```

The following example checks for open nets, antenna violations, and voltage-area violations, but not for DRC violations.

```
prompt> verify_zrt_route -drc false
```

The following example checks for open nets, antenna violations, and voltage-area violations for nets N1, N2, and N3. Note that these rules are checked by default, and DRC violations are not checked.

```
prompt> verify_zrt_route -nets {N1 N2 N3}
```

The following example reports only antenna and voltage-area violations for nets N1 and N2.

```
prompt> verify_zrt_route -nets {N1 N2} -open_net false
```

SEE ALSO

`route_zrt_detail(2)`

which

Locates a file and displays its pathname.

SYNTAX

```
string which  
filename_list
```

Data Types

```
filename_list      list
```

ARGUMENTS

```
filename_list  
    List of files to locate.
```

DESCRIPTION

Displays the location of the specified files. This command uses the `search_path` to find the location of the files. This command can be a useful prelude to `read_db` or `link_design`, because it shows how these commands expand filenames. The **which** command can be used to verify that a file exists in the system.

If an absolute pathname is given, the command searches for the file in the given path and returns the full pathname of the file.

EXAMPLES

The following examples are based on the following `search_path`.

```
prompt> set search_path "/u/foo /u/foo/test"
```

The following command searches for the file name `foo1` in the `search_path`.

```
prompt> which foo1  
/u/foo/foo1
```

The following command searches for files `foo2`, `foo3`.

```
prompt> which {foo2 foo3}  
/u/foo/test/foo2 /u/foo/test/foo3
```

The following command returns the full pathname.

```
prompt> which ~/test/designs/sub_design.db  
/u/foo/test/designs/sub_design.db
```

SEE ALSO

`search_path(3)`

widen_zrt_wires

Performs wire widening on the current design to prevent open nets.

SYNTAX

```
status widen_zrt_wires  
[-timing_preserve_nets {collection_of_nets}]  
[-timing_preserve_setup_slack_threshold slack_value]  
[-timing_preserve_hold_slack_threshold slack_value]
```

Data Types

<i>collection_of_nets</i>	collection
<i>slack_value</i>	float

ARGUMENTS

`-timing_preserve_nets {collection_of_nets}`
Specifies the nets to preserve timing on.
If you specify this option, the command considers the specified nets as timing critical and skips wire widening on these nets and the neighboring nets.

`-timing_preserve_setup_slack_threshold slack_value`
Specifies the setup slack threshold for timing critical nets.
If you specify this option, the command skips wire widening on all nets with a worst setup slack less than or equal to this value and the neighboring nets.

`-timing_preserve_hold_slack_threshold slack_value`
Specifies the hold slack threshold for timing critical nets.
If you specify this option, the command skips wire widening on all nets with a worst hold slack less than or equal to this value and the neighboring nets.

DESCRIPTION

The **widen_zrt_wires** command performs wire widening to prevent open nets. It attempts to widen every wire in the design. The options control the maximum number of search and repair iterations performed after wire widening to resolve any violations.

Multicorner-Multimode Support

This command uses information from active scenarios only.

EXAMPLES

The following command performs wire widening.

```
prompt> widen_zrt_wires
```

SEE ALSO

`route_zrt_detail(2)`
`spread_zrt_wires(2)`

window_stretch

Stretches one or more objects by moving the objects points or edges that are containing within a bounding box

SYNTAX

```
status window_stretch  
-delta vector  
-rectangle rect  
[-keep_placement]  
[-keep_pad_to_core_distance]  
[-ignore_fixed]  
objects
```

Data Types

```
rect           x1 y1 x2 y2  
objects       collection
```

ARGUMENTS

- delta *vector*
Specifies the distance to move the objects points or edges which are inside the specified bounding box.
- rectangle *rect*
Specifies the bounding box which defines which points or edges of the specified objects are moved.
- keep_placement
This option is effective only when applied to the core or the die. It will be ignored if applied on any other object.
Cells in the core and IO pad cells are re-placed if necessary. The relative positions of the cells are not guaranteed to be preserved, although an attempt will be made to preserve them as much as possible. Furthermore, the output of the re-placement is not legalized. The user needs to explicitly legalize the placement using `legalize_placement`.
The default is **false**.
- keep_pad_to_core_distance
This option is effective only when applied to the core or the die. It will be ignored if applied on any other object.
This maintains the distance between the core and the pad cells and by implication the distance between the core and the die, thus forcing changes to the core and the die to be made in tandem with each other. If pad cells are absent then the core-to-die distance will still be maintained.
The default is **false**.
- ignore_fixed
Normally fixed objects will not be stretched. If this flag is supplied then fixed objects will also be stretched.

objects

Specifies the objects to be stretched.

DESCRIPTION

This command stretches one or more objects by a specified distance in the x and/or y directions.

For wires and paths, stretching is achieved by moving all points inside the specified bounding box by the specified delta and retaining the position of all points outside the bounding box.

For all other objects, stretching is achieved by moving all edges inside the specified bounding box by the specified delta and retaining the position of all edges outside the bounding box.

See `get_edit_property(2)` man page for more details on which objects can be stretched using the `resizable` property

NOTES

Snapping is done automatically using global snap settings.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example stretches the currently selected objects which are inside the bounding box {100 100 200 200} by 100 units to the right and 200 units above the original object.

```
prompt> set bbox {100 100 200 200}
prompt> set delta {100 200}
prompt> window_stretch -rectangle $bbox -delta $delta [get_selection]
```

SEE ALSO

`resize_objects(2)`
`set_object_snap_type(2)`
`get_object_snap_type(2)`

write

Writes a design netlist or schematic from memory to a file.

SYNTAX

```
status write
[-format output_format]
[-hierarchy]
[-no_implicit]
[-modified]
[-output output_file_name]
[-names_file name_mapping_files]
[-donot_expand_dw]
[-scenarios scenario_list]
[design_list]
[-library library_name]
```

Data Types

<i>output_format</i>	string
<i>output_file_name</i>	string
<i>name_mapping_files</i>	list
<i>scenario_list</i>	list
<i>design_list</i>	list
<i>library_name</i>	string

ARGUMENTS

-format *output_format*
Specifies the output format of the design. Supported output formats and their descriptions are the following:

ddc - Synopsys internal database format (the default format)
vhdl - IEEE Standard VHDL

-hierarchy
Writes all designs in the hierarchy. You can use this option with any other option, If a design hierarchy does not completely link, unresolved design references do not write.

-no_implicit
Indicates not to write (save) the synthetic design hierarchy that is implicitly created during optimization of the design. By default, the command writes designs created in the hierarchy through the use of synthetic libraries, even if you do not use the **-hierarchy** option.

-modified
Writes only the designs that have changed since the last write.

-output *output_file_name*
Specifies a single file into which designs are to be written. By default, the command writes each design into a separate file named *design.suffix*, where

design is the name of each design (a full UNIX path) and *suffix* is the default suffix for the specified format. The default format suffixes and their descriptions are:

.ddc - ddc
.vhd - vhd1

-names_file *name_mapping_files*
Lists files for name changes.

-donot_expand_dw
Indicates that DesignWare components are not to be linked. This option is effective only if you have previously run the **insert_dft** command. If you used **insert_dft** to synthesize boundary-scan using DesignWare components, you can use the **-donot_expand_dw** option to avoid auto linking of the DesignWare components, which would otherwise appear in instance statements as modules having no implementation.

-scenarios *scenario_list*
Lists scenarios whose constraints should be included in the output file. This option applies to ddc format only. By default, the command includes all scenarios.

design_list
Lists designs or design files to write. After a design is written, it still exists in memory. To remove a design from memory, use the **remove_design** command. By default, the command lists the current design.

-library *library_name*
Writes the Synopsys database format object to the specified library.

DESCRIPTION

This command outputs designs from memory to disk. If a design is modified in the tool, you must use the **write** command to save the design.

Multicorner-Multimode Support

By default, this command uses information from the current scenario. You can select different scenarios by using the **-scenarios** option.

EXAMPLES

The following are examples of saving designs in .ddc format.

The following example shows the most common use of saving designs in .ddc format, which is to write a design to its assigned file.

```
prompt> write {A B C}
Writing to file 'A.ddc'
Writing to file 'B.ddc'
```


Writing to file 'C.ddc'

The following example shows that all designs saved in .ddc format are written if you specify a design that is one of many designs in a file.

```
prompt> write top
Writing to file 'top.ddc'
```

The following example shows multiple designs being saved in .ddc format are being written to a single file.

```
prompt> write -output mine.ddc {ADDER MULT16}
Writing to file 'mine.ddc'
```

The following example shows the use of the **-hierarchy** option. All designs (being saved in .ddc format) in the hierarchy of the specified designs are written.

```
prompt> write -hierarchy top
Writing to file 'top.ddc'
Writing to file 'A.ddc'
Writing to file 'B.ddc'
```

The following example shows another use of the **-hierarchy** option. All designs (being saved in .ddc format) in the hierarchy of the specified designs are written.

```
prompt> write -hierarchy -output ~bill/dc/top.ddc top
Writing to '~bill/dc/top.ddc'
```

The following example shows the specifying of the default design filename instead of the design name. It writes all designs that reside in a file named top.ddc.

```
prompt> write top.ddc
Writing to file '/osi4/bill/dc/top.ddc'
```

The following example shows how to resolve an ambiguous file name. You can read more than one design with the same name into memory. To avoid ambiguity, specify the file name when writing the design. You might also have more than one file with the same name in memory (for example, top.ddc and top.ddc). To resolve this conflict, prepend the file name with its full pathname. In the following example, the file name "top.ddc" is ambiguous, and the ambiguity is resolved by using the full pathname.

```
prompt> write top.ddc
Error: 'top' doesn't specify a unique design
Please use complete specification: full_file_name:design_name
prompt> write /osi4/bill/dc/top.ddc
Writing to file '/osi4/bill/dc/top.ddc'
```

SEE ALSO

`change_names(2)`
`define_name_rules(2)`
`read_file(2)`
`write_verilog(2)`
`view_write_file_suffix(3)`

write_def

Writes the design data of the specified design to a file in DEF format, including the physical layout, netlist, and design constraints.

SYNTAX

```
status write_def
-output output_file_name
[-version def_version]
[-unit conversion_factor]
[-compressed]
[-rows_tracks_gcells]
[-vias]
[-all_vias]
[-nondefault_rule]
[-lef lef_file_name]
[-regions_groups]
[-components]
[-macro]
[-fixed]
[-placed]
[-pins]
[-blockages]
[-specialnets]
[-notch_gap]
[-pg_metal_fill]
[-nets]
[-routed_nets]
[-diode_pins]
[-floating_metal_fill]
[-scanchain]
[-no_legalize]
[-verbose]
```

Data Types

<i>output_file_name</i>	string
<i>def_version</i>	real
<i>conversion_factor</i>	integer
<i>lef_file_name</i>	string

ARGUMENTS

-output *output_file_name*
Specifies the name of the output DEF file written by **write_def**. This is a required argument.

-version *def_version*
Specifies the version of DEF to be written to output DEF file. Valid *def_version* values are 5.3, 5.4, 5.5 and 5.6. The default is 5.5.

`-unit conversion_factor`

Specifies the value used in the DEF UNITS DISTANCE MICRONS statement. Valid values for conversion factor are 100, 200, 1000, 2000, 10000 and 20000. The values 10000 and 20000 are supported in DEF 5.6.

`-compressed`

Specifies that the DEF file is compressed using gzip format.

`-rows_tracks_gcells`

Specifies that the ROW, TRACK and GCELLGRID sections are to be included in the output DEF file. It can be combined with options for writing out other DEF sections.

`-vias`

Specifies that the VIAS section is to be included in the output DEF file. It can be combined with options for writing out other DEF sections.

`-all_vias`

Specifies that all vias (those present in the design, as well as those present in technology file) are included in the output DEF file. If this option is not specified, only vias present within the design are written out.

`-nondefault_rule`

Specifies that the NONDEFAULTRULES section is to be included in the output DEF file. The option only works for DEF output of version 5.6 because the NONDEFAULTRULES syntax is not supported in DEF 5.5 or lower DEF version.

`-lef lef_file_name`

Specifies that rotated vias and design-specified nondefault rule are to be written to the specified LEF file.

`-regions_groups`

Specifies that the REGIONS and GROUPS sections are to be included in the output DEF file. It can be combined with options for writing out other DEF sections.

`-components`

Specifies that the COMPONENTS section is to be included in the output DEF file. It can be combined with options for writing out other DEF sections.

`-macro`

Specifies that only macro cells are to be included in the COMPONENTS section. If this option is not specified, standard and macro cells are included in the COMPONENTS section.

`-fixed`

Specifies that only fixed cells are to be included in the COMPONENTS section. If this option is not specified, fixed, cover, placed, and unplaced cells are included in the COMPONENTS section.

`-placed`

Specifies that only fixed and placed cells are to be included in the COMPONENTS section. If this option is not specified, fixed, cover, placed, and unplaced cells are included in the COMPONENTS section.

-pins
Specifies that the PINS section is to be included in the output DEF file. It can be combined with options for writing out other DEF sections.

-blockages
Specifies that the BLOCKAGES section is to be included in the output DEF file. It can be combined with options for writing out other DEF sections.

-specialnets
Specifies that the SPECIALNETS section is to be included in the output DEF file. To include notch/gap and P/G metal fills as well, use "-notch_gap" and "-pg_metal_fill". It can be combined with options for writing out other DEF sections.

-notch_gap
Specifies that the notch/gap is to be included in SPECIALNETS section. By default, -notch_gap option is off.

-pg_metal_fill
Specifies that the P/G metal fills are to be included in SPECIALNETS section. By default, -pg_metal_fill option is off.

-nets
Specifies that the NETS section is to be included in the output DEF file.

-routed_nets
Specifies that only routed nets are to be included in the SPECIALNETS and NETS sections. If this option is not specified, then all nets are included in the SPECIALNETS and NETS sections.

-diode_pins
Specifies that the diode (extra) pins are to be included in the NETS section. By default, -diode_pins option is on.

-floating_metal_fill
Specifies that the floating metal fill is to be included in the FILLS section. By default, -floating_metal_fill option is off.

-scanchain
Specifies that the SCANCHAINS section is to be included in the output DEF file. It can be combined with options for writing out other DEF sections.

-no_legalize
Specifies that name legalization should not occur. Name legalization escapes DEF special characters that do not actually have special meaning. Escaping the special character causes its special meaning to be ignored by the DEF reader. For example, if a component name contains bus bit characters, the bus bit characters will be escaped because it is not really a bus. By default, -no_legalize option is off.

-verbose
Enables the printing of additional debugging messages, of two types: those that contain information and those that contain warnings.

DESCRIPTION

The **write_def** command writes the design data of the specified design to a file in DEF format, including physical layout, netlist and design constraints.

NOTE: write_def uses the current_design from the current, open Milkyway design library. User need to open Milkyway design before write_def.

If none of **-rows_tracks_gcells**, **-vias**, **-regions_groups**, **-components**, **-macro**, **-fixed**, **-placed**, **-pins**, **-blockages**, **-specialnets**, **-nondefault_rule**, **-nets**, **-routed_nets**, **-diode_pins**, **-scanchain**, **-notch_gap**, **-floating_metal_fill**, and **-pg_metal_fill** are specified, all DEF sections are completely written out. Otherwise only the specified DEF sections are written out.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example writes out a file *out.def* from the existing design *top*.

```
prompt> write_def -output out.def
```

SEE ALSO

```
read_def(2)  
save_mw_cel(2)
```

write_design_lib_paths

Writes into a file the paths to which design libraries are mapped.

SYNTAX

```
status write_design_lib_paths  
[-filename file_name]  
[-dc_setup]
```

Data Types

file_name string

ARGUMENTS

-filename *file_name*

Specifies the file to which design library information is to be written. If **-dc_setup** is specified, the default file to which the data is written is **.synopsys_dc.setup** in the current working directory. If **-dc_setup** is not specified, the default file to which the data is written is determined by setting the **design_library_file** variable.

-dc_setup

Indicates that the *file_name* is to be written in **.synopsys_dc.setup** format. By default, the file is written in **synopsys_vss.setup** format (the format used by the Synopsys simulator). If **-dc_setup** is specified, the file written contains **define_design_lib** commands. The **-dc_setup** option changes the default *file_name* value to **.synopsys_dc.setup** in the current working directory.

DESCRIPTION

This command writes into a file the paths to which design libraries are mapped. The format is either **.synopsys_vss.setup** or **.synopsys_dc.setup**. The **write_design_lib_paths** command does not modify an existing file. If the file to which paths are written already exists, **write_design_lib_paths** fails.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

In the following example, **write_design_lib_paths** writes the current design library paths:

```
prompt> write_design_lib_paths -file ~/.synopsys_dc.setup -dc_setup
```

SEE ALSO

`get_design_lib_path(2)`
`read_file(2)`
`report_design_lib(2)`

write_design_settings

Displays the saved design settings.

SYNTAX

```
status write_design_settings  
-library  
[-output file]
```

Data Types

file string

ARGUMENTS

```
-library                      Displays the design settings saved in the current library.  
  
-output file                Writes the design settings to the specified file.
```

DESCRIPTION

This command displays the design settings that were previously saved into the current library with the **save_design_settings** command.

EXAMPLES

The following example displays the design settings saved in the current library.

```
prompt> write_design_settings -library  
set LIBRARY_LIB /usr/LIB/130nm/SMALL  
set LIBRARY_DESIGN /usr/DATA/Mini/SMALL  
  
set search_path                " . \  
                              $LIBRARY_DESIGN/orig_data \  
                              $LIBRARY_LIB/DB \  
                              $LIBRARY_LIB/PDB \  
                              $LIBRARY_LIB/RCXT \  
                              "  
  
set physical_library        [list design1_mw.pdb REF1.pdb REF2.pdb REF3.pdb \  
                              REF4.pdb REF5.pdb]  
  
set target_library        "REF1 _Worst.db REF2_Worst.db \  
                              REF3_Worst.db REF4_Worst.db \  
                              REF5_Worst.db "  
  
set link_library        "*" $target_library"
```

```

set mw_reference_library " \
                        $LIBRARY_LIB/MW/REF1 \
                        $LIBRARY_LIB/MW/REF2 \
                        $LIBRARY_LIB/MW/REF3 \
                        $LIBRARY_LIB/MW/REF4 \
                        $LIBRARY_LIB/MW/REF5 \
                        "

set mw_design_library design
set design_name SMALL
set DESIGN $design_name

set mw_tech_file $LIBRARY_DESIGN/orig_data/top.tf

set mdb_tlu_plus_max_file "$LIBRARY_LIB/RCXT/worst.itf.tlu"
set mdb_tlu_plus_min_file "$LIBRARY_LIB/RCXT/best.itf.tlu"
set mdb_tlu_plus_map_file "$LIBRARY_LIB/RCXT/astro.map"

set mw_power_net VDD
set mw_ground_net VSS
set mw_logic1_net VDD
set mw_logic0_net VSS

suppress_message PSYN-024
suppress_message PSYN-039
suppress_message PSYN-025 ;# physical only cells
suppress_message PSYN-036 ;# linking logical with physical library
suppress_message PSYN-040 ;# dont touch on fixed cells
suppress_message PSYN-088 ;# dont touch on pad
suppress_message PSYN-087 ;# io pad location for port
suppress_message OPT-1022 ;# io pad and usable for logic
suppress_message LINT-30 ;# unconnected ports

set bus_name_style      {[%d]}
1

```

SEE ALSO

```

restore_design_settings(2)
save_design_settings(2)

```

write_environment

Writes the variable settings and constraints for the specified cells or designs.

SYNTAX

```
return_val write_environment
[-cells cell_list | -designs design_list]
[-format dcsh | dctcl]
[-output file_name]
[-suffix suffix]
[-environment_only]
[-constraints_only]
[-no_lib_info]
[-consistency]
```

Data Types

<i>cell_list</i>	string
<i>design_list</i>	string
<i>file_name</i>	string
<i>suffix</i>	string

ARGUMENTS

-cells *cell_list*

The list of instances for which the environment is to be created. This option and the **-designs** option are mutually exclusive. You can specify one or the other or neither. If you specify neither, the command writes the environment for the current design.

-designs *design_list*

The list of designs for which the environment is to be created. This option and the **-cells** option are mutually exclusive. You can specify one or the other or neither. If you specify neither, the command writes the environment for the current design.

-format dcsh | dctcl

Specifies the format for the constraints file.
If you do not specify a format, the command uses Tcl format.

-output *file_name*

Specifies the output file name. If you specify the file name, this command generates a single file that contains the requested information. If you do not specify an output file name, this command creates an output file for each partition named *partition_name.suffix*, where the suffix value is determined by the **-suffix** option.

-suffix *suffix*

Specifies the output file suffix used by this command when you do not specify the file name. The default suffix is "env".

`-environment_only`
Writes out only the variable settings. By default, both the variable settings and the constraints are written.

`-constraints_only`
Writes out only the constraints. By default, both the variable settings and constraints are written.

`-no_lib_info`
Specifies not to include attributes that come from library cells. By default, instances inherit attributes from their reference library cells.

`-consistency`
Writes out variables, attributes, constraints etc. for consistency checking. When this option specified, `-output` option needs to be specified as well.

DESCRIPTION

For each specified design, this command writes out the following information:

- Settings for all variables whose settings have changed since the session began.
- Design constraints.
- Attributes inherited from the reference library cells

You can exclude information using the **`-constraints_only`**, **`-environment_only`**, and **`-no_lib_info`** options.

By default, an output file is generated for each design. You can generate a single output file by using the **`-output`** option.

Multicorner-Multimode Support

This command uses information from the current scenario only.

SEE ALSO

`derive_constraints(2)`

write_flip_chip_bumps

Write bump locations and connected nets to a specified AIF format file.

SYNTAX

```
status write_flip_chip_bumps  
file_name
```

Data Types

```
file_name      string
```

ARGUMENTS

```
file_name  
    Specifies the name of the output file.
```

DESCRIPTION

This command writes bump locations and net connections to a text file in the AIF format. The file can be specified with a relative or absolute path. The coordinates of the bump cells are referencing to the center of the die. But in the ICC the coordinates

of the bump cells are referencing to the left bottom of the die.

So $AIF\_Pad\_x = ICC\_Pad\_x - die\_width/2$, $AIF\_Pad\_y = ICC\_Pad\_y - ide\_hight/2$.

The following is an example of AIF file.

```
;This is a comment  
;File stars from here
```

```
[DATABASE]  
TYPE=AIF  
VERSION=2.0  
UNITS=um
```

```
[DIE]  
WIDTH=8785  
HEIGHT=8785  
NAME=DIE1G
```

```
[PADS]  
PAD1=SQ 100 10  
PAD2=RECT 200 100 10
```

```
[NETLIST]  
;Netname Pad# Type Pad_X Pad_Y Ball#  
GND BUMP_1-PAD BUMPCELL 3795.0 -3795.0  
GND BUMP_2-PAD BUMPCELL 3565.0 -3795.0  
GND BUMP_3-PAD BUMPCELL 3335.0 -3795.0  
GND BUMP_4-PAD BUMPCELL 3105.0 -3795.0
```

```
GND BUMP_5-PAD BUMPCELL 2875.0 -3795.0
GND BUMP_6-PAD BUMPCELL 2645.0 -3795.0
GND BUMP_7-PAD BUMPCELL 2415.0 -3795.0
GND BUMP_8-PAD BUMPCELL 2185.0 -3795.0
GND BUMP_9-PAD BUMPCELL 1955.0 -3795.0
GND BUMP_10-PAD BUMPCELL 1725.0 -3795.0
GND BUMP_11-PAD BUMPCELL 1495.0 -3795.0
GND BUMP_12-PAD BUMPCELL 1265.0 -3795.0
GND BUMP_13-PAD BUMPCELL 1035.0 -3795.0
GND BUMP_14-PAD BUMPCELL 805.0 -3795.0
GND BUMP_15-PAD BUMPCELL 575.0 -3795.0
GND BUMP_16-PAD BUMPCELL 345.0 -3795.0
GND BUMP_17-PAD BUMPCELL 115.0 -3795.0
GND BUMP_18-PAD BUMPCELL -115.0 -3795.0
GND BUMP_19-PAD BUMPCELL -345.0 -3795.0
GND BUMP_20-PAD BUMPCELL -575.0 -3795.0
GND BUMP_21-PAD BUMPCELL -805.0 -3795.0
```

Descriptions:

[DATABASE]

Identifies the file as an AIF file and contains the version and units of the coordinates.

[DIE]

Describes the width and height of the die and a name for the die. Width is measured along the horizontal X axis and height is measured along the vertical Y axis. The width and height are defined per die nominal described dimensions.

[PADS]

Defines all pad types needed for the die.

[NETLIST]

Defines each net and the coordinates of each die pad. Every column must have an entry.

If a particular data item is not present, a hyphen must be used as a placeholder.

EXAMPLES

The following example writes bump locations and net connections to the file bump_loc_net.

```
prompt> write_flip_chip_bumps ../scripts/bump_loc_net
```

SEE ALSO

```
read_flip_chip_bumps(2)
place_flip_chip_array(2)
expand_flip_chip_cell_locations(2)
```

write_flip_chip_nets

Writes the flip-chip nets into a text file.

SYNTAX

```
status write_flip_chip_nets  
-file_name nets_file
```

Data Types

```
nets_file      string
```

DATA TYPES

```
nets_file  string
```

ARGUMENTS:

```
-file_name nets_file  
           Specifies the net file name. This is a required option.
```

DESCRIPTION

This command writes the flip-chip nets into a text file.

A flip-chip net is a net that connects a flip-chip driver I/O cell to a bump cell (which is also called a flip-chip pad).

EXAMPLES

The following example writes the flip-chip nets into a text file called flipChipNetFile.

```
prompt> write_flip_chip_nets -file_name flipChipNetFile
```

SEE ALSO

```
set_route_flip_chip_options(2)  
route_flip_chip(2)  
create_stack_via_on_pad_pin(2)  
remove_flip_chip_route(2)  
push_flip_chip_route(2)  
display_flip_chip_route_flylines(2)  
optimize_flip_chip_route(2)
```

write_floorplan

Writes a Tcl script that can be used to recreate elements of the floorplan of the specified design.

SYNTAX

```
status write_floorplan
[-placement {io std_cell hard_macro soft_macro}]
[-create_terminal]]
[-row]
[-track]
[-no_bound]
[-create_bound]
[-no_placement_blockage]
[-no_route_guide]
[-preroute]
[-no_plan_group]
[-no_voltage_area]
[-no_create_boundary]
[-pin_guide]
[-objects heterogeneous_collection]
[-all]
[-cell design_name]
[-sm_placement {io std_cell hard_macro soft_macro}]
[-sm_placement_blockage]
[-sm_route_guide]
[-sm_plan_group]
[-sm_voltage_area]
[-sm_bound]
[-sm_cell_row]
[-sm_track]
[-sm_preroute]
[-sm_all]
file_name
```

Data Types

<i>heterogeneous_collection</i>	collection
<i>design_name</i>	string
<i>file_name</i>	string

ARGUMENTS

-placement {io std_cell hard_macro soft_macro}

Write the specified placement information. The valid values are **io**, **std_cell**, **hard_macro**, and **soft_macro**. You can specify one or more of these values.

If you specify the **io** value, you can also use the **-create_terminal** option. An I/O can be a terminal or a pad, depending on where this command is executed.

If you specify the **soft_macro** value, black box cells are also output, because they are considered soft macros.

By default, no placement information is written.

-create_terminal
Write the **create_terminal** command for terminals. This option requires that the **-placement {io}** option is also specified.

-row
Write the row information. By default, no row information is written.

-track
Write the track information. By default, no track information is written.

-no_bound
Do not write bound information. By default, bound information is written.

-create_bound
Specifies to write "create_bounds" command instead of "update_bounds" for bound objects. This option does not take any effect if **-no_bound** option is specified.

-no_placement_blockage
Do not write placement blockage information. By default, placement blockage information is written.

-no_route_guide
Do not write route guide information. By default, route guide information is written.

-preroute
Write preroute information. By default, no preroute information is written.

-no_plan_group
Do not write plan group information. By default, plan group information is written.

-no_voltage_area
Do not write voltage area information. By default, voltage area information is written.

-no_create_boundary
Do not write boundary information. By default, boundary information is written.

-pin_guide
Write pin guide information. By default, no pin guide information is written.

-objects *heterogeneous_collection*
Specifies the objects to be written by **write_floorplan**. This option takes a collection that contains one or more of the following object types: cell instance, placement blockage, route guide, plan group, voltage area, bound, die area, core area, and cell row.
The objects specified by this option are written only if their associated object type is enabled for output.

-all
Write all floorplan information. You can use **-no_XXX** options together with this option to disable the writing of certain types of information. When using

-all, -create_terminal is ignored and always turn on.

-cell *design_name*

Specifies the design name for which the floorplan information is written. By default, the floorplan information is written for the current design.

-sm_placement {io std_cell hard_macro soft_macro}

Write the specified placement information for each soft macro. By default, no placement information is written for soft macros.

-sm_placement_blockage

Write placement blockage information for each soft macro. By default, no placement blockage information is written for soft macros.

-sm_route_guide

Write route guide information for each soft macro. By default, no route guide information is written for soft macros.

-sm_plan_group

Write plan group information for each soft macro. By default, no plan group information is written for soft macros.

-sm_voltage_area

Write voltage area information for each soft macro. By default, no voltage area information is written for soft macros.

-sm_bound

Write bound information for each soft macro. By default, no bound information is written for soft macros.

-sm_cell_row

Write cell row information for each soft macro. By default, no cell row information is written for soft macros.

-sm_track

Write track information for each soft macro. By default, no track information is written for soft macros.

-sm_preroute

Write preroute information for each soft macro. By default, no preroute information is written for soft macros.

-sm_all

Write all floorplan information for each soft macro. By default, no floorplan information is written for soft macros.

file_name

Specifies the name of the file to which to write the floorplan information.

DESCRIPTION

This command writes a Tcl script file that describes the floorplan information for the current design or the user-specified design. You restore the floorplan information by reading the generated file with the **read_floorplan** command.

The **write_floorplan** command writes commands relative to the top of the design, regardless of the current instance. The output file it writes must be read from the top of the design.

By default, the floorplan information is written only for the top-level design and includes the boundary information, as well as the following objects: bounds, placement blockages, route guides, plan groups, and voltage areas.

To output hierarchical information, you must specify **-sm_xxx** options.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command writes the floorplan to a file named top.fp and restores the floorplan by reading it.

```
prompt> current_mw_cel
"top"
prompt> write_floorplan -placement {io std_cell hard_macro} \
-row -track top.fp
1
prompt> read_floorplan top.fp
```

The following command writes all placed standard cells to a file called placed_std.fp.

```
prompt> write_floorplan -placement {std_cell} \
-objects [get_cells -filter "is_placed == true"] placed_std.fp
1
```

The following command writes a hierarchical floorplan that contains floorplan for each of the soft macros to a file called hier.fp.

```
prompt> write_floorplan -sm_all hier.fp
```

SEE ALSO

current_mw_cel(2)
read_floorplan(2)

write_interface_timing

Generates an interface timing ASCII report for a gate-level netlist or an interface logic model (ILM).

SYNTAX

```
int write_interface_timing  
file_name  
[-ignore_ports port_list]  
[-significant_digit digits]  
[-nosplit]
```

Data Types

<i>file_name</i>	string
<i>port_list</i>	list
<i>digits</i>	int

ARGUMENTS

file_name
Specifies the name of the file to which the interface timing information is to be written.

-ignore_ports *port_list*
Specifies a list of ports that are to be excluded from the timing file. By default, all ports are included.

-significant_digit *digits*
Specifies the number of digits to the right of the decimal point that are to be reported following the method used in the **report_timing** command. Allowed values are 0-13. The default value is 3. Use this option if you want to override the default. Fixed-precision floating-point arithmetic is used for delay calculation; thus the actual precision might depend on the platform.

-nosplit
The **-nosplit** option prevents line-splitting. This is most useful for doing diffs on previous scripts or for postprocessing the script.

DESCRIPTION

Generates an interface timing ASCII report for a gate-level netlist or an interface logic model (ILM). The report is a file containing interface timing information for a gate-level netlist or an ILM. The command performs an implicit **update_timing** if necessary for the selected data sections. A return code of **1** indicates that the command ran to completion; a **0** indicates an error.

You normally use the **compare_interface_timing** command to compare two report files that have been generated by the **write_interface_timing** command.

The output report file contains the following sections:

a) **Worst Slack Section** – Contains slack values of critical paths starting from input ports or ending at output ports. For each critical path, it reports slack values for following arc types: min_rise, min_fall, max_rise, max_fall.

b) **Type Section** – Contains worst-case max/min arc values for critical paths starting from input ports or ending at output ports. For each path timing relationship, it reports arc values for the following possible arc types: min_rise, min_fall, max_rise, max_fall.

EXAMPLES

The following example writes timing information for the current design to the *model.rpt* file.

```
prompt> write_interface_timing model.rpt
```

SEE ALSO

```
compare_interface_timing(2)  
update_timing(2)
```

write_io_constraints

Writes an I/O constraints file.

SYNTAX

```
status write_io_constraints  
[-library lib_name]  
[-cell cell_name]  
[-constraint_type side_only | side_order | side_location]  
[-pin_only | -pad_only]  
[-objects object_list]  
file_name
```

Data Types

<i>lib_name</i>	string
<i>cell_name</i>	string
<i>object_list</i>	collection
<i>file_name</i>	string

ARGUMENTS

- library *lib_name***
Specifies the name of the library in which the Milkyway cell is open. If the library is open, you do not need to specify this option.
- cell *cell_name***
Specifies the name of the Milkyway cell from which the I/O constraints are extracted. If the Milkyway cell is open, you do not need to specify this option.
- constraint_type *side_only* | *side_order* | *side_location***
Specifies the constraint type. By default, this command writes side and location information in the I/O constraints file.
The supported values are mutually exclusive. Use only one.
- **side_only**
When **-constraint_type side_only** is specified, the command writes the side constraints based on the actual locations of the specified or current Milkyway cell into the constraints file.
 - **side_order**
When **-constraint_type side_order** is specified, the command writes the side and order constraints based on the actual locations of the specified or current Milkyway cell into the constraints file.
 - **side_location** (the default)
When **-constraint_type side_location** is specified, the command writes the side and location constraints based on the actual locations of the specified or current Milkyway cell into the constraints file. into file.

-pin_only

Specify this option to write only pin constraints.

This option should be used with the **-constraints** option. This option and the **-pad_only** option are mutually exclusive; specify only one.

-pad_only

Specify this option to write only pad constraints.

This option should be used with the **-constraints** option. This option and the **-pin_only** option are mutually exclusive; specify only one.

-objects *object_list*

Specifies the objects whose constraints are to be dumped. The objects include terminals, nets and pads. However when **-cell** option is specified, **-objects** accepts soft macro pins to be in **object_list**.

If you do not specify this argument, the command processes all pins/terminals or pads of the top cell, and their constraints are written to the specified file.

file_name

Specifies the name of the file to which the I/O constraints are written.

DESCRIPTION

This command writes the physical constraints information based on the actual locations of the specified or the current Milkyway cell to the specified file. The constraint file uses the Tcl format.

The contents in the database will not change. If the physical constraints based on the actual locations need to be stored in the database, use the **write_io_constraints** command to write the constraints to a file first, and then use the **read_io_constraints** command to read the constraints into the database from that file.

For the layer constraint, the **write_io_constraints** command writes the layer mask name.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example writes I/O constraints for the MULT16 Milkyway cell to a disk file called mult16.tcl.

```
prompt> write_io_constraints -lib design -cell MULT16 \  
-constraint_type side_order -pin_only mult16.tcl
```

SEE ALSO

`read_io_constraints(2)`
`report_io_constraints(2)`

```
remove_io_constraints(2)
initialize_floorplan(2)
```


write_lib

Writes a compiled library to disk in Synopsys database, EDIF, or VHDL format.

SYNTAX

```
int write_lib  
library_name  
[-format db | edif | vhdl]  
[-compress compression_format]  
[-output file_name]  
[-names_file file_list]  
[-macro_only]
```

Data Types

<i>library_name</i>	string
<i>file_name</i>	string
<i>file_list</i>	list

ARGUMENTS

library_name

If you are writing in the Synopsys database (db) format, this argument specifies the name of the technology or symbol library to save. If you are writing in EDIF format, this argument specifies the name of the symbol library used to generate the EDIF library. If you are writing in VHDL format, the argument indicates the name of the technology library used to generate the VHDL library.

-format db | edif | vhdl

Specifies the external format of the specified library. The default is *db*, the Synopsys database format.

-compress *compression_format*

This option has effect only if the output format is db. For db format, only gzip is allowed as the *compression_format*, and a file name extension of .gz will be added if it is not already specified.

-output *file_name*

Specifies an output filename or pathname for the library.

-names_file *file_list*

When used in conjunction with the **-format edif** option, this option specifies a set of one or more name-mapping files. Design Compiler's EDIF writer uses this set of files to change the names of symbols or symbol pins in the incoming EDIF file (*file_name*) to resolve name-based consistency issues. If these files are required, you must create them before executing the **write_lib** command. When specifying more than one name-mapping file, enclose them in braces ({}).

-macro_only

When used for a physical library, this option allows only the macro

information being written out to Synopsys database format. And the physical technology information will not be written out. By default this option is off, and both technology information and macro information are written out to Synopsys database format.

DESCRIPTION

The **write_lib** command saves to disk a compiled technology or symbol library in db format.

For *db* format, if the **-output** option is used, the disk file is written to the specified filename. If **-output** is not used, the library is written to the current directory with filename *library_name.db*.

If the format is EDIF, **write_lib** takes a compiled symbol library, generates an EDIF library, and writes the library to disk.

The **-names_file** option for the **write_lib** command enables you to change the names of symbol and symbol pins. The general format of the name-mapping files is the same for the **write_lib** command as it is for the **change_names** command. The object types for the **write_lib** command are *reference* or *component* (to change symbol names), and *in* (to change symbol pin names).

The **-macro_only** option for the **write_lib** command enables you to write out only the macro information in a physical library to Synopsys database format. And the physical technology information will not be written out. By default this option is off, and both technology information and macro information are written out to Synopsys database format.

If the format is VHDL, **write_lib** takes a compiled technology library, generates a detailed VHDL library report, and library and writes the library to disk.

The VHDL library is configured based on the values of the following *dc_shell* or *lc_shell* environment variables:

```
vhdl1lib_architecture
vhdl1lib_glitch_handle
vhdl1lib_logic_system
vhdl1lib_logical_name
vhdl1lib_pulse_handle
vhdl1lib_tb_compare
vhdl1lib_tb_x_eq_dontcare
vhdl1lib_timing_checks
vhdl1lib_timing_mesg
vhdl1lib_timing_xgen
```

Based on the VHDL library configuration, the following files may be created:

- a components file with the name *library_name_components.vhd*
- a (VITAL) components file with the name *library_name_Vcomponents.vhd*
- a (VITAL) package file with the name *library_name_Vtables.vhd*
- an entity/architecture file with the name *library_name_FTGS.vhd*
- an entity/architecture file with the name *library_name_VITAL.vhd*

- an entire library testbench file with the name *library_name_tb.vhd*
- a testbench file for every cell with the name *library_name_cell_tb.vhd*
- an input stimulus file with the name *library_name_tb.sen*
- a verification script file with the name *library_name.csh*

If the **-output** option is used, the disk files are written with the specified principal and extension filename.

Before writing a library, you must first read the technology library file into the system using the **read_lib** command. **read_lib** automatically compiles the file and activates the **write_lib** command. Otherwise, the built-in security mechanism aborts **write_lib** before generating a VHDL library.

Note: This command overwrites files without warning.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reads and writes a technology library.

```
prompt> read_lib my_lib.lib
prompt> write_lib my_lib
```

The following example reads and writes a symbol library.

```
prompt> read_lib schem_lib.slib
prompt> write_lib schem_lib.sdb
```

The following example reads a symbol library and writes an EDIF symbol library.

```
prompt> read_lib schem_lib.slib
prompt> write_lib -f edif -output schem_lib.edif schem_lib.sdb
```

The following example writes a technology library to a specified file.

```
prompt> write_lib my_lib -output ~synopsys/test/test.db
```

The following example reads a technology library, sets the target architecture to FTGS, and writes a VHDL library as *my_lib_FTGS.vhd* and "*my_lib_components.vhd*". The log report is redirected to *my_lib.LOG*.

```
prompt> read_lib mylib.lib
prompt> vhdllib_architecture=FTGS
prompt> write_lib -f vhd1 my_lib > my_lib.LOG
```

The following example reads a technology library, sets the target architecture to FTGS, and writes a VHDL library to files, *spec_FTGS.vhd* and *spec_components.vhd*. A log file is displayed.

```

prompt> read_lib mylib.lib
prompt> vhdllib_architecture=FTGS
prompt> write_lib -f vhd1 -output spec.vhd my_lib

```

The following example shows a name-mapping file called lib.names and its usage.

Design	Type	Old Name	New Name
lib	reference	NR2	nr2
NR2	pin A	a	
NR2	pin B	b	
NR2	pin Z	z	
lib	reference	HO22	aoi2
HO22	pin I1	a	
HO22	pin I2	b	
HO22	pin I3	c	
HO22	pin O1	z	

```

prompt> read_lib foo.slib

```

```

prompt> write_lib foo -format EDIF -names_file lib.names -o foo.edif

```

SEE ALSO

```

compare_lib(2)
read_lib(2)

```

write_lib_specification_model

Command to dump out PG library on-the-fly conversion and specification TCL commands into output file.

SYNTAX

Boolean **write_lib_specification_model**
[*output_file_name*]

ARGUMENTS

output_file_name
Specify the output file to dump out all the commands.

DESCRIPTION

This **write_lib_specification_model** command will write out library updating commands and on-the-fly specification commands into output file, plus the `update_lib_model` command itself. This TCL file can be sourced in to do the entire library updating without reference library(FRAM). Libraries will be updated to PG library.

EXAMPLES

```
prompt> write_lib_specification_model library.tcl
```

SEE ALSO

```
update_lib_voltage_model(2)  
update_lib_pg_pin_model(2)  
update_lib_pin_model(2)  
update_lib_model(2)  
set_voltage_model(2)  
set_pg_pin_model(2)  
set_pin_model(2)  
set_always_on_cell(2)  
set_retention_cell(2)  
set_level_shifter_cell(2)  
set_isolation_cell(2)  
set_power_switch_cell(2)
```

write_link_library

Writes shell commands to save the current link library settings for design instances.

SYNTAX

```
int write_link_library  
[-full_path_lib_names] [-nosplit]  
[-full_path_lib_names] [-nosplit]  
[-output file_name]  
[-target target]
```

Data Types

<i>file_name</i>	string
<i>target</i>	string

ARGUMENTS

-full_path_lib_names
Writes library names with full pathnames. The default is to write only the library name without the path.

-nosplit
Indicates that lines are not to be split when column fields overflow. This is most useful for diffing on previous scripts or for post-processing the script.

-output *file_name*
Writes the script to the specified file. By default, the command writes shell commands to standard output.

-target *target*
Specifies the target to write out scripts. The only valid value is **ptsh**.

DESCRIPTION

This command writes a script of shell commands. It is used to recreate link library settings for each instance in the design.

The redirection operator (>) can be used to redirect the output of this command to a disk.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows output redirected to a file named *output.tcl* and uses the full pathname for output for the link libraries.

```
prompt> write_link_library -output output.tcl \  
-full_path_lib_names
```

SEE ALSO

link_library(3)

write_mw_lib_files

Writes the technology, or plib, or reference control file of the Milkyway library.

SYNTAX

```
status_value write_mw_lib_files  
[-technology]  
[-plib]  
[-reference_control_file]  
-output file_name  
libName
```

Data Types

<i>file_name</i>	string
<i>libName</i>	string

ARGUMENTS

-technology
Indicates to dump technology information.
This option and the **-plib** option, the **-reference_control_file** option are mutually exclusive.

-plib
Indicates to dump plib file.
This option and the **-technology** option, the **-reference_control_file** option are mutually exclusive.

-reference_control_file
Indicates to dump the reference control file.
This option and the **-technology** option, the **-plib** option are mutually exclusive.

-output *file_name*
Specifies the file name in which the technology information or the reference library information is stored.

libName
Specifies the Milkyway library to be reported.

DESCRIPTION

Dumps the technology information or the reference library information to an ASCII file that you can edit.

At least one of the **-technology**, or **-plib**, or the **-reference_control_file** options must be specified.

A status indicating success or failure is returned.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example displays unit information about a Milkyway library:

```
prompt> write_mw_lib_files -reference_control_file -output ref.out design
1
```

SEE ALSO

```
create_mw_lib(2)
set_mw_lib_reference(2)
set_mw_technology_file(2)
```

write_parasitics

Writes parasitics to a disk file for delay calculation tools.

SYNTAX

```
status write_parasitics
[-output file_name]
[-format SPEF | SBPF]
[-compress]
[-routed_nets_only]
[-no_name_mapping]
```

Data Types

file_name string

ARGUMENTS

-output *file_name*

Specifies the name of the output file to which parasitics for the current design are written. If you do not specify this option, the following naming convention is used for the output file name:

design_name.format_name.

In the multicorner-multimode flow while writing the parasitic file, the tool uses the name of the t1plus file and the temperature associated with the corner, along with the file name you specified, to derive the file name of the parasitic file.

If you specify the scaling factors for capacitance, resistance and coupling capacitance by using the **set_extraction_options** command you will see a scaling section in the output file name.

The naming convention used by the tool is in the following format:

<t1plus file name>.<temperature>.<user scaling>

The number in the scaling section <user scaling> is in the following format:

<res>_<cap>_<ccap>

where *design_name* is the name of the current design.

-format SPEF | SBPF

Specifies whether to write parasitics in Standard Parasitic Exchange Format (SPEF) or Standard Binary Format (SBPF) format. If you do not specify this option, parasitics are written in SPEF format.

-compress

Specifies that the generated SPEF file is to be compressed in gzip format. The name of the compressed file will then be the name of the output file with .gz appended to the file name. This option is valid only when you write parasitics in SPEF format. By default, output is not compressed.

-routed_nets_only

Writes parasitics only for fully routed nets. By default, parasitics are

output for both routed and unrouted nets. Use this option to revert to the default behavior of versions prior to A2007.12.

-no_name_mapping

Specifies whether the net name will be used directly in the file. This option is valid only when you write parasitics in SPEF format. With no name mapping, the actual net names are used in the .spef file. With name mapping, the net names are mapped to numbers that are later used when writing RC details.

DESCRIPTION

This command writes parasitics for the current design to a disk file.

Multicorner-Multimode Support

This command uses information from active scenarios only.

EXAMPLES

The following example writes parasitics for all the nets in the current design. If there are minimum and maximum operating conditions and the output format is SPEF, parasitics for both conditions are written.

```
prompt> write_parasitics -format spef -output top
```

```
*SPEF 1.0
*DESIGN "filter"
*DATE "Tue May 28 10:01:39 1996"
*VENDOR "SYNOPTICS INC"
*PROGRAM "Synopsys Design Compiler cmos"
*VERSION "3.5a-SI2-SCM"
*DESIGN_FLOW "SYNTHESIS"
*DIVER /
*DELIMITER :
*T_UNIT 1.0 NS
*C_UNIT 1.0 PF
*R_UNIT 1.0 KOHM
*L_UNIT 1.0 HENRY
```

```
*PORTS
```

```
micro_d_1 I *L 0.000 *S 2.980 2.980
micro_d_2 I *L 0.000 *S 2.980 2.980
micro_d_3 I *L 0.000 *S 2.980 2.980
micro_d_4 I *L 0.000 *S 2.980 2.980
micro_d_5 I *L 0.000 *S 2.980 2.980
micro_d_6 I *L 0.000 *S 2.980 2.980
micro_d_7 I *L 0.000 *S 2.980 2.980
micro_d_8 I *L 0.000 *S 2.980 2.980
micro_d_9 I *L 0.000 *S 2.980 2.980
micro_d_10 I *L 0.000 *S 2.980 2.980
```

```
micro_d_11 I *L 0.000 *S 2.980 2.980
micro_d_12 I *L 0.000 *S 2.980 2.980
preset\[15\] I *L 0.000 *S 0.249 0.204 *D IVA
```

```
*D_NET W03[14] 3.350e-02
```

```
*CONN
```

```
*P W03[14] O *L 0.0
```

```
*I U111112:ZN O *L 0.0
```

```
*CAP
```

```
1 W03[14] 1.300e-03
```

```
2 U111112:ZN 0.0
```

```
3 W03[14]:16 1.300e-03
```

```
4 W03[14]:6 1.301e-04
```

```
5 W03[14]:14 2.380e-03
```

```
6 W03[14]:5 1.301e-04
```

```
*RES
```

```
1 W03[14] W03[14]:16 1.880e-02
```

```
2 U111112:ZN W03[14]:6 1.600e-03
```

```
3 W03[14]:16 W03[14]:14 1.600e-03
```

```
4 W03[14]:6 W03[14]:5 9.081e-04
```

```
5 W03[14]:14 W03[14]:19 2.792e-02
```

```
*END
```

SEE ALSO

```
extract_rc(2)
read_parasitics(2)
set_load(2)
set_resistance(2)
```

write_physical_constraints

Writes floorplan information in Tcl format for use in Design Compiler topographical mode.

SYNTAX

```
status write_physical_constraints  
-output tcl_file  
[-port_side]
```

Data Types

tcl_file string

ARGUMENTS

-output *tcl_file*
 Specifies the name of the file to which the physical constraints are written.
 This argument is required.

-port_side
 Writes port side constraints in addition to writing out the default floorplan information.

DESCRIPTION

Writes floorplan information in Tcl format to the specified file. This file is for use in Design Compiler topographical mode. The default floorplan information includes core area, port locations, I/O pad locations and orientations, macro locations and orientations, and placement blockages.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example writes floorplan information to the file named a.out:

```
prompt> write_physical_constraints -output a.out
```

The following example writes floorplan information and port side constraints to the file named b.out:

```
prompt> write_physical_constraints -output b.out -port_side
```

SEE ALSO

`create_bounds(2)`
`create_net_shape(2)`
`create_placement_blockage(2)`
`create_site_row(2)`
`create_voltage_area(2)`
`set_cell_location(2)`
`set_port_location(2)`

write_physical_script

Writes a Tcl script to save the current physical settings.

SYNTAX

```
status write_physical_script  
-mpc  
-nosplit  
-output file_name
```

ARGUMENTS

-mpc
Write out only the mpc constraints

-nosplit
Do not split each command if long to multiple lines, default is to split into multiple lines.

-output *file_name*
Indicates that the script is to be written to the specified file.

DESCRIPTION

The **write_physical_script** command writes a Tcl script that recreates the physical attributes on the current design.

User-defined attributes are not supported. Attributes and objects created by the following commands are supported:

```
set_floorplan_options  
set_floorplan_pnet_options  
set_floorplan_port_options  
set_floorplan_macro_options  
set_floorplan_macro_array
```

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

In the following example, the settings on the current design are written to the file test.scr.

```
prompt> write_physical_script -mpc -output test.scr
```

SEE ALSO

`current_design(2)`
`reset_design(2)`
`write(2)`

write_plib

Writes the library data of the specified library to a file in Synopsys physical library (.plib) format.

SYNTAX

```
int write_plib  
[-lib_name lib_name]  
[-cell_name cell_name]  
[-ignore_tech_info]  
[-signal_em_only]  
[-antenna_rule_only]  
[-cell_info]  
[-ignore_cell_geom]  
[-antenna_prop_only]  
[-metal_density_only]  
[-basic_cell_pin_info_only]  
plib_file_name
```

Data Types

<i>lib_name</i>	string
<i>cell_name</i>	string
<i>plib_file_name</i>	string

ARGUMENTS

- lib_name *lib_name***
Specifies the Milkyway library that contains the design cell. The *lib_name* can include the path of the library. If it does not, the library is resolved from the current working directory.
- cell_name *cell_name***
Specifies the cell names that physical layout information is written to in the macro section of the .plib file. These cells should exist in the FRAM view. If you want to write multiple cells, you should delimit the cell name list by using quotation marks (") and separate the names with a comma (,). For example, the *cell_name* argument could be "AND2, BUFF2".
- ignore_tech_info**
Causes technology information *not* to be written out. By default, the command writes out technology information to the .plib file.
- signal_em_only**
Writes out only signal EM information.
- antenna_rule_only**
Writes out only antenna rules information.
- cell_info**
Writes out cell information. By default, the command does not write out cell information to the .plib file.

`-ignore_cell_geom`

Causes the command *not* to write out all cell geometries. Cell information includes only macro attributes and pin attributes. By default, when the **-cell_info** option is specified, the command writes out physical cell layout information to the .plib file.

`-antenna_prop_only`

Writes out only antenna property information.

`plib_file_name`

Specifies the name of the .plib file to which the command writes library information. If the file already exists, the command overwrites it. If the file does not exist, the command creates it. The *plib_file_name* argument can include the path of the output file. If it does not, the file is written to the current working directory.

DESCRIPTION

This command writes the library data of the specified library to a file in .plib format.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example writes out a complete .plib file named *out.plib* from a library named *cells*.

```
prompt> write_plib -lib_name cells -cell_info out.plib
```

SEE ALSO

`write_mw_lib_files(2)`

write_qtm_model

Writes the Quick Timing Model (QTM) file.

SYNTAX

```
string write_qtm_model  
-out_dir output_qtm_directory  
[-text]
```

Data Types

output_qtm_directory string

ARGUMENTS

```
-out_dir output_qtm_directory  
        Specifies the name of the output QTM directory  
  
-text  
        To write out text format file for the QTM model in addition to the .db format  
        file.
```

DESCRIPTION

This command writes out the QTM model file in Synopsys .db format.

If the *-text* option is provided then the QTM model file is written out in text format also.

This command writes out all the QTM Model(s) in the internal QTM database with different names in one output directory. You must provide the output directory as a command-line option.

The name of the QTM Model is given in **create_qtm_model** command. The name of the output QTM Model file is derived from the QTM Model Name. The output QTM Model file name is *<ModelName>.qtm.db*.

This command writes out the in-memory .db to on-disk .db file. This command runs slower than **save_qtm_model** command and requires disk I/O. Usually, this command is expected to be called once at the end of an ICC session to get a final version of the QTM model.

For a more detailed description about QTM please refer to the *ICC Design Planning User Guide*.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example writes the QTM Model **adder** in internal QTM database in directory **qtm** as **adder.qtm.db**. This will generate a file **./qtm/adder.qtm.db**

```
prompt> write_qtm_model -out_dir qtm
```

SEE ALSO

```
create_qtm_model(2)  
save_qtm_model(2)
```

write_route

Writes the routing information to the specified file.

SYNTAX

```
status write_route  
-output file_name  
[-nets collection_of_nets]  
[-skip_route_guide]  
[-output_metal_fill]  
[-objects collection_of_objects]
```

ARGUMENTS

-output *file_name*
Specifies the output file name. Routing information is written to *file_name*. This is a required argument.

-nets *collection_of_nets*
Specifies the collection of nets for which to write out routing information. If you specify more than one net, separate them with a space. If you do not use the **-nets** option, the default is to write out all routing information.

-skip_route_guide
Do not write route guide information to the output file. By default, route guide information is written.

-output_metal_fill
Write metal fill information to the output file. By default, metal fill information is not written.

-objects *collection_of_objects*
Specifies a collection of wires and/or vias. The **-nets** and **-objects** are mutually exclusive.

DESCRIPTION

The **write_route** command outputs the routing information as a set of tcl commands to the specified file. Users can source the file back into IC Compiler.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following examples write the routing information to a file called my_route.txt.

```
prompt> write_route -output my_route.txt
```

```
prompt> write_route -objects [get_selection] -output my_route.txt
```

SEE ALSO

```
write_def(2)  
create_net_shape(2)  
create_via(2)  
create_user_shape(2)  
create_route_guide(2)
```

write_rp_groups

Writes out the relative placement constraints for the specified relative placement groups.

SYNTAX

```
collection write_rp_groups  
rp_groups | -all  
[-hierarchy]  
[-quiet]  
[-nosplit]  
[-output filename]  
[-create]  
[-leaf]  
[-keepout]  
[-instance]  
[-include]
```

Data Types

<i>rp_groups</i>	list or collection
<i>filename</i>	string

ARGUMENTS

rp_groups
Specifies the relative placement groups to be written out to the script. This option is mutually exclusive with the **-all** option.

-all
Specifies that all relative placement groups are to be written out to the generated script. This option cannot be used with either the *rp_groups* argument or the **-hierarchy** option.

-hierarchy
Specifies that all relative placement groups within the hierarchy of the groups in *rp_groups* are to be written out. By default, subgroups are not written out. This option is mutually exclusive with the **-all** option.

-quiet
Turns off informational messages that would otherwise be issued if no groups are written.

-nosplit
Specifies not to split long commands into multiple lines. The default is to split long commands into multiple lines.

-output *filename*
Specifies that the script is to be written to *filename*. The default is to write to standard output.

-create
Specifies that the **create_rp_group** commands are included in the generated script.

-leaf
Specifies that the **add_to_rp_group -leaf** commands are included in the generated script.

-keepout
Specifies that the **add_to_rp_group -keepout** commands are included in the generated script.

-instance
Specifies that the **add_to_rp_group -hierarchy -instance** commands are included in the generated script.

-include
Specifies that the **add_to_rp_group -hierarchy** commands (without **-instance**) are included in the generated script.

DESCRIPTION

The **write_rp_groups** command writes the relative placement constraints for the specified relative placement groups. You can use the generated commands to recreate the relative placement groups and their items on the same designs.

When you specify any of the **-create**, **-leaf**, **-keepout**, **-instance**, and **-include** options, the generated script generated contains only the commands related to the specified options. If you do not specify any of these options, all commands are output to the generated script.

The command returns a collection that contains the relative placement groups that are written out. If no objects are written out, the empty string is returned.

Relative placement usage is available with Design Compiler Ultra.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses **write_rp_groups** to recreate relative placement groups after their removal:

```
prompt> get_rp_groups
{mul::grp_mul ripple::grp_ripple example3::top_group}

prompt> write_rp_groups -all-out my_groups.tcl
{mul::grp_mul ripple::grp_ripple example3::top_group}
```



```
prompt> remove_rp_groups -all -quiet  
1
```

```
prompt> get_rp_groups  
Error: Can't find objects matching '*'. (UID-109)
```

```
prompt> source my_groups.tcl  
{example3::top_group}
```

```
prompt> get_rp_groups  
{example3::top_group ripple::grp_ripple mul::grp_mul}
```

SEE ALSO

```
add_to_rp_group(2)  
create_rp_group(2)  
remove_rp_groups(2)
```

write_saif

Writes a backward Switching Activity Interchange Format (SAIF) file.

SYNTAX

```
status write_saif  
-output file_name  
[-instances instances]  
[-no_hier]  
[-rtl]  
[-propagated]  
[-exclude_sdpgd]
```

Data Types

```
file_name      string  
instances     list
```

ARGUMENTS

- output *file_name*
Specifies the name of the output SAIF file.
- instances *instances*
Specifies an optional list of instances for which the SAIF file will be generated. If this argument is not specified, then the SAIF file will be generated for the current instance.
- no_hier
This flag specifies that the SAIF file will contain switching activity information for only the top-level design objects. When this flag is not specified, **write_saif** generates a SAIF file containing switching activity information for all the objects in the hierarchy.
- rtl
Specifies that the generated SAIF file contains the switching activity for the synthesis invariant objects. These are the design ports, hierarchical cell pins, and outputs of sequential and tri-state cells. When this option is used on an unmapped design, the generated SAIF file will be similar to one generated by RTL simulation flow using an RTL SAIF file. The generated SAIF file can then be read correctly into the synthesised design. If this option is not used when writing a SAIF file for an unmapped design, then the generated SAIF file may not be read correctly after the design has been synthesised, although it can be read correctly on the unmapped design. Note that the *-rtl_direct* of the **read_saif** command should not be used when reading SAIF files generated by **write_saif**.
- propagated
When this option is used, **write_saif** will propagate the user annotated switching activity to estimate the switching activity of unannotated objects, and generate a SAIF file containing both the annotated and estimated switching activity. When this option is not specified, the **write_saif** command

generates a SAIF file containing the user-annotated switching activity only.

-exclude_sdpd

Does not output State Dependent static probabilities and State Dependent and/or Path Dependent toggle rates in the SAIF file.

DESCRIPTION

The **write_saif** command generates a backward SAIF file containing the design switching activity information.

The **write_saif** command can be used to generate both user-annotated and propagated switching activity information. If the **-propagated** flag is used then the **write_saif** command will propagate the switching activity information and generate a SAIF file with both user-annotated and propagated switching activity. If the **-propagated** flag is not used then only the user-annotated switching activity is generated.

The top-level instance name in the generated SAIF file is the current design name. So this name needs to be specified as the instance name when reading the SAIF file with the **read_saif** command.

When the **-exclude_sdpd** flag is used, the generated SAIF file contains only simple switching activity information: the toggle rate and static probability of the design nets, ports, and cell pins that are not connected to nets.

Note that the related clock information cannot be included in the SAIF file, and the toggle rates in the SAIF file will be converted to those relative to the synthesis time unit automatically.

EXAMPLES

The following example shows how a SAIF file containing the user annotated switching activity information on all design objects can be generated.

```
prompt> write_saif -output design.saif
```

The following example shows how a SAIF file containing both user annotated and propagated switching activity information can be generated.

```
prompt> write_saif -output design.saif -propagated
```

The following example shows how a SAIF file on a synthesised design object can be generated by **write_saif** and read back using **read_saif**.

```
prompt> write_saif -output design.saif
```

```
prompt> reset_switching_activity
```

```
prompt> read_saif -input design.saif -instance top
```

The following example shows how to generate a SAIF file with user-annotated and propagated switching activity, and which does not contain SDPD information.

```
prompt> write_saif -output design.saif -propagated -exclude_sdpd
```

The following example shows how an RTL SAIF file can be generated from an unmapped design and read in back into the synthesised design.

```
prompt> write_saif -output rtl.saif -rtl
```

```
prompt> reset_switching_activity
```

```
prompt> compile
```

```
prompt> read_saif -input rtl.saif -instance top
```

SEE ALSO

```
set_switching_activity(2)  
reset_switching_activity(2)  
read_saif(2)  
merge_saif(2)  
report_saif(2)
```

write_script

Writes shell commands to save the current settings.

SYNTAX

```
int write_script  
[-no_annotated_check] [-no_annotated_delay] [-no_cg]  
[-full_path_lib_names] [-nosplit]  
[-format dtc1 | dcsh]  
[-include loop_breaking]  
[-output file_name]
```

ARGUMENTS

-no_annotated_check

Indicates that **set_annotated_check** commands are not to be written. By default, annotation commands are written to standard output. Use this option to avoid creating a very large script for designs that contain a large amount of annotated information. Annotated timing checks can be written to a file using the **write_timing** command.

-no_annotated_delay

Indicates that **set_annotated_delay** commands are not to be written. By default, annotation commands are written to standard output. Use this option to avoid creating a very large script for designs that contain a large amount of annotated information. Annotated delays can be written to a file using the **write_timing** command.

-no_cg

Indicates that Power Compiler clock gating attributes are not to be written. By default, **set_attribute** commands are written to standard output for all relevant clock gating attributes. Use this option if clock gate information is not needed for later flow.

-full_path_lib_names

Indicates that library names are to be written with full pathnames. The default is to write only the library name, without the path.

-nosplit

Indicates that lines are not to be split when column fields overflow. This is most useful for doing diffs on previous scripts, or for post-processing the script.

-format dtc1 | dcsh

Indicates that the script is to be written in Tcl mode or dcsh mode.

-include loop_breaking

Write the **set_disable_timing** commands for internally disabled timing arcs.

-output *file_name*

Indicates that the script is to be written to the specified file.

DESCRIPTION

The **write_script** command writes a script of dc_shell commands. This command is used to recreate the attributes on the current design.

By default, the **write_script** command writes dc_shell commands to standard output.

The redirection operator (>) can be used to redirect the output of this command to a disk file.

User-defined attributes are not supported. Attributes and objects created by the following dc_shell commands are supported:

```
create_clock  
group_path  
set_annotated_check  
set_annotated_delay  
set_boundary_optimization  
set_disable_timing  
set_dont_retime  
set_dont_touch  
set_dont_touch_network  
set_drive  
set_driving_cell  
set_equal  
set_false_path  
set_fanout_load  
set_fix_hold  
set_fix_multiple_port_nets  
set_flatten  
set_implementation  
set_input_delay  
set_ideal_latency  
set_ideal_net  
set_ideal_network  
set_ideal_transition  
set_load  
set_local_link_library  
set_logic_one  
set_logic_zero  
set_logic_dc  
set_max_area  
set_max_delay  
set_max_fanout  
set_max_power  
set_max_time_borrow  
set_max_transition  
set_min_delay  
set_multicycle_path  
set_operating_conditions  
set_opposite  
set_optimize_registers  
set_output_delay  
set_register_type  
set_resistance
```

```

set_rtl_load
set_signal_type
set_structure
set_switching_activity
set_test_assume
set_test_hold
set_timing_ranges
set_unconnected
set_ungroup
set_wire_load_model
set_wire_load_mode
set_wire_load_selection_group
set_wire_load_min_block_size
set_wired_logic_disable

```

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

In the following example, the settings on the current design are written to the file test.scr.

```
prompt> write_script > test.scr
```

In the following example, the settings on the current design are written to standard output. An example of the output file is also provided.

```

prompt> write_script
/*****

Created by write_script() on Fri Jun 21 17:57:25 1991

*****/
/* Set the current_design */
current_design TOP

set_operating_conditions WCCOM
set_wire_load_model WL_SMALL_BLOCK
create_clock -period 100 -waveform {0 5} [get_ports c1]
set_input_delay 1.5 -clock c1 [get_ports B]
set_input_delay 2.8 -clock c1 [get_ports A]

```

SEE ALSO

```

current_design(2)
reset_design(2)
write(2)

```

write_sdc

Writes out a script in Synopsys Design Constraints (SDC) format.

SYNTAX

```
int write_sdc  
file_name  
[-nosplit]  
[-version sdc_version]
```

Data Types

file_name string

ARGUMENTS

file_name
 Specifies the name of the file to which the SDC script is to be written.

-nosplit
 Indicates that lines are not to be split when column fields overflow. This is most useful when comparing previous script files with the UNIX diff command, or for post-processing the script.

-version *sdc_version*
 Specifies the version of SDC to write. Allowed values are **1.2**, **1.3**, **1.4**, **1.5**, **1.6**, **1.7**, and **latest** (the default).

DESCRIPTION

The **write_sdc** command writes out a script file in the latest Synopsys Design Constraints (SDC) format. This script contains commands that can be used with PrimeTime or with Design Compiler. SDC is also licensed by external vendors through the Tap-in program. SDC-formatted script files are read into PrimeTime or Design Compiler using the **read_sdc** command.

By default, the script is written as simple text. Also, by default the latest version of SDC is written. Some earlier versions of SDC can be written using the **-version** option.

SDC is a subset of the commands already supported by Design Compiler, Physical Compiler, IC Compiler, Astro, Jupiter, and PrimeTime. Of the commands supported in the latest SDC version, the following may be written by **write_sdc**. Those added for versions 1.3 and later are noted.

General Purpose Commands:

expr
list
set

Object Access Functions:

- all_clocks
- all_inputs
- all_outputs
- all_registers (1.7)
- current_design
- current_instance
- get_cells
- get_clocks
- get_libs
- get_lib_cells
- get_lib_pins
- get_nets
- get_pins
- get_ports
- set_hierarchy_separator

Basic Timing Assertions:

- create_clock
- create_generated_clock (1.3)
- group_path (1.7)
- set_clock_gating_check
- set_clock_groups (1.7)
- set_clock_latency
- set_clock_sense (1.7)
- set_clock_transition
- set_clock_uncertainty
- set_false_path
- set_ideal_latency (1.7)
- set_ideal_transition (1.7)
- set_input_delay
- set_max_delay
- set_min_delay
- set_multicycle_path
- set_output_delay
- set_propagated_clock

New options to existing supported commands:

- create_clock -add (1.4)
- create_generated_clock -add (1.4)
- create_generated_clock -master_clock (1.4)
- create_generated_clock -combinational (1.7)

Secondary Assertions:

- set_disable_timing
- set_max_time_borrow

Environment Assertions:

- create_voltage_area (1.6)
- set_case_analysis

```

set_drive
set_driving_cell
set_fanout_load
set_input_transition
set_ideal_network (1.7)
set_level_shifter_strategy (1.6)
set_level_shifter_threshold (1.6)
set_load
set_logic_dc
set_logic_one
set_logic_zero
set_max_area
set_max_capacitance
set_max_dynamic_power (1.4)
set_max_fanout
set_max_leakage_power (1.4)
set_max_transition
set_min_capacitance
set_min_fanout
set_min_porosity (1.4)
set_operating_conditions
set_port_fanout_number
set_resistance
set_wire_load_min_block_size
set_wire_load_mode
set_wire_load_model
set_wire_load_selection_group

```

Like **write_script**, the **write_sdc** command writes out commands relative to the top of the design, regardless of the current instance. SDC files written by **write_sdc** must be read in from the top of the design.

For a complete guide to using SDC with Synopsys applications, see the *Using the Synopsys Design Constraints Format Application Note* which is available through SolvNET at <http://solvnet.synopsys.com>.

The usage of some of the supported commands is restricted when reading SDC. In some cases, some options are not allowed. The **write_sdc** command supports the restricted usage by restricting what is written. The following restrictions apply for SDC Version 1.3:

- For **set_driving_cell**, the **-min** and **-max** options are not supported.
- For **set_port_fanout_number**, the **-min** and **-max** options are not supported.

When the hierarchy has been partially flattened, embedded hierarchy separators can make names ambiguous. It is not clear which hierarchy separator characters are part of the name, and which are real separators. Beginning with SDC Version 1.2, hierarchical names can be made non-ambiguous using the **set_hierarchy_separator** SDC command and/or the **-hsc** option available on the **get_cells**, **get_lib_cells**, **get_lib_pins**, **get_nets**, and **get_pins** SDC object access commands. By default, PrimeTime and Design Compiler write an SDC file using these features to create non-ambiguous names. It is considered best practice to write SDC files that contain

names that are not ambiguous. However, if you are using a third-party application that does not fully support SDC 1.2 or later (that is, it does not support the non-ambiguous hierarchical names features of SDC), then you can suppress these features by setting **sdc_write_unambiguous_names** variable to **true**.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following command writes the SDC script to the file named *top.sdc*:

```
prompt> write_sdc top.sdc  
1
```

SEE ALSO

```
read_sdc(2)  
write_script(2)  
sdc_write_unambiguous_names(3)
```

write_sdf

Writes a Standard Delay Format (SDF) back-annotation file.

SYNTAX

```
string write_sdf  
[-version sdf_version] [-significant_digits digits]  
[-instance inst_name]  
file_name
```

Data Types

<i>sdf_version</i>	string
<i>inst_name</i>	string
<i>file_name</i>	string

ARGUMENTS

-version *sdf_version*
Selects which SDF version to use. Supported SDF versions are 1.0 or 2.1. SDF 2.1 is the default.

-significant_digits *digits*
Specifies the number of digits to the right of the decimal point that are reported. Allowed values are from 0 through 13; the default is 3. Using this option overrides the value set by the variable **report_default_significant_digits**.

-instance *inst_name*
Specifies that the SDF is to be written only for the instance named *inst_name*.

file_name
Specifies the name of the SDF file to write.

DESCRIPTION

Writes leaf cell pin-to-pin timing information to a disk file. Timing information is written in SDF format using version v1.0 or v2.1. The timing file contains data associated with the netlist from which it is created.

Use **write_sdf** only when the instance names in the design agree with the naming conventions of the system to which the timing file is written.

Timing information can be written in multiples of ns, ps, and us. The time unit in the SDF file is the time unit specified in the technology library and is written in the timing file under 'timescale'. If there is no time unit in the library, the unit is assumed to be a multiple of nanoseconds.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example writes timing information for the design **MULT16** to a disk file called mult16.sdf, using SDF version 2.1.

```
prompt> write_sdf -version 2.1 mult16.sdf
```

SEE ALSO

```
read_sdf(2)  
set_annotated_check(2)  
set_annotated_delay(2)  
remove_annotated_check(2)  
remove_annotated_delay(2)  
report_annotated_check(2)  
report_annotated_delay(2)  
report_default_significant_digits(3)
```

write_stream

Writes a design (library) into a GDSII or Oasis file.

SYNTAX

```
status write_stream  
[-lib_name lib_name]  
[-format gds | oasis]  
[-cells list_of_cells]  
[-cell_file cell_name_file]  
stream_file_name
```

ARGUMENTS

-lib_name *lib_name*
Specifies the library name which contains the layout cells. If not specified, the current opened library is used.

-format gds | oasis
Format of output file. The value must be gds or oasis. Default format is gds.

-cells *list_of_cells*
Specifies the cell names to be output. If not specified, write_stream outputs all the cells in the specified library.

-cell_file *cell_name_file*
Specifies the name of the file that contains the cell names to be output. This option is for GDS only.

stream_file_name
Specifies the file name where to write library data. This is a required argument.

DESCRIPTION

Command **write_stream** writes the data in specified library to a file in GDS or OASIS format. The options are set by **set_write_stream_options** and reported by **report_write_stream_options**.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example writes out the existing library "design" into a file "out.gds".

```
prompt> write_stream -lib_name design -format gds out.gds  
or
```

```
prompt> write_stream -lib_name design out.gds
```

SEE ALSO

```
set_write_stream_options(2)  
report_write_stream_options(2)
```

write_verilog

Outputs a hierarchical Verilog file for the current design.

SYNTAX

```
status write_verilog
[-pg]
[-split_bus]
[-empty_module]
[-no_physical_only_cells]
[-no_pg_pin_only_cells]
[-no_corner_pad_cells]
[-no_pad_filler_cells]
[-no_core_filler_cells]
[-no_flip_chip_bump_cells]
[-no_cover_cells]
[-no_chip_cells]
[-no_io_pad_cells]
[-no_tap_cells]
[-no_unconnected_cells]
[-unconnected_ports]
[-keep_backslash_before_hiersep]
[-diode_ports]
[-wire_declaration]
[-output_net_name_for_tie]
[-macro_definition]
[-force_output_references reference_names]
[-force_no_output_references reference_names]
[-supply_statement [all | none]]
verilog_file_name
```

Data Types

<i>reference_names</i>	string
<i>verilog_file_name</i>	string

ARGUMENTS

-pg

Output power and ground nets and ports for all cell instances and module instances.

By default, the power and ground ports are ignored.

-split_bus

Outputs bus ports as individual bits. The port names are treated as scalar port names and are escaped.

For example, if you have the following module definition,

```
module top (out, in);
    output [3:0] out;
    input in;
    ...
endmodule
```



```

It is output as
module top(\out[3] , \out[2] , \out[1] , \out[0] , in);
    output \out[3] ;
    output \out[2] ;
    output \out[1] ;
    input \out[0] ;
    output in;
    ...
endmodule

```

-empty_module

Output empty module definitions for leaf-level cells, such as standard cells and macro cells. These module definitions contain port definitions but no internal module instances.

By default, no module definitions are written for leaf-level cells.

-no_physical_only_cells

Do not output physical-only cell instances. When you specify this option, the behavior is the same as when all the following options are specified: -

no_pg_pin_only_cells -no_core_filler_cells

-no_pg_pin_only_cells

Do not output cell instances that do not fall in the other five subtypes of physical-only cells and have only power or ground pins.

-no_corner_pad_cells

Do not output corner pad cell instances.

-no_pad_filler_cells

Do not output pad filler cell instances.

-no_core_filler_cells

Do not output core filler cell instances.

-no_flip_chip_bump_cells

Do not output flip-chip bump cell instances.

-no_cover_cells

Do not output cover cell instances.

-no_chip_cells

Do not output chip cell instances.

-no_io_pad_cells

Do not output I/O pad cell instances.

-no_tap_cells

Do not output tap cell instances.

-no_unconnected_cells

Do not output unconnected cell instances. Unconnected cell instances are instances with no ports connected except for power and ground ports.

-unconnected_ports

Output unconnected ports on module instances, macro instances, and standard

cell instances. All unconnected ports are output with a "SYNOPSIS_UNCONNECTED_%d" net. The "%d" portion of the name represents the unconnected net number, which is incrementally increased every time a net is generated for uniqueness.
By default, the unconnected ports are ignored.

-keep_backslash_before_hiersep

Keep the backslash character preceding a hierarchy separator for all identified instances.
By default, the backslash character is removed.

-diode_ports

Output diode ports for cell instances.
By default, the diode ports are ignored.

-wire_declaration

Output wire declarations in the hierarchical Verilog description. This option controls only the scalar wires. For bus wires, a declaration with bus limits is output regardless of this option.
By default, the command outputs wire declarations only for buses.
For example, if you use **-wire_declaration**, the wires are written as
wire n246;
wire [1:0] n245;
BUF U12 (.A(n245[0]), .Y(n246));
If you do not use **-wire_declaration**, the wires are written as
wire [1:0] n245;
BUF U12 (.A(n245[0]), .Y(n246));

-output_net_name_for_tie

Output an indirect tie net as its net name and a direct tie net that has a name of SNPS_LOGIC1 or SNPS_LOGIC0 as 1'b1 or 1'b0, respectively.
By default, both direct and indirect tie nets are written out as 1'b1 for logic 1 or 1'b0 for logic 0.

-macro_definition

Output module definitions for soft macros. A soft macro is a user-partitioned macro that is generated during design planning. Most big designs tend to be divided into several partitioned macros. When you use this option, the Verilog writer looks inside the soft macro and outputs the internal details. The full contents of CELL are also output.
By default, a soft macro is treated as a leaf macro and its definition is not output.

-force_output_references *reference_names*

Specifies the reference cell names, separated with a space, for which all cell instances must be output, overriding the following options:

-no_physical_only_cells
-no_pg_pin_only_cells
-no_corner_pad_cells
-no_pad_filler_cells
-no_core_filler_cells
-no_tap_cells
-no_cover_cells
-no_chip_cells
-no_flip_chip_bump_cells

-no_io_pad_cells

-no_unconnected_cells

The maximum allowed string size is 1023 chars.

-force_no_output_references *reference_names*

Specifies the reference cell names, separated with a space, for which all cell instances must not be output, overriding the same options as listed for

-force_output_references.

For example, consider a cell "top" with four types of core filler cells: FILL1, FILL2, FILL4, and FILL8. To generate the Verilog description with only FILL4 and FILL8 instances, use the **-no_core_filler_cells** option and specify **-force_output_references FILL4 FILL8**.

-supply_statement [[all | none]

By default, without the **-supply_statement** option, **write_verilog** skips generation of supply statements for PG nets connected with a power switch. In this case, **change_names** is required before using **write_verilog**.

With the **-supply_statement all**, option, **write_verilog** generates supply statements for all PG nets, regardless of the power switch output port connections.

With the **-supply_statement none**, option, **write_verilog** skips generation of supply statements for all PG nets in the output Verilog file.

DESCRIPTION

This command generates a Verilog netlist for the current Milkyway design. By specifying different options, you can control the behavior of this command.

Before running the **write_verilog** command, you should always run the **change_names -rule verilog -hierarchy** command. This command ensures correct Verilog naming conventions and also identifies the power switches present in the design.

If you run the **change_names** command before running **write_verilog**, the power switches are not output in the Verilog netlist. Otherwise, power switches are output by using supply1 statements.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows the most common use of the **write_verilog** command.

```
prompt> write_verilog top_hvo.v
```

```
Generating description for top level cell.
```

```
Processing module sub
```

```
Processing module top
```

```
Elapsed = 0:00:00, CPU = 0:00:00
```

```
Write verilog completed successfully.
```

SEE ALSO

`change_names(2)`
`read_verilog(2)`