

Synthesis 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

acs_check_directories	1
acs_compile_design	3
acs_customize_directory_structure	5
acs_get_parent_partition	6
acs_get_path	7
acs_merge_design	10
acs_read_hdl	14
acs_recompile_design	19
acs_refine_design	22
acs_remove_dont_touch	24
acs_report_attribute	25
acs_report_directories	26
acs_report_user_messages	28
acs_reset_directory_structure	30
acs_set_attribute	31
acs_submit	33
acs_submit_large	35
acs_write_html	37
add_module	38
add_pg_pin_to_db	40
add_pg_pin_to_lib	43
add_port_state	45
add_pst_state	47
add_to_collection	49

add_to_rp_group	52
alias	56
alib_analyze_libs	58
all_active_scenarios	59
all_clock_gates	60
all_clocks	62
all_connected	63
all_critical_cells	65
all_critical_pins	67
all_designs	69
all_dont_touch	70
all_drc_violated_nets	72
all_fanin	74
all_fanout	76
all_high_fanout	78
all_ideal_nets	80
all_inputs	82
all_operand_isolators	84
all_outputs	85
all_registers	87
all_rp_groups	90
all_rp_hierarchicals	92
all_rp_inclusions	94
all_rp_instantiations	96
all_rp_references	98
all_scenarios	100
all_threestate	101
all_tieoff_cells	103
analyze	105
append_to_collection	107
apply_clock_gate_latency	109
apropos	111
balance_buffer	113
balance_registers	115
cd	118
cell_of	120
change_link	122
change_macro_view	125

change_names	127
change_selection	133
characterize	136
check_bindings	141
check_bsd	143
check_budget	146
check_ccs_lib	149
check_design	152
check_error	155
check_implementations	157
check_isolation_cellsfp	159
check_level_shifters	161
check_library	163
check_license	169
check_mv_design	170
check_rp_groups	176
check_scan_def	178
check_synlib	180
check_target_library_subset	181
check_timing	182
check_tlu_plus_files	187
clean_buffer_tree	189
close_mw_lib	191
Collections_and_Querying	192
compare_collections	197
compare_delay_calculation	199
compare_interface_timing	201
compare_lib	205
compile	207
compile_partitions	215
compile_ultra	216
connect_logic_one	220
connect_logic_zero	222
connect_net	224
connect_pin	226
connect_power_domain	228
connect_power_net_info	230
connect_supply_net	232

context_check.....	234
copy_collection.....	236
copy_design.....	238
copy_mw_lib.....	241
cputime.....	242
create_bounds.....	244
create_bsd_patterns.....	247
create_bus.....	250
create_cache.....	253
create_cell.....	255
create_clock.....	257
create_command_group.....	260
create_design.....	261
create_die_area.....	263
create_generated_clock.....	265
create_ilm.....	269
create_multibit.....	276
create_mw_lib.....	281
create_net.....	283
create_net_shape.....	285
create_operating_conditions.....	289
create_pass_directories.....	292
create_placement_blockage.....	293
create_port.....	295
create_power_domain.....	297
create_power_net_info.....	300
create_power_switch.....	302
create_pst.....	305
create_rp_group.....	307
create_scenario.....	310
create_site_row.....	311
create_supply_net.....	313
create_supply_port.....	315
create_test_protocol.....	317
create_voltage_area.....	319
create_wiring_keepouts.....	322
current_design.....	323
current_design_name.....	325

current_dft_partition	326
current_instance	328
current_mw_lib	332
current_scenario	333
current_test_mode	334
date	335
dc_allocate_budgets	336
define_design_lib	340
define_dft_design	341
define_dft_partition	345
define_name_rules	347
define_proc_attributes	361
define_scaling_lib_group	365
define_test_mode	367
define_user_attribute	371
delete_operating_conditions	373
derive_constraints	374
dft_drc	376
disconnect_net	380
disconnect_power_net_info	382
distance	383
drive_of	385
echo	387
elaborate	388
encrypt_lib	393
error_info	394
exit	395
extract_physical_constraints	396
extract_rc	399
filter	401
filter_collection	404
find	407
foreach_in_collection	414
format_lib	416
get_alternative_lib_cells	420
get_always_on_logic	423
get_attribute	424
get_buffers	426

get_cells	429
get_clocks	433
get_clusters	435
get_command_option_values	438
get_cts_scenario	440
get_design_lib_path	441
get_designs	442
get_generated_clocks	445
get_ilm_objects	447
get_ilms	449
get_lib_attribute	451
get_lib_cells	453
get_lib_pins	456
get_libs	459
get_license	462
get_message_info	463
get_multibits	465
get_nets	467
get_object_name	471
get_path_groups	473
get_physical_hierarchy	475
get_pins	476
get_ports	480
get_power_domains	483
get_power_switches	486
get_references	488
get_rp_groups	490
get_scan_cells_of_chain	493
get_scan_chains	494
get_selection	495
get_supply_nets	498
get_supply_ports	500
get_timing_paths	502
get_zero_interconnect_delay_mode	509
getenv	510
group	512
group_path	517
group_variable	521

help.....	523
history.....	525
hookup_power_gating_ports	528
hookup_retention_register	530
identify_clock_gating	531
index_collection	534
infer_power_domains.....	536
insert_buffer.....	538
insert_clock_gating.....	544
insert_dft.....	547
insert_isolation_cell	555
insert_level_shifters	558
is_false.....	560
is_true.....	562
lib2saif	564
license_users	566
link	568
list_attributes	573
list_designs.....	576
list_duplicate_designs	578
list_files.....	579
list_instances	580
list_libs	583
list_licenses	585
list_test_models	586
list_test_modes.....	587
lminus.....	588
load_of	590
load_upf	591
man.....	596
map_isolation_cell	597
map_level_shifter_cell	599
map_power_switch.....	601
map_retention_cell.....	603
mem	605
merge_saif	607
mw_cel_collection	611
name_format	613

open_mw_lib	615
optimize_registers	617
pipeline_design	623
preview_dft	629
print_message_info	637
print_proc_new_vars	639
print_suppressed_messages	641
print_variable_group	642
printenv	644
printvar	645
proc_args	647
proc_body	648
propagate_annotated_delay_up	649
propagate_constraints	650
propagate_placement	656
propagate_switching_activity	658
push_down_model	661
pwd	662
query_objects	663
quit	666
read_aocvm	667
read_bsdl	669
read_db	670
read_ddc	671
read_file	672
read_lib	677
read_parasitics	681
read_partition	684
read_pin_map	685
read_saif	687
read_sdc	692
read_sdf	697
read_sverilog	700
read_test_model	701
read_test_protocol	703
read_verilog	706
read_vhdl	707
rebuild_mw_lib	708

redirect	710
remove_annotated_check	714
remove_annotated_delay	716
remove_annotated_transition	719
remove_annotations	720
remove_aocvm	722
remove_attribute	724
remove_boundary_cell	726
remove_boundary_cell_io	728
remove_bounds	730
remove_bsd_compliance	732
remove_bsd_instruction	734
remove_bsd_linkage_port	735
remove_bsd_power_up_reset	736
remove_buffer	737
remove_bus	740
remove_cache	742
remove_case_analysis	745
remove_cell	747
remove_cell_degradation	749
remove_clock	750
remove_clock_gating	751
remove_clock_gating_check	754
remove_clock_groups	756
remove_clock_latency	758
remove_clock_sense	760
remove_clock_transition	762
remove_clock_uncertainty	763
remove_congestion_options	766
remove_constraint	767
remove_cts_scenario	769
remove_data_check	770
remove_design	772
remove_dft_connect	774
remove_dft_design	776
remove_dft_equivalent_signals	778
remove_dft_location	779
remove_dft_partition	780

remove_dft_signal	781
remove_disable_clock_gating_check.	783
remove_disable_timing.	784
remove_dp_int_round.	786
remove_driving_cell	787
remove_from_collection	789
remove_from_rp_group	791
remove_generated_clock.	793
remove_host_options.	795
remove_ideal_latency.	796
remove_ideal_net.	798
remove_ideal_network	800
remove_ideal_transition	802
remove_ignored_layers	804
remove_input_delay	805
remove_isolate_ports	808
remove_isolation_cell.	809
remove_level_shifters.	810
remove_license	811
remove_multibit	812
remove_net.	816
remove_operand_isolation	818
remove_output_delay	820
remove_pass_directories	822
remove_pin_map	823
remove_pin_name_synonym	824
remove_port	827
remove_power_domain	829
remove_power_net_info.	831
remove_preferred_routing_direction	832
remove_propagated_clock.	833
remove_rp_group_options	835
remove_rp_groups	837
remove_rtl_load	839
remove_scaling_lib_group	841
remove_scan_group.	842
remove_scan_link.	843
remove_scan_path.	845

remove_scan_register_type	847
remove_scan_replacement	848
remove_scan_suppress_toggling	849
remove_scenario	850
remove_sdc	851
remove_target_library_subset	852
remove_test_mode	853
remove_test_model	854
remove_test_point_element	855
remove_test_protocol	856
remove_unconnected_ports	857
remove_upf	859
remove_user_attribute	860
remove_verification_priority	862
remove_voltage_area	863
remove_wire_load_min_block_size	864
remove_wire_load_model	865
remove_wire_load_selection_group	867
rename	869
rename_design	870
rename_mw_lib	873
replace_clock_gates	874
replace_synthetic	876
report_ahfs_options	877
report_annotated_check	878
report_annotated_delay	880
report_annotated_transition	882
report_aocvm	883
report_area	887
report_attribute	890
report_auto_ungroup	895
report_autofix_configuration	897
report_autofix_element	899
report_boundary_cell	901
report_boundary_cell_io	903
report_bounds	905
report_bsd_compliance	907
report_bsd_instruction	908

report_bsd_linkage_port.	911
report_bsd_power_up_reset.	912
report_buffer_tree.	913
report_buffer_tree_qor	915
report_bus	917
report_cache.	919
report_case_analysis	924
report_cell.	926
report_check_library_options	930
report_clock	932
report_clock_gating	935
report_clock_gating_check.	944
report_clock_timing	947
report_clock_tree	959
report_compile_options	965
report_congestion.	968
report_congestion_options	970
report_constraint.	972
report_crpr	980
report_delay_calculation.	985
report_delay_estimation_options	990
report_design	992
report_design_lib	995
report_dft_configuration	997
report_dft_connect	999
report_dft_design	1000
report_dft_drc_rules	1001
report_dft_equivalent_signals.	1003
report_dft_insertion_configuration	1004
report_dft_location	1005
report_dft_partition	1006
report_dft_signal.	1007
report_direct_power_rail_tie	1010
report_disable_timing	1011
report_dp_smartgen_options	1013
report_extraction_options.	1014
report_fsm	1015
report_hierarchy	1017

report_host_options	1019
report_ideal_network	1020
report_ignored_layers.	1024
report_ilm	1025
report_interclock_relation	1029
report_internal_loads	1031
report_isolate_ports	1033
report_isolation_cell	1035
report_level_shifter	1038
report_lib.	1042
report_logicbist_configuration.	1057
report_mode	1059
report_multibit.	1061
report_mw_lib.	1065
report_name_rules	1067
report_names	1070
report_net	1072
report_net_changes	1077
report_net_fanout	1079
report_operand_isolation	1082
report_operating_conditions	1087
report_partitions	1089
report_pass_data	1091
report_path_budget	1093
report_path_group	1098
report_physical_constraints	1100
report_pin_map	1103
report_pin_name_synonym	1105
report_port	1107
report_power	1113
report_power_calculation	1122
report_power_domain.	1128
report_power_gating.	1131
report_power_net_info	1133
report_power_pin_info	1135
report_power_switch	1140
report_preferred_routing_direction.	1142
report_pst	1143

report_qor	1145
report_reference	1149
report_resources	1152
report_retention_cell	1156
report_rp_group_options	1159
report_saif	1161
report_scan_chain	1165
report_scan_compression_configuration	1167
report_scan_configuration	1169
report_scan_group	1172
report_scan_link	1173
report_scan_path	1175
report_scan_register_type	1178
report_scan_replacement	1179
report_scan_state	1181
report_scan_suppress_toggling	1182
report_scenario_options	1183
report_scenarios	1185
report_supply_net	1187
report_supply_port	1189
report_synlib	1191
report_target_library_subset	1195
report_test_assume	1196
report_test_model	1197
report_test_point_element	1199
report_testability_configuration	1201
report_threshold_voltage_group	1203
report_timing	1205
report_timing_derate	1219
report_timing_requirements	1221
report_tlu_plus_files	1225
report_transitive_fanin	1226
report_transitive_fanout	1228
report_units	1231
report_use_test_model	1232
report_voltage_area	1233
report_wire_load	1235
report_wrapper_configuration	1238

reset_autofix_configuration	1240
reset_autofix_element	1241
reset_bsd_configuration	1242
reset_clock_gate_latency	1243
reset_design	1244
reset_dft_configuration	1247
reset_dft_drc_rules	1248
reset_dft_insertion_configuration	1250
reset_logicbist_configuration	1251
reset_mode	1252
reset_path	1254
reset_physical_constraints	1258
reset_pipeline_scan_data_configuration	1259
reset_scan_compression_configuration	1260
reset_scan_configuration	1261
reset_switching_activity	1262
reset_test_mode	1264
reset_testability_configuration	1265
reset_timing_derate	1266
reset_wrapper_configuration	1267
rewire_clock_gating	1269
rp_group_inclusions	1272
rp_group_instantiations	1274
rp_group_references	1276
rtl2saif	1278
saif_map	1280
save_upf	1287
set_active_scenarios	1288
set_ahfs_options	1290
set_always_on_strategy	1295
set_annotated_check	1297
set_annotated_delay	1300
set_annotated_transition	1303
set_aocvm_coefficient	1305
set_aspect_ratio	1307
set_attribute	1308
set_auto_disable_drc_nets	1310
set_autofix_configuration	1313

set_autofix_element	1316
set_balance_registers	1319
set_boundary_cell	1321
set_boundary_cell_io	1327
set_boundary_optimization.	1329
set_bsd_ac_port.	1331
set_bsd_compliance.	1332
set_bsd_configuration	1334
set_bsd_instruction.	1337
set_bsd_linkage_port	1343
set_bsd_power_up_reset.	1345
set_case_analysis	1347
set_cell_degradation	1349
set_cell_internal_power	1351
set_cell_location.	1353
set_check_library_options	1355
set_clock_gate_latency	1366
set_clock_gating_check	1369
set_clock_gating_registers.	1372
set_clock_gating_style	1374
set_clock_groups	1385
set_clock_latency	1388
set_clock_sense.	1391
set_clock_transition	1393
set_clock_uncertainty.	1395
set_combinational_type	1399
set_compile_directives	1401
set_compile_partitions	1404
set_congestion_options	1407
set_connection_class.	1409
set_context_margin	1411
set_cost_priority	1413
set_critical_range	1415
set_cts_scenario.	1417
set_current_command_mode.	1419
set_data_check	1421
set_datapath_optimization_effort	1424
set_default_drive	1426

set_default_driving_cell	1428
set_default_fanout_load	1431
set_default_input_delay	1433
set_default_load	1434
set_default_output_delay	1436
set_delay_calculation	1437
set_delay_estimation_options	1439
set_design_license	1442
set_design_top	1444
set_dft_clock_controller	1445
set_dft_configuration	1447
set_dft_connect	1450
set_dft_drc_configuration	1454
set_dft_drc_rules	1456
set_dft_equivalent_signals	1458
set_dft_insertion_configuration	1459
set_dft_location	1461
set_dft_signal	1463
set_direct_power_rail_tie	1470
set_disable_clock_gating_check	1471
set_disable_timing	1473
set_domain_supply_net	1475
set_dont_retime	1477
set_dont_touch	1479
set_dont_touch_network	1482
set_dont_use	1484
set_dp_int_round	1486
set_dp_smartgen_options	1489
set_drive	1494
set_driving_cell	1496
set_equal	1501
set_extraction_options	1502
set_false_path	1504
set_fanout_load	1510
set_fix_hold	1512
set_fix_multiple_port_nets	1514
set_flatten	1516
set_fsm_encoding	1519

set_fsm_encoding_style.....	1521
set_fsm_minimize.....	1523
set_fsm_order.....	1524
set_fsm_preserve_state.....	1525
set_fsm_state_vector.....	1526
set_fuzzy_query_options.....	1527
set_host_options.....	1532
set_ideal_latency.....	1533
set_ideal_net.....	1535
set_ideal_network.....	1537
set_ideal_transition.....	1540
set_ignored_layers.....	1542
set_impl_priority.....	1544
set_implementation.....	1546
set_input_delay.....	1548
set_input_transition.....	1552
set_isolate_ports.....	1554
set_isolation.....	1556
set_isolation_control.....	1558
set_leakage_power_model.....	1560
set_level_shifter.....	1562
set_level_shifter_strategy.....	1565
set_level_shifter_threshold.....	1567
set_lib_attribute.....	1569
set_libcell_dimensions.....	1571
set_libpin_location.....	1572
set_load.....	1574
set_local_link_library.....	1578
set_logic_dc.....	1580
set_logic_one.....	1582
set_logic_zero.....	1584
set_logicbist_configuration.....	1586
set_map_only.....	1588
set_max_area.....	1590
set_max_capacitance.....	1592
set_max_delay.....	1594
set_max_dynamic_power.....	1600
set_max_fanout.....	1602

set_max_leakage_power	1604
set_max_lvth_percentage	1606
set_max_net_length	1608
set_max_time_borrow	1610
set_max_total_power	1612
set_max_transition	1614
set_message_info	1616
set_message_severity	1618
set_min_capacitance	1619
set_min_delay	1621
set_min_library	1626
set_minimize_tree_delay	1628
set_mode	1630
set_model_drive	1631
set_model_load	1633
set_model_map_effort	1635
set_multibit_options	1636
set_multicycle_path	1638
set_mw_lib_reference	1645
set_mw_technology_file	1647
set_operand_isolation_cell	1649
set_operand_isolation_scope	1651
set_operand_isolation_slack	1653
set_operand_isolation_style	1655
set_operating_conditions	1657
set_opposite	1661
set_optimize_registers	1662
set_output_clock_port_type	1667
set_output_delay	1669
set_physical_hierarchy	1674
set_pin_name_synonym	1675
set_pipeline_scan_data_configuration	1677
set_placement_area	1678
set_port_fanout_number	1679
set_port_location	1681
set_port_side	1683
set_power_gating_signal	1685
set_power_gating_style	1687

set_power_prediction	1690
set_prefer	1692
set_preferred_routing_direction	1694
set_preferred_scenario.	1695
set_propagated_clock	1696
set_pulse_clock_cell.	1698
set_register_merging	1700
set_register_replication	1702
set_register_type	1703
set_related_supply_net	1706
set_relative_always_on	1708
set_replace_clock_gates	1710
set_resistance	1713
set_resource_allocation	1715
set_retention.	1717
set_retention_control	1719
set_retention_control_pins	1721
set_rp_group_options.	1723
set_rtl_load.	1726
set_scaling_lib_group.	1732
set_scan_compression_configuration	1734
set_scan_configuration.	1737
set_scan_element	1742
set_scan_group	1745
set_scan_link	1749
set_scan_path	1751
set_scan_register_type	1760
set_scan_replacement.	1763
set_scan_state	1765
set_scan_suppress_toggling	1766
set_scenario_options	1769
set_scope.	1772
set_size_only	1774
set_structure.	1776
set_svf	1778
set_switching_activity.	1780
set_synlib_dont_get_license	1785
set_tap_elements.	1787

set_target_library_subset	1788
set_test_assume	1790
set_test_point_element	1792
set_testability_configuration	1795
set_timing_derate	1797
set_timing_ranges	1800
set_tlu_plus_files	1803
set_transform_for_retiming	1806
set_true_delay_case_analysis	1808
set_unconnected	1810
set_ungroup	1812
set_units	1814
set_user_attribute	1816
set_user_budget	1818
set_utilization	1820
set_verification_priority	1822
set_voltage	1824
set_vsd	1826
set_wire_load_min_block_size	1828
set_wire_load_mode	1830
set_wire_load_model	1832
set_wire_load_selection_group	1837
set_wrapper_configuration	1839
set_zero_interconnect_delay_mode	1844
setenv	1845
shell_is_in_topographical_mode	1847
shell_is_in_upf_mode	1848
shell_is_in_xg_mode	1849
simplify_constants	1850
size_cell	1851
sizeof_collection	1853
sort_collection	1855
source	1857
sub_designs_of	1859
sub_instances_of	1861
suppress_message	1863
syntax_check	1864
translate	1866

unalias	1868
ungroup	1869
uniquify	1873
unsuppress_message	1876
update_bounds	1877
update_lib	1879
update_timing	1882
use_test_model	1883
which	1884
write	1886
write_bsdlib	1890
write_compile_script	1892
write_design_lib_paths	1895
write_environment	1897
write_file	1899
write_interface_timing	1903
write_lib	1905
write_link_library	1909
write_makefile	1911
write_milkyway	1913
write_mw_lib_files	1916
write_parasitics	1918
write_partition	1921
write_partition_constraints	1923
write_physical_constraints	1925
write_rp_groups	1928
write_rtl_load	1931
write_saif	1932
write_scan_def	1935
write_script	1938
write_sdc	1942
write_sdf	1946
write_test	1948
write_test_model	1950
write_test_protocol	1952

acs_check_directories

Checks the Automated Chip Synthesis (ACS) directory structure settings for correctness.

SYNTAX

```
status acs_check_directories  
[-verbose verbosity_level]
```

Data Types

verbosity_level integer

ARGUMENTS

-verbose *verbosity_level*
Prints out more messages. Valid values for the *verbosity_level* are **1** to produce verbose output, or **0** for the standard (minimal) output.

DESCRIPTION

The **acs_check_directories** command checks the template that is used to create the Automated Chip Synthesis directory structure, and reports errors, if any are found. This command checks the three variables that comprise the directory structure template: **list_of_file_types**, **pass_dependent_file_types**, and **acs_dir**, and searches for the following:

1. Is every file type defined with the proper format in **list_of_file_types**? (The proper format is *file_type(suffix)*, where *suffix* is optional.)
2. Do any two file types have the same name?
3. Are all items in **pass_dependent_file_types** shown in **list_of_file_types**?
4. Are all file types assigned to one path?
5. Is any file type shown in a path not listed in **list_of_file_types**?
6. Is any file type assigned to more than one path?
7. Are any non pass-dependent file types going into a pass-dependent path (a path that contains the pass variable)?

EXAMPLES

The command returns 1 for a correct directory structure setting; otherwise an error message is issued and the command returns 0.

```
prompt> acs_check_directories  
1
```

The following example does not bind the *pass_cstr* file type to any path. This causes problems later when Automated Chip Synthesis attempts to generate constraint files.

```
prompt> acs_check_directories  
Error: file type "pass_cstr" is not assigned to any path.  
0
```

SEE ALSO

```
acs_get_path(2)  
acs_report_directories(2)
```

acs_compile_design

Compiles the constrained RTL to a netlist using constraints propagated from the top-level design.

SYNTAX

```
status acs_compile_design  
[-destination pass_name]  
[-prepare_only]  
[-force]  
[-update]  
[-update_source source_pass]  
design_name
```

Data Types

<i>pass_name</i>	string
<i>source_pass</i>	string
<i>design_name</i>	string

ARGUMENTS

- destination *pass_name***
Specifies the destination pass directory name for this run. A directory structure for the destination pass is automatically created with the name *pass_name*. All of the generated .ddc files, constraints, scripts, and reports are put under this pass directory.
If you do not specify this option, Automated Chip Synthesis assumes that the pass directory name is @default_pass_name@default_pass_number, where the default pass name is defined by the **acs_default_pass_name** variable, and the default pass number is either 0 (compile_design), or 1 (refine_design).
- prepare_only**
Specifies that only the preparation steps like constraint generation, script generation, creation of partition .ddc files, and makefile generation are performed. The actual distributed compile is not done.
- force**
Forces the preparation steps such as constraint generation, script generation, creation of partition .ddc files, and makefile generation. By default, if a makefile already exists, the preparation steps are skipped.
- update**
Performs an incremental design update by recompiling any compile partitions that have been updated by a previous **acs_merge_design** command, and then performing a top-level boundary compile.
- update_source *source_pass***
Specifies the data directory that contains the merged design, which is the result of the the **acs_merge_design** command. The **acs_compile_design -update** command loads the designs from *source_pass*/db/pre_compile before performing

the incremental design update.
If this option is not specified, but the **-update** option is specified, Automated Chip Synthesis looks for the design in memory. If the design is not found in memory, Automated Chip Synthesis looks in the default directory, `${acs_default_pass_name}0_incr/db/pre_compile`.

`design_name`
Specifies the name of the top-level design.

DESCRIPTION

The **acs_compile_design** command performs an initial mapping from RTL to gates using parallel compile strategy. You must provide a constrained GTECH design in memory or as a .ddc file in the `${acs_work_dir}/elab/db` directory. By default, **acs_compile_design** partitions the design at the first level of the logical hierarchy if no user partitions are set. The top-level constraints are propagated to the partitions, the compile scripts are generated, and the makefile is written out. A parallel compile is performed using LSF or the gmake utility, according to the settings of the **acs_num_parallel_jobs** variable.

Before running this command, review the default settings in the `.synopsys_dc.setup` file located in your project directory.

This command is implemented as a Tcl procedure that is visible and customizable. For details about how to change the behavior of this procedure, see the *Automated Chip Synthesis User Guide*.

EXAMPLE

The following example shows how to perform an incremental design update. Assume that you have compiled your design using **acs_compile_design**. Later, you modified one subdesign named `desA` and created a new design database in `${acs_work_dir}/elab/db` with the changed subdesign `desA`. Assume that you are using the default directories. Then you can merge the changed design with an existing mapped design database of a previous compile run and recompile the changes by using the following commands:

```
prompt> acs_merge_design top_design_name \  
        -update desA -des merged_pass  
  
prompt> acs_compile_design top_design_name \  
        -update -update_source merged_pass
```

SEE ALSO

`acs_merge_design(2)`
`acs_recompile_design(2)`
`acs_refine_design(2)`

acs_customize_directory_structure

This command is obsolete. See the **acs_user_dir** variable for information about customizing the Automated Chip Synthesis directory structure.

acs_get_parent_partition

Creates a collection of designs that are compiled partitions containing the specified subdesign.

SYNTAX

```
string acs_get_parent_partition  
design_name  
[-hierarchy]  
[-list]
```

Data Types

design_name string

ARGUMENTS

design_name
 Specifies the name of a subdesign.

-hierarchy
 Causes the command to collect all parent partitions of the specified design. By default only the immediate parent partition is returned.

-list
 Causes the command to return a list of names. Without this option a collection of design objects is returned.

DESCRIPTION

The **acs_get_parent_partition** command returns the lowest-level compile partition that contains the specified subdesign. If the specified subdesign is a compile partition, the command returns the subdesign. If the **-hierarchy** option is specified, all partitions that contain the subdesign are included in the collection. If the subdesign is in the subtree of a multiply instantiated design, all parent partitions of all instances of this multiply instantiated design are included in the result. The command creates a collection containing the design objects, unless the **-list** option is specified. In this case, the command returns a Tcl list with the names of the designs.

SEE ALSO

set_compile_partitions(2)

acs_get_path

Gets the path location for the specified file. To specify a file, you specify its file type and for pass-dependent files, its pass directory.

SYNTAX

```
string acs_get_path  
-file_type filetype  
[-mode read | write]  
[-pass pass_name]  
[-name filename [-append]]  
[-relative]
```

Data Types

<i>filetype</i>	string
<i>pass_name</i>	string
<i>filename</i>	string

ARGUMENTS

-file_type *filetype*

Specifies the file type. The **acs_get_path** command returns the path location that contains the specified file. The file type must be one of the Automated Chip Synthesis file types. For information about file types, see the *Automated Chip Synthesis User Guide*.

-mode read | write

Specifies whether the command runs in read mode or write mode. In either mode, the command returns the directory name for the specified file. In write mode, in addition to returning the directory name, the **acs_get_path** command creates the directory if it does not exist.
By default, the command runs in read mode.

-pass *pass_name*

Specifies the pass name for pass-dependent files. In read mode only, you can request the generic path for a pass-dependent file type by specifying the pass variable (@pass@) for the *-pass_name* value.
The @ character cannot be used as a literal character in Automated Chip Synthesis. This character is reserved for specifying the names of pass-dependent directories.

-name *filename*

Appends the specified file name to the end of the returned path. You must specify the full name of the file if you don't specify the **-append** option. If the **-append** option is specified, only the base name of the file is needed. By default, only the directory is returned.

-append

Appends the pass name, file type, and the default file suffix to the base name of the file name specified by the **-name** option, if necessary to create unique file names for pass-dependent files located in pass-independent

directories.

By default only the specified file name (plus the default file suffix) is returned.

You must use this option with the **-name** option. If you specify **-append** and do not specify the **-name** option, an error occurs.

-relative

Returns the directory location relative to the Automated Chip Synthesis project directory (as specified by the **acs_work_dir** variable).

By default, the **acs_get_path** command returns the absolute path.

DESCRIPTION

The **acs_get_path** command locates the files used by Automated Chip Synthesis.

Because you can customize the Automated Chip Synthesis directory structure, Automated Chip Synthesis must provide a dynamic method for locating files. The **acs_get_path** command provides this capability. In general, you use this command when writing custom scripts for Automated Chip Synthesis, instead of hardcoding file locations. This ensures that your custom scripts work with both the default and customized directory structures.

EXAMPLES

The following examples are based on the default directory setting. The output may vary for non-default directory structures.

The following command returns the absolute path for the postcompile .ddc file in the PASSX directory tree:

```
prompt> acs_get_path -file_type post_ddc -pass PASSX  
/usr/my_design/PASSX/db/post_compile
```

The following command returns the path for the same file, relative to the project root directory, /usr/my_design:

```
prompt> acs_get_path -file_type post_ddc -pass PASSX -relative  
PASSX/db/post_compile
```

The following command returns the path for the same file, including the specified file name:

```
prompt> acs_get_path -file_type post_ddc -pass PASSX -name my_design  
/usr/my_design/PASSX/db/post_compile/my_design.ddc
```

The following command returns the generic path for the postcompile .ddc files:

```
prompt> acs_get_path -file_type post_ddc -pass @pass@  
/usr/my_design/${pass}/db/post_compile
```

acs_get_path

Assume that the directory structure is customized so that the precompile and postcompile .ddc files for each pass are placed in the \$acs_work_dir/db directory. In this case, Automated Chip Synthesis appends the file type and pass name to make each file name unique. The following command returns the unique file name used by Automated Chip Synthesis.

```
prompt> acs_get_path -file_type post_ddc -pass test \  
? -name my_design -append  
/usr/my_design/db/my_design.test.post_ddc.ddc
```

SEE ALSO

```
acs_report_directories(2)  
create_pass_directories(2)  
acs_work_dir(3)
```

acs_merge_design

Preprocesses a design for incremental design update by merging the modified designs and their parent compile partitions with the mapped design being updated.

SYNTAX

```
status acs_merge_design
[-auto_update | -update design_list]
[-unmapped source_dir]
[-mapped data_dir]
[-type pre | post]
[-destination dest_dir]
[-reference pre_compile_partition_dir]
design_name
```

Data Types

<i>design_list</i>	string
<i>source_dir</i>	string
<i>data_dir</i>	string
<i>dest_dir</i>	string
<i>pre_compile_partition_dir</i>	string
<i>design_name</i>	string

ARGUMENTS

- auto_update**
Automatically determines the designs to be updated using the information in timestamp tables written by the **acs_read_hdl -[auto_]update** command.
- update *design_list***
Specifies the designs to be updated in the design already mapped.
- unmapped *source_dir***
Specifies the directory with the .ddc files of the design database that is the source for the modified designs. From this design database, the **acs_merge_design** command fetches the modified designs specified with the **-update** option and all parent partitions of these designs upwards to the top-level hierarchy. These designs are usually RTL/unmapped (GTECH) designs, but they can also be mapped designs.
All .ddc files in this directory must comprise a complete design database for the top-level design. The designs may be arbitrarily distributed among the .ddc files.
If you do not specify this option, **acs_merge_design** picks up the .ddc files from the elab_ddc directory. The default location of this directory is \$acs_work_dir/elab/db.
- mapped *data_dir***
Specifies the data directory that contains the mapped .ddc files of the top-level design and its compile partitions. The directory must be an Automated Chip Synthesis data directory and the .ddc files located under the subdirectory must contain partition and constraints information.

acs_merge_design

If you do not specify the **-type** option, **acs_merge_design** reads the mapped design from the `post_ddc` directory. The default location of this directory is `$acs_work_dir/data_dir/db/post_compile`.

If you specify the **-type pre** option, the **acs_merge_design** command reads the mapped design from the `pre_ddc` directory. The default location of this directory is `$acs_work_dir/data_dir/db/pre_compile`.

If you do not specify this option or the **-type** option, **acs_merge_design** reads the mapped design from the `post_ddc` directory in the default data directory. The default data directory is `${default_pass_name}0`, where the default pass name is defined by the **acs_default_pass_name** variable.

If you do not specify this option, but you do specify the **-type pre** option, **acs_merge_design** reads the mapped design from the `pre_ddc` directory in the default data directory.

-type pre | post

Specifies whether the mapped design is read from the `pre_ddc` directory or the `post_ddc` directory of the mapped data directory. (See the description of the **-mapped** option for more information about the mapped data directory).

If you do not specify this option, the mapped design is read from the `post_ddc` directory.

-destination dest_dir

Specifies the name of the destination directory for this run. The **acs_merge_design** command automatically creates the directory structure for the destination directory. All of the generated `.ddc` files are put in the `pre_ddc` directory in the destination directory.

If you do not specify this option, **acs_merge_design** uses a default destination directory named `${default_pass_name}0_incr`, where the default pass name is defined by the **acs_default_pass_name** variable.

-reference pre_compile_partition_dir

Specifies the directory in which the `pre_compile` partitions are found. The earliest timestamp of the `precompile` partitions is used as the time at which the last compile command was issued. Any design with a later timestamp is selected for update when the **-auto_update** option is specified. The default location is `pass0/db/pre_compile`.

design_name

Specifies the name of the top-level design.

DESCRIPTION

The **acs_merge_design** command loads the mapped design from the mapped data directory (see the description of the **-mapped** option), overrides the updated designs and all its parent partitions up to the top level in the hierarchy with the GTECH implementations from the unmapped source directory. (See the description of the **-unmapped** option). For any updated design the parent partitions up to the top level are also replaced in order to ensure correct design functionality, since these partitions may have been optimized using design properties of the old version of the updated designs, such as logical constants. Finally, the merged design is saved in the destination directory. (See the description of the **-destination** option).

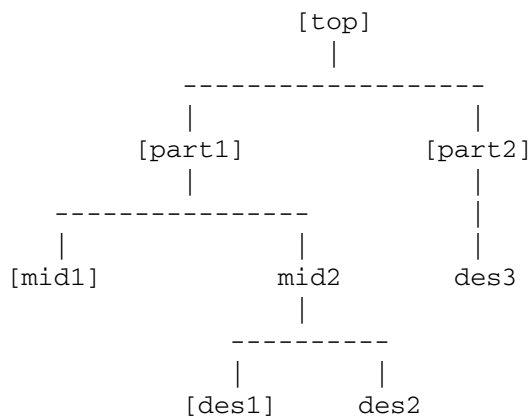
The **acs_merge_design** command marks all partitions that are replaced as being updated, so that a subsequent Automated Chip Synthesis **compile** command automatically

detects the updated designs for recompile when using the **-update** option.

The **-auto_update** option can be used to automatically determine the designs to replace. This is done by comparing the timestamp of the designs in the unmapped design (as recorded in the timestamp tables created by the **acs_read_hdl -[auto_]update** command) and the timestamp of the precompile partitions. If a design has a later timestamp than the earliest timestamp of the precompile partition, then it is placed in the list of updated designs.

EXAMPLES

The following is an example of the **acs_merge_design** command. Assume you have the following design:



In this example, the designs with brackets ([]) are Automated Chip Synthesis compile partitions. The designs without brackets are non-partitions. The compile partition named part1 is the parent partition of des2.

When you perform an initial Automated Chip Synthesis compile, the tool takes the unmapped RTL design from the elab/db directory or from memory, partitions the design, places the GTECH .ddc files of the unmapped partitions in the \$acs_work_dir/pass0/db/pre_compile directory, and the .ddc files for the mapped design in the \$acs_work_dir/pass0/db/post_compile directory. In these directories, each partition in the design has a corresponding .ddc file whose name matches the partition's name.

Assume that you modify the des2 subdesign and put the new RTL design database with the changed design into the elab/db directory. Then you merge your changes in des2 into the design that is already mapped from the initial Automated Chip Synthesis compile, as shown in the following example:

```
prompt> acs_merge_design top -update des2 \
        -unmapped elab/db -mapped pass0
```

The **-unmapped** and **-mapped** options are not necessary, since **acs_merge_design** picks up the modified designs from elab/db and merges them with \$acs_work_dir/pass0/db/post_compile by default. The tool picks up the des2 design and its parent partitions named part1 and top containing designs des2, mid2, part1, and top from the directory specified by the **-unmapped** option (elab/db in this example) and merges them with the

design in the directory specified by the **-mapped** option (\$acs_work_dir/pass0/db/post_compile in this example). The result of this merge is placed by default in the \$acs_work_dir/pass0_incr/db/pre_compile directory.

SEE ALSO

acs_compile_design(2)
acs_read_hdl(2)
acs_recompile_design(2)
acs_refine_design(2)

acs_read_hdl

Reads in the HDL source code of a design and generates the GTECH representation.

SYNTAX

```
status acs_read_hdl
[design_name]
[-hdl_source file_or_dir_list]
[-exclude_list file_or_dir_list]
[-format {verilog | vhdl}]
[-recurse]
[-no_dependency_check]
[-no_elaborate]
[-library design_lib_name]
[-verbose]
[-auto_update | -update file_list]
[-destination destination_dir]
[-sv_package_files file_list]
```

Data Types

<i>design_name</i>	string
<i>file_or_dir_list</i>	list
<i>design_lib_name</i>	string
<i>file_list</i>	list
<i>destination_dir</i>	string

ARGUMENTS

design_name
Specifies the name of the top-level design. Elaboration starts at the specified module and elaborates the entire design subtree under that module. This argument is required unless the **-no_elaborate** option is specified.

-hdl_source *file_or_dir_list*
Specifies a list of files or directories containing the source code of the design. The **acs_read_hdl** command analyzes the contents of all files in the list (and in the subdirectories if the **-recurse** option is set) according to the **acs_verilog_extensions**, **acs_vhdl_extensions**, **acs_exclude_list**, and **acs_exclude_extensions** variables.
If this option is not specified, the command uses the **acs_hdl_source** Tcl variable by default. If the **acs_hdl_source** variable is not defined, the **search_path** variable is used. The command-line argument overrides the variables. The names in the **-hdl_source** lists can contain wildcards. The command performs a Tcl glob-style expansion on each list item.

-exclude_list *file_or_dir_list*
Specifies a list of files and directories that are not to be analyzed. If the command expands an item from the **-hdl_source** list, it checks the file or directory against all entries of this list and ignores the file or directory if it is covered by at least one of the entries.
If this option is omitted, the command uses the **acs_exclude_list** Tcl

acs_read_hdl

variable. The command-line argument overrides the variable. It is possible to apply Tcl's UNIX-like filename globbing features by using the *, ?, and [] in file and directory names in this list. Add a file name (wildcards are allowed) without a path to exclude all files with that name regardless of their location.

-format {verilog | vhdl}

Specifies that **acs_read_hdl** is to only look for files of the specified language with extensions from the corresponding extensions list for the **acs_verilog_extensions** or **acs_vhdl_extensions** variables. The default is that the command reads in all files it finds with Verilog extensions and VHDL extensions.

Use this option when you want to specify file names in the **-hdl_source** list that do not have one of the language-specific extensions.

-recurse

Specifies that the command is to look into subdirectories of directories specified in the **-hdl_source** list, and recursively collect source files from that location.

The default is that the command looks into the specified directories, but does not look in their subdirectories.

-no_dependency_check

Causes the command not to detect Verilog include files, not to determine VHDL libraries, and not to reorder the source file list. Instead, the command processes the HDL source file list from left to right and analyzes VHDL files into the current working library or the library specified with the **-library** option.

-no_elaborate

Prevents **acs_read_hdl** from performing the elaboration. The command stops after all source files are analyzed. Specify this option when you want to manually elaborate the design after the command finishes.

-library *design_lib_name*

Specifies the library that is used as the working library for all VHDL files.

-verbose

Prints out more messages.

-auto_update

Automatically determines the HDL files to update, puts them in order according to the dependency (unless the **-no_dependency_check** option is specified) and analyzes and elaborates the files. The designs defined in the specified HDL file(s) replace the corresponding designs in the unmapped design in the destination directory. The option also creates the tables of timestamps used by the subsequent **acs_read_hdl -auto_update**, **acs_read_hdl -update**, and **acs_merge_design -auto_update** commands.

This option is mutually exclusive with the **-update** option.

-update *file_list*

Analyzes and elaborates the specified HDL source files in the given order. The designs defined in the specified HDL file(s) replace the corresponding designs in the unmapped design in the destination directory. The option also creates the tables of timestamps used by the subsequent **acs_read_hdl -**

auto_update, **acs_read_hdl -update**, and **acs_merge_design -auto_update** commands.

This option is mutually exclusive with the **-auto_update** option.

-destination *destination_dir*

Specifies the directory to which the unmapped design is written when either the **-update** or the **-auto_update** option is specified. The default is that the command writes the unmapped design and the timestamp tables to the elab/db directory.

DESCRIPTION

The **acs_read_hdl** command reads in the HDL source files of a design, analyzes them, and elaborates the design starting at the specified top-level module. It retains the resulting GTECH representation in memory.

To locate the source files, the command expands each item from the **-hdl_source** list. The list is either specified as an option with **-hdl_source** at the command line, or set as the value of the **acs_hdl_source** Tcl variable. If neither of these methods are used, then the **search_path** variable is used. The **acs_read_hdl** command performs a Tcl glob-style expansion on an item and checks if the result is excluded by the **acs_exclude_list** or the **acs_exclude_extensions** variable. If it is not excluded and a file ends with one of the extensions from the **acs_vhdl_extensions** variable extensions list, it is considered a VHDL source file. If the file ends with one of the **acs_verilog_extensions**, it is considered a Verilog source file.

If an expansion is a directory, the command collects the files from the directory, and recursively from its subdirectories if the **-recursive** option is set. The command then performs all extension checks on these files. If the **-format** option is set, only files with Verilog or VHDL extensions are collected (based on the value of the option).

After **acs_read_hdl** collects all of the source files, it performs the following dependency checks (unless the **-no_dependency_check** option is specified).

- Detects Verilog include files to verify that they are not analyzed as source code files.
- If a Verilog include file that appears only in the list of HDL source files is detected, but its directory does not appear in the global **search_path** variable, and it is included in the Verilog source code without giving the path, then **acs_read_hdl** appends the include file's directory to the **search_path**. This enables the **analyze** or **read_verilog** commands to locate the file. You are notified about the changes made.
- Detects Verilog macro usage and definition and reorders files accordingly to ensure that they are analyzed in the correct order. This might not always be possible, such as when a macro is defined several times in different files or when macros are defined in files included by other include files.
- Determines the target library for each VHDL file. However, if the **-library** option is specified, this step is skipped and all VHDL files are analyzed into the specified design library.
- Orders the VHDL source files to ensure that they are analyzed in the correct

order.

After analyzing all source files, the design is elaborated from the top-level module, unless the **-no_elaborate** option is specified. The resulting GTECH representation is retained in memory.

If either the **-update** or the **-auto_update** option is specified, then the GTECH representation (unmapped design) is written to the specified destination directory. The default destination directory is `elab/db`.

When the **acs_read_hdl -auto_update** or **acs_read_hdl -update** command is called the first time, all of the source HDL files are analyzed and elaborated to create the unmapped design and the timestamp tables. Subsequent **acs_read_hdl -auto_update** commands only analyze and elaborate the updated HDL source files and subsequent **acs_read_hdl -update** commands only analyze and elaborate the specified HDL source files. The updated designs are replaced in the unmapped design. The first issuance of the **acs_read_hdl** command is determined by the presence of the unmapped design file and timestamp tables in the destination directory.

EXAMPLES

The following example assumes that the current directory is the source directory. It specifies the source file list at the command line and calls the command with the name of the top-level entity.

```
prompt> acs_read_hdl -hdl_source {.} -recurse E1
```

The next example specifies extensions for Verilog files that are different than the default (`{.v}`), sets the **-hdl_source** list and the **exclude_list** list, and calls the command with the name of the top-level module name, forcing it to only collect files with Verilog extensions.

```
prompt> set acs_verilog_extensions {.ve .VE}
prompt> set acs_hdl_source {mod1/src mod2/*/src}
prompt> set acs_exclude_list {mod1/src/incl mod2/*/src/incl}
prompt> acs_read_hdl -f verilog -no_dep TOP
```

Note that excluding include directories explicitly is only necessary if include files have the same extensions as source files and not all include files are included in the source.

```
prompt> acs_read_hdl -hdl_source src/mixed -auto_update
```

The first time the above command is issued, all of the mixed (Verilog and VHDL) HDL files from the specified directory are analyzed and elaborated, and the resulting unmapped design is written to the default destination directory along with the timestamp tables. The timestamp tables are used by the **acs_merge_design -auto_update** command to determine which designs are new, so that the partitions which depend on the updated designs are recompiled by the **acs_compile_design -update** command. The subsequent calls to the above command only analyze and elaborate the updated HDL files, and the corresponding designs are replaced in the unmapped design in the

destination directory.

SEE ALSO

`acs_merge_design(2)`
`acs_compile_design(2)`
`analyze(2)`
`elaborate(2)`
`acs_exclude_extensions(3)`
`acs_exclude_list(3)`
`acs_hdl_source(3)`
`acs_verilog_extensions(3)`
`acs_vhdl_extensions(3)`
`search_path(3)`

acs_recompile_design

Compiles an unmapped constrained .ddc file using time budgets. The time budgets are created by using a previously mapped design.

SYNTAX

```
status acs_recompile_design
-budget_source budget_pass
-destination destination_pass
[-source source_pass]
[-prepare_only]
[-force]
[-update]
[-update_source source_pass]
design_name
```

Data Types

<i>budget_pass</i>	string
<i>destination_pass</i>	string
<i>source_pass</i>	string
<i>design_name</i>	string

ARGUMENTS

- budget_source *budget_pass*
Specifies a pass directory from which the mapped design is picked up to generate time budgets for this pass.
- destination *destination_pass*
Specifies the pass name for this run. A directory structure for the pass is automatically created with the name *destination_pass*. All of the generated .ddc files, constraints, scripts, and reports are put under this pass directory.
- source *source_pass*
Specifies the pass directory from which Automated Chip Synthesis gets unmapped designs. The unmapped design is mapped using the budgeting information provided by the design from *budget_pass*. If this option is not used, Automated Chip Synthesis looks for the design in the memory. If the design is not in memory, the tool looks for the design from the pass0, which is the default source pass.
- prepare_only
Specifies that only preparation steps like constraint generation, script generation, creation of partition .ddc files, and makefile generation are performed. The actual distributed compile is not done.
- force
Specifies to force preparation steps such as constraint generation, script generation, creation of partition .ddc files, and makefile generation. By default, if a makefile already exists, the preparation steps are skipped.

-update

Performs an incremental design update by recompiling any compile partitions that have been updated by a previous **acs_merge_design** command, and then doing a top-level boundary compile.

-update_source *source_pass*

Specifies a pass directory that contains the merged design (the result of the **acs_merge_design** command) with the updated designs to be recompiled. If this option is not specified while the **-update** option is specified, then it looks for the design in memory. If the design is not in memory, then the tool looks for the default directory `${acs_default_pass_name}1_incr/db/pre_compile`.

design_name

Specifies the name of the top-level design. This argument is required.

DESCRIPTION

The **acs_recompile_design** command performs mapping from RTL to gates using parallel compile strategy. You must provide a constrained GTECH design in memory or specify an appropriate GTECH pass directory using the **-source** option. The command uses the partition information from the unmapped source design. The constraints for the partitions are derived by performing time budgeting on the mapped design coming from the *budget_pass*. The compile scripts are generated, and the makefile is written out. A parallel compile is then performed using LSF or the gmake utility, according to the settings of the **acs_num_parallel_jobs** variables.

Before you run this command, be sure to review the default settings in the `.synopsys_dc.setup` file in your project directory.

This command is implemented as a Tcl procedure that is visible and customizable. For details about how to change the behavior of this procedure, see the *Automated Chip Synthesis User Guide*.

EXAMPLES

The following example shows how to compile the design and rebudget the mapped netlist using **acs_refine_design**:

```
prompt> acs_compile_design DESIGN
```

```
prompt> acs_refine_design DESIGN
```

In the example above, **acs_compile_design** saves the compiled design in the `pass0` directory. The **acs_refine_design** command reads the mapped design netlist from the `pass0` directory, performs budgeting on the mapped netlist and saves the refined design in the `pass1` directory.

The following example incrementally refines the design that has been updated by the previous **acs_merge_design** command.

```
prompt> acs_merge_design -auto_update -dest merged_pass DESIGN
```

acs_recompile_design


```
prompt> acs_compile_design -update -update_source merged_pass \  
        -dest compiled_pass DESIGN
```

```
prompt> acs_refine_design -source compiled_pass \  
        -destination refined_pass DESIGN
```

The **acs_merge_design** command automatically identifies subdesigns that have been updated and creates a new design database in `merged_pass`. The subsequent **acs_compile_design** performs compile only on partitions that are marked as changed by **acs_merge_design** and saves the database in `compiled_pass`. Finally, **acs_refine_design** reads the mapped design database from `compiled_pass` and performs budgeting on the updated design.

SEE ALSO

```
acs_compile_design(2)  
acs_merge_design(2)  
acs_refine_design(2)  
acs_num_parallel_jobs(3)
```

acs_refine_design

Refines an already mapped design.

SYNTAX

```
status acs_refine_design
[-source pass_name]
[-destination pass_name]
[-prepare_only]
[-force]
[-update]
[-update_source source_pass]
design_name
```

Data Types

<i>pass_name</i>	string
<i>source_pass</i>	string
<i>design_name</i>	string

ARGUMENTS

- source *pass_name***
Specifies the Automated Chip Synthesis pass from which to get the mapped design.
If you do not specify this option, Automated Chip Synthesis assumes that the pass directory name is @pass@0, where string @pass@ is defined by the **acs_default_pass_name** variable.
If you do not specify a source pass, Automated Chip Synthesis uses pass0.
- destination *pass_name***
Specifies the pass name for this run. A directory structure for the pass is automatically created with the name *pass_name*. All of the generated .ddc files, constraints, scripts, and reports are put in this pass directory.
If you do not specify this option, Automated Chip Synthesis assumes that the pass directory name is @pass@1, where the string @pass@ is defined by the **acs_default_pass_name** variable.
- prepare_only**
Only the preparation steps such as constraint generation, script generation, creation of partition .ddc files, and makefile generation are performed. The actual distributed compile is not done.
- force**
Forces the preparation steps such as constraint generation, script generation, creation of partition .ddc files, and makefile generation. By default, if a makefile already exists, the preparation steps are skipped.
- update**
Performs an incremental design update by refining any compile partitions that have been updated by a previous **acs_merge_design** command, and then doing a top-level boundary compile.

`-update_source source_pass`

Specified a pass directory which contains the merged design (result of **acs_merge_design**) with the updated designs to be refined. If this option is not specified when the **-update** option is specified, then the tool looks for the design in memory. If the design is not in memory, the the tool looks for the default directory `${acs_default_pass_name}1_incr/db/pre_compile`.

`design_name`

The name of the top-level design. This argument is required.

DESCRIPTION

The **acs_refine_design** command remaps a design that is already mapped coming from the source pass directory specified. The partitioning of the source pass is preserved. The constraints for the partitions are derived by performing time budgeting on the mapped design coming from the source pass. The compile scripts are generated, and the makefile is written out. A parallel incremental or top-level compile is then performed using LSF or the gmake utility according to the settings of the **acs_num_parallel_jobs** variables.

Before you run this command, review the default settings in the `.synopsys_dc.setup` file located in your project directory.

This command is implemented as a Tcl procedure that is visible and customizable. For details of how to change the behavior of this procedure, refer to the *Automated Chip Synthesis User Guide*.

SEE ALSO

`acs_compile_design(2)`
`acs_merge_design(2)`
`acs_recompile_design(2)`
`acs_default_pass_name(3)`
`acs_num_parallel_jobs(3)`

acs_remove_dont_touch

Removes the dont_touch attributes on the specified designs.

SYNTAX

```
status acs_remove_dont_touch  
[-force | design_list]
```

Data Types

design_list list

ARGUMENTS

-force

Removes the dont_touch attributes on all compile partitions, including multiply-instantiated designs.

This option and the *design_list* argument are mutually exclusive.

design_list

Removes the dont_touch attributes on the specified designs.

This argument and the **-force** option are mutually exclusive.

DESCRIPTION

The **acs_remove_dont_touch** command removes the dont_touch attributes on designs according to the options. If you do not specify any options, the command removes the dont_touch attributes on all compile partitions except the multiply-instantiated designs.

This command applies to compile partitions only. It cannot remove the dont_touch attributes from nonpartition designs. For detailed information about partitions, see the man pages for the **set_compile_partitions** and **report_partitions** commands.

SEE ALSO

```
get_attribute(2)  
report_partitions(2)  
set_attribute(2)  
set_compile_partitions(2)  
set_dont_touch(2)
```

acs_report_attribute

Reports the ACS attributes on the specified partitions.

SYNTAX

```
status acs_report_attribute  
[-partitions partition_list]  
[attribute_name]
```

Data Types

<i>partition_list</i>	list
<i>attribute_name</i>	string

ARGUMENTS

`-partitions partition_list`
Specifies the compile partitions whose ACS attributes are reported.
If you do not specify this option, the command reports all compile partitions that have the specified attribute.

`attribute_name`
Specifies the attribute to report.
If you do not specify this option, the command reports all attributes set on the specified compile partitions.

DESCRIPTION

The **acs_report_attribute** command reports the ACS attributes set on the specified compile partitions.

SEE ALSO

```
acs_set_attribute(2)  
set_compile_partitions(2)  
report_partitions(2)
```

acs_report_directories

Reports the current directory structure settings.

SYNTAX

```
status acs_report_directories  
[-file_types type_list]
```

Data Types

type_list list

ARGUMENTS

-file_types type_list
The list of file types you would like to be reported.

DESCRIPTION

Returns all the paths that are defined in the directory structure template. Currently, only elements in `acs_dir` are printed.

When `-file_types` option is not used, `acs_report_directories` returns all the paths that are defined in the directory structure template and the file types bound to each path. If a list of file types is given with `-file_types` option, the command prints out the path where each given file type is assigned to.

EXAMPLES

Example 1:

```
prompt> acs_report_directories  
Report of ACS directory structure  
/remote/user/acs/new/@pass@: {makefile oc.tcl cstr.tcl}  
/remote/user/acs/new/@pass@/constraints: {pass_cstr}  
/remote/user/acs/new/@pass@/db/post_compile: {post_ddc}  
/remote/user/acs/new/@pass@/db/pre_compile: {pre_ddc}
```

Example 2:

```
prompt> acs_report_directories \  
      -file_type {pass_compile_script pass_cstr elab_ddc}  
Report of ACS directory structure  
pass_compile_script -> /remote/dtg215/rwang/acs/new/@pass@/scripts  
pass_cstr -> /remote/dtg215/rwang/acs/new/@pass@/constraints  
elab_ddc -> /remote/dtg215/rwang/acs/new/elab/db
```

SEE ALSO

`acs_check_directories(2)`

`acs_report_directories`

acs_get_path(2)

acs_report_user_messages

Reports the number of error, warning, and information messages.

SYNTAX

```
status acs_report_user_messages  
[-total]  
[-errors]  
[-warnings]  
[-infos]  
[-reset]
```

ARGUMENTS

-total

Prints the total number of error messages that have occurred since the the current session began. The default is to print the number of error messages since the last call to the **acs_report_user_messages** command.

-errors

Prints the number of error messages that have occurred since the last call to **acs_report_user_messages**.

-warnings

Prints the number of warning messages that have occurred since the last call to **acs_report_user_messages**.

-infos

Prints the number of informational messages that have occurred since the last call to **acs_report_user_messages**.

-reset

Resets the counters for the errors, warnings, and informational messages. However, subsequent calls to **acs_report_user_messages** with the **-total** option prints the number of error, warning, and information messages since the start of the session.

DESCRIPTION

The **acs_report_user_messages** command reports the number of error, warning, and information messages.

You can use any combination of the **-errors**, **-warnings**, and **-infos** options with the command. If no options are specified, then the effect is the same as if all three of the options are specified; all three types of messages are printed. Note that the **-total** option is independent of these three options.

The ACS chip-level compile commands **acs_compile_design**, **acs_recompile_design**, and **acs_refine_design** issue the **acs_report_user_messages -reset** command when the commands begin.

EXAMPLES

The following example prints the number of error, warning, and information messages since the last call to **acs_report_user_messages**:

```
prompt> acs_report_user_messages
```

The following example prints the number of warning and error messages that have occurred since the last call to **acs_report_user_messages**:

```
prompt> acs_report_user_message -errors -warnings
```

The following example prints the number of errors since the start of the session:

```
prompt> acs_report_user_message -errors -total
```

SEE ALSO

```
acs_compile_design(2)  
acs_recompile_design(2)  
acs_refine_design(2)
```

acs_reset_directory_structure

Resets the project directory tree of Automated Chip Synthesis to its default setting.

SYNTAX

```
status acs_reset_directory_structure
```

ARGUMENTS

This command has no arguments.

DESCRIPTION

This command resets the project directory tree for Automated Chip Synthesis to its default setting. The command also resets any changes you make to the default directory structure using the **acs_user_dir** array variable.

SEE ALSO

```
acs_get_path(2)  
acs_report_directories(2)  
create_pass_directories(2)
```

acs_set_attribute

Sets Automated Chip Synthesis compile attributes on one or more partitions.

SYNTAX

```
status acs_set_attribute
[-partitions partitions]
attribute_name
[attribute_value]
```

Data Types

<i>partitions</i>	list or collection
<i>attribute_name</i>	string
<i>attribute_value</i>	data type

ARGUMENTS

-partitions *partitions*
Specifies a partition design list or collection. The command checks each design to determine if it is a partition. If the design is not a partition, a warning message occurs, and the command continues with the attribute setting on that design.
If this option is not specified, the attribute is set on all of the partitions.

attribute_name
Specifies the name of the attribute that you want to set. This argument is required.

attribute_value
Specifies the attribute value that you want to set.
If this option is not specified, the command uses the default value of the attribute, if any. If there is no default value, the attribute value is set to an empty string ("").

DESCRIPTION

The **acs_set_attribute** command sets the Automated Chip Synthesis supported attributes on partitions.

The following lists the ACS supported attributes, their valid values, and their default value.

Attribute Name	Attribute Value	Default
OptimizationPriorities	area/timing/ area_timing/timing_area	timing_area
PreserveBoundaries	true/false	flow specific
CompileVerify	true/false	false

TestReadyCompile	true/false	false
MaxArea	<area estimate in lib units>	unspecified
CanFlatten	true/false	false
FullCompile	none/low/medium/high	medium
IncrementalCompile	none/low/medium/high	medium
BoundaryCompile	none/low/medium/high/top	top
UltraOptimization	true/false	false
AutoUngroup	false/delay/area [numcells]	false
CompileUltra	true/false	false
TargetCompiler	dc_shell/psyn_shell/fpga_shell	dc_shell

Note: Default value of PreserveBoundaries is true for compile and false for compile_ultra.

EXAMPLES

The following example sets the Automated Chip Synthesis **TestReadyCompile** attribute to **true** on the {STACK_TOP, STACK_MEM} partitions and the ALU design, which is not a partition:

```
prompt> acs_set_attribute TestReadyCompile true \
      -partition [get_designs {STACK_TOP STACK_MEM ALU}]
```

WARNING: ALU is not a partition

SEE ALSO

```
acs_report_attribute(2)
report_partitions(2)
set_compile_partitions(2)
```

acs_submit

Specifies the compile configuration for normal compile partitions.

SYNTAX

```
status acs_submit  
[-command command]  
[-host host]  
[-exec exec]  
[-show]
```

Data Types

<i>command</i>	string
<i>host</i>	string
<i>exec</i>	string

ARGUMENTS

- command *command***
Specifies the command to use to submit the compile jobs; for example, "bsub ..." or "qsub ...".
ACS will automatically append the following after the user specified command "-o <compile_log>" so the user specified command must support the -o option for specifying a log file. If the command is machine executable which does not support -o option, then the executable can be invoked from a shell script supporting -o option.
If you do not specify this option, ACS uses the default batch submission command, "bsub -K -P".
- host *host***
Specifies from which host to submit the compile jobs.
If you do not specify this option, ACS submits the compile jobs from the your host machine.
- exec *exec***
Specifies which executable file to use for the compile jobs.
If you do not specify this option, ACS uses the executable file specified by the `acs_dc_exec` variable (if you are using a logic synthesis flow) If it is not defined, ACS uses the current executable file.
- show**
Prints the current settings of the following variables: `acs_submit_command`, `acs_submit_host`, and `acs_submit_exec`.

DESCRIPTION

The **acs_submit** command specifies the compile configuration for normal compile partitions, including the batch submission command, the executable file, and the host used to run or submit the jobs.

NOTE: This command applies to normal compile partitions only. For information about large compile partitions and setting the compile configuration for them, see the **acs_submit_large** man page.

SEE ALSO

acs_submit_large(2)
set_compile_partitions(2)
report_partitions(2)
acs_dc_exec(3)

acs_submit_large

Specifies the compile configuration for large compile partitions.

SYNTAX

```
status acs_submit_large
[-command command]
[-host host]
[-exec exec]
[-partitions part_list]
[-show]
```

Data Types

<i>command</i>	string
<i>host</i>	string
<i>exec</i>	string
<i>part_list</i>	list

ARGUMENTS

- command *command***
Specifies the command to use to submit the compile jobs for large compile partitions.
ACS will automatically append the following after the user specified command "-o <compile_log>" so the user specified command must support the -o option for specifying a log file. If the command is machine executable which does not support -o option, then the executable can be invoked from a shell script supporting -o option.
If you do not specify this option, ACS uses the default batch submission command, "bsub -K -P".
- host *host***
Specifies from which host to submit the compile jobs for the large compile partitions.
If you do not specify this option, ACS submits the compile jobs from the your host machine.
- exec *exec***
Specifies which executable file to use for the compile jobs for the large compile partitions.
If you do not specify this option, ACS uses the executable file specified by the acs_dc_exec variable (if you are using a logic synthesis flow) If this variables is not defined, ACS uses the current executable file.
- partitions *part_list***
Specifies a list of partitions which are "large partitions". The top-level design is always considered a large partition, whether you explicitly specify it or not.
- show**
Prints the current compile configuration for large compile partitions.

DESCRIPTION

The **acs_submit_large** command specifies the compile configuration for large compile partitions, including the batch submission command, the executable file, the host used to run or submit the jobs, and which partitions are considered large partitions.

SEE ALSO

acs_submit(2)
set_compile_partitions(2)
report_partitions(2)
acs_dc_exec(3)

acs_write_html

Creates an HTML page in the destination directory.

SYNTAX

```
status acs_write_html  
-destination pass_name  
[-acs_work_dir acs_working_directory]
```

Data Types

<i>pass_name</i>	string
<i>acs_working_directory</i>	string

ARGUMENTS

-destination *pass_name*
Specifies the destination pass directory name for this run. If the *pass_name* directory does not exist, the HTML page is not created.

-acs_work_dir *acs_working_directory*
Specifies the ACS working directory. The default is the current directory.

DESCRIPTION

The **acs_write_html** command creates an HTML page in the destination directory. The HTML page contains links to compile scripts, constraint files, and logfiles generated with the **acs_compile_design**, **acs_refine_design**, or **acs_recompile_design** command. The command also provides links to user-created constraint files and script files.

The HTML file is created once during the preparation stage of **acs_compile_design**, **acs_refine_design**, and **acs_recompile_design**. The file is updated again after compiling the top-level partition.

EXAMPLE

The following example creates an HTML page for *top_design_name* in the *pass0* destination directory:

```
prompt> acs_write_html top_design_name -destination pass0
```

SEE ALSO

```
acs_recompile_design(2)  
acs_refine_design(2)  
acs_merge_design(2)
```

add_module

Reads in a specified library file containing module functional information and uses it to update an existing technology library.

SYNTAX

```
status add_module
[-overwrite]
[-force]
[-permanent]
file_name
library_name
[-no_warnings]
```

Data Types

<i>file_name</i>	string
<i>library_name</i>	string

ARGUMENTS

-overwrite

Indicates that a group declared in *file_name* is to overwrite any group in *library_name* having the same name. Only groups added by **add_module** or **model** can be overwritten. Groups included in the original library and groups added with **-permanent** cannot be modified. 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.

-force

Indicates that some library-level groups/attributes declared in *file_name* can overwrite the same groups/attributes included in the original library. Such library-level groups are : wire_load, power_supply and operating_conditions. Such library-level attributes are: <k_factors>, in_place_swap_mode, default_wire_load_mode, default_wire_load_selection, default_wire_load_capacitance, default_wire_load_resistance, default_wire_load_area, default_operating_conditions.

-permanent

Indicates to store groups declared in *file_name* in the library in such a way that they cannot be overwritten.

file_name

Specifies the name of the file in which one or more new groups containing module functional information 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.

library_name

Specifies the name of the library to be updated; the library must be resident in memory. This option can be a simple filename with no directory

add_module

specification and, therefore, no slash ("/") (for example, `test_db` or `library.db`). Simple filenames must be found in a directory listed in the `search_path`. Alternatively, this option can be a complex filename, which has a directory specification (for example, `~synopsys/dc/test.lib`). Complex filenames are not included in the `search_path`.

-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

This command reads in a library file and adds to the specified library only those groups with module functional information declared in the library file. Groups are added to the library as if they were entered at the end of the original text of the library. If Library Compiler finds any errors while processing a group, it does not add that group to the library. You do not need a Library Compiler License to add groups with module functionality to a technology library.

For details on library file syntax, refer to the Library Compiler manuals.

EXAMPLES

The following example adds the library file `add_ram.lib`, which contains module information, (that is, a RAM group) to the library memory.

```
prompt> add_module add_ram.lib memory
```

The following example adds the library file `romgen.lib`, which contains a cell description of a ROM cell, to the library `cmos`. The **-permanent** flag indicates that the new cell cannot be overwritten.

```
prompt> add_module -permanent romgen.lib cmos
```

SEE ALSO

```
read_lib(2)  
update_lib(2)  
write_lib(2)
```

add_pg_pin_to_db

Converts a .db logic library that is not based on PG pin syntax to a .db logic library with PG pin syntax.

SYNTAX

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

Data Types

<i>input_db_filename</i>	string
<i>mw_lib_name</i>	string
<i>pg.map</i>	string
<i>template_filename</i>	string
<i>pg_db_filename</i>	string

ARGUMENTS

input_db_filename

Specifies the name of the input logic library .db file that is not based on PG pin syntax. This option is required.

-mw_library_name mw_lib_name

Specifies one or more Milkyway library names or FRAM views that correspond to the input .db file.

You must specify either the **-mw_library_name** option or the **-pg_map_file** option for single-rail cells. If the cells in the library are multiple-rail cells, you must specify the **-pg_map_file** option. You do not need to specify the **-mw_library_name** option if the PG map file is complete; however, if the map file is not complete, you must specify **-mw_library_name**.

-pgap_file pgap

Specifies the file name for mapping between data that is not based on PG pin syntax and data that is based on PG pin syntax. Specify either the **-pg_map_file** option or the **-mw_library_name** option for single-rail cells. If the cells in the library are multiple-rail cells, specify the **-pg_map_file** option. You do not need to specify the **-mw_library_name** option if the PG map file is complete; however, if the map file is not complete, you must specify **-mw_library_name**.

The **-pg_map_file** option and **-pg_map_template** option are mutually exclusive. You can use one option, but not both.

add_pg_pin_to_db

-pg_map_template *template_filename*

Specifies the file name for the PG map file to be automatically generated. Use this option if you want to quickly generate a PG map file template to use as the first step for building the final PG map file. Later, you can use this PG map file template to complete the final conversion step with the **-pg_map_file** option.

The **-pg_map_template** option and **-pg_map_file** option are mutually exclusive. You can use one option, but not both.

You can use the **-pg_map_template** option with the input non-PG library file only or with the **-mw_library_file** option with the Milkyway FRAM view. Include Milkyway FRAM views during map template generation so that the **add_pg_pin_to_db** command can generate a nearly complete PG map file. This is because PG information that is missing from the logic library can be derived from the Milkyway views during PG map file generation, creating a more complete PG map file.

-expanded

Generates a PG map file with all the wildcards expanded. The option is turned off by default, meaning that the **add_pg_pin_to_db** command generates a PG map file that has as many wildcards as possible in compressed format in order to reduce the file size.

-verbose

Outputs all sections of the PG map data in a log file, whether the PG map data is generated automatically or expanded from the map file.

-output *pg_db_filename*

Specifies the name of new PG pin-based file that is generated after the conversion. It cannot be the same name as the input .db file. This option is required.

DESCRIPTION

Converts a .db logic library that is not based on PG pin syntax to a .db logic library with PG pin syntax. You can use **add_pg_pin_to_db** in the following scenarios:

With a complete Milkyway library

If all of the cells have a single power and ground rail (1P1G), the **-pg_map_file** option is not required. However, you should specify the **-mw_library_name** option to derive the mapping information from the Milkyway Fram view. The **-pg_map_file** option is required for cells that have multiple power and ground rails.

Without a Milkyway library (.db only)

If you specify a logic library (.db) only and no Milkyway library for a single power and ground rail library or a multiple-rail library, you must specify the **-pg_map_file** option to do the conversion.

With a partial Milkyway library

If some cells exist in both the logic library and the Milkyway library, while others exist in the logic library only, specify the **-pg_map_file** for the cells that are missing in the Milkyway library and for the power management (PM) cells.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example converts the old.db logic library, which is not based on PG pin syntax, into the pg.db logic library, which uses PG pin syntax:

```
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}
```

SEE ALSO

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

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]
```

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]
```

add_to_rp_group

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 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.

-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}
```

add_to_rp_group

```
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)
```

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)`

alib_analyze_libs

Reads in the target library files, analyzes each of them separately, and creates the corresponding alib files.

SYNTAX

```
int alib_analyze_libs
```

DESCRIPTION

Parses the *target_library* string. For each of the specified target libraries, tries to find a valid alib. If no valid alib is found for target library X, library analysis occurs for X and the result is stored as X.alib in the *alib_library_analysis_path* directory. Note that the analysis is not affected by *dont_use* settings in the script where the command is issued.

EXAMPLES

To analyze the libraries x.db and y.db:

```
set target_library "x.db y.db"  
set alib_library_analysis_path "./out"  
alib_analyze_libs
```

Two files, x.db.alib and y.db.alib are generated in a versioned directory under "./out". The directory is named "./out/alib-N", where N is the alib version number. To load these alibs, set *alib_library_analysis_path* to "./out" in your compile script. You need not specify the versioned directory name, since this detail is managed entirely by the tool.

SEE ALSO

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.

Creation and usage of scenarios is available with Design Compiler Graphical.

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_clock_gates

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

SYNTAX

```
collection all_clock_gates
[-no_hierarchy]
[-clock clock_name]
[-cells]
[-enable_pins]
[-clock_pins]
[-output_pins]
[-test_pins]
[-observation_pins]
```

Data Types

clock_name 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 clock gating cells for registers originally clocked by *clock_name* in the search. By default, this option is off.
- cells
Returns cells. This is the default.
- enable_pins
Returns enable pins instead of cells. By default, this option is off.
- clock_pins
Returns clock input pins instead of cells. By default, this option is off.
- output_pins
Returns gated clock output pins instead of cells. By default, this option is off.
- test_pins
Returns test_mode or scan_enable input pins instead of cells. By default, this option is off.

DESCRIPTION

This command returns a collection of clock gating cells or pins in the current instance, filtered as specified by the options. By default, the command lists clock gating cells in the design. If you specify *clock_name*, it considers clock gating

all_clock_gates

cells in the transitive fanout of the specified clock in the search. Normally, it considers all clock gating cells in the search. The **-cells** option, which is the default, can be combined with one or more of the pins options.

EXAMPLES

The following example tags all clock gates for registers originally clocked by *CLK* for removal.

```
prompt> remove_clock_gating -gating all_clock_gates(-clock CLK)
```

The following example returns a list of test pins for all clock gates for registers originally clocked by *CLK*.

```
prompt> all_clock_gates -test_pins -clock CLK
```

SEE ALSO

```
current_design(2)  
current_instance(2)  
report_clock_gating(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_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)  
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_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.

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_operand_isolators

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

SYNTAX

```
collection all_operand_isolators  
[-no_hierarchy]  
[-cells]  
[-control_pins]  
[-data_pins]  
[-output_pins]
```

ARGUMENTS

-no_hierarchy
Limits the search to only the current level of hierarchy. Subdesigns are not searched. By default, this option is off.

-cells
Returns cells. This is the default.

-control_pins
Returns control pins instead of cells. By default, this option is off.

-data_pins
Returns data input pins instead of cells. By default, this option is off.

-output_pins
Returns data output pins instead of cells. By default, this option is off.

DESCRIPTION

This command returns a collection of operand isolation cells or pins in the current instance, filtered as specified by the options. By default, the command lists operand isolation cells in the design. The **-cells** option, which is the default, can be combined with one or more of the pins options.

EXAMPLES

The following example returns the data output pins of all operand isolation cells in the design.

```
prompt> all_operand_isolators -output_pins
```

SEE ALSO

```
current_design(2)  
current_instance(2)  
report_operand_isolation(2)
```

all_operand_isolators

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_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]
```

all_registers

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)  
compile_seqmap_enable_output_inversion(3)
```

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}
```

all_rp_inclusions

```
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}
```

all_rp_instantiations

```
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.

Creation and usage of scenarios is available with Design Compiler Graphical.

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_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  
[input_coll]
```

Data Types

input_coll collection

ARGUMENTS

input_coll
Searches the specified input collection and returns a collection of tie-off cells. Objects are to be searched only from the content of the specified input collection, rather than from the design.

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)
```

analyze

Analyzes the HDL files and stores the intermediate format in the specified library.

SYNTAX

```
status analyze
[-library library_name]
[-work library_name]
[-format verilog | sverilog]
[-vcs vcs_opts]
[-create_update]
[-update]
[-define define_list]
file_list
```

Data Types

<i>library_name</i>	string
<i>vcs_opts</i>	string
<i>define_list</i>	string
<i>file_list</i>	list

ARGUMENTS

-library *library_name*
Remaps the work library to *library_name*. By default, **analyze** stores all output in the work library. To store design elements in libraries other than **-work**, use **-library**.

-work *library_name*
Specifies an alias for **-library**.

-format verilog | sverilog
Specifies the format of the files that are to be analyzed. The supported formats are VHDL, Verilog, and SystemVerilog.

-vcs *vcs_opts*
Specifies all VCS-specific command-line options. The options be enclosed with double quotation marks (""). Options specified by **-vcs** follow the VCS command-line syntax:
[-sverilog | -verilog] Specifies the format of the files that are to be analyzed.
[-y <directory_path>] Specifies directories contain the library files that will be searched for unresolved module instantiations in the user design.
[+libext+<.extension1>+...] States the extension to consider during files search in the -y library directories (default: no extension).
[-v <library_file>] Holds several module definitions, to be used during the search for unresolved modules.
[-f <command_file>] Specifies a command file.
[+define+<macro_name>+...] Defines a macro.
[+incdir+<dir1>+...] Include directories into search list.
<src_file1> Specifies the files that are to be analyzed.

-create_update
Facilitates the packaging of HDL files for distribution performed by developers of DesignWare parts For details, see the *DesignWare Developer's Guide*.

-update
Facilitates the installation of DesignWare parts in DesignWare installation scripts. For details, see the *DesignWare Developer's Guide*.

-define *define_list*
The labels listed act as though passed to define without a value. This is useful for conditional inclusion of Verilog code enclosed by ``ifdef`/`endif`. For details, see the *HDL Compiler (Presto Verilog) Reference Manual*.

file_list
Specifies a list of files to be analyzed. When specifying more than one file, enclose the files in braces `{}`.

DESCRIPTION

This command translates the specified HDL files and stores the intermediate format in the specified library.

EXAMPLES

The following example analyzes *packages.vhd* and stores the results in the *MY_LIB* design library:

```
prompt> analyze -work MY_LIB -f vhd1 packages.vhd
```

SEE ALSO

`define_design_lib(2)`
`elaborate(2)`
`read_file(2)`
`report_design_lib(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)
```


apply_clock_gate_latency

Annotates the clock latencies on the existing clock gating cells based on the settings previously specified using the **set_clock_gate_latency** command.

SYNTAX

status **apply_clock_gate_latency**

ARGUMENTS

The **apply_clock_gate_latency** command has no arguments.

DESCRIPTION

This command annotates the clock latencies on the existing clock gating cells based on the settings previously specified using the **set_clock_gate_latency** command.

Use this command to do the following:

- Manually annotate clock latency values specified by **set_clock_gate_latency** command to existing clock gating cells
- Modify clock latency values applied to clock gating cells after gate-level clock gating is finished.

If an inconsistent clock latency setting is detected during annotation, the tool tries to resolve it to produce a situation where clock latencies are monotonically increasing values in the path going from the clock source to the gated registers. The following actions can be performed to achieve this:

- If a clock gating cell A is directly driving one or more registers, and some clock latency settings have been provided for the timing of these gated registers (by using value 0 for **stage** option of **set_clock_gate_latency** command or by propagation of latency specified for clock object), and in this situation the latency annotated on cell A is higher than the latency provided for gated registers, then latency on clock gating cell A is overwritten with the latency provided for gated registers. If this fix is done, it is informed by warning message PWR-746.
- If a clock gating cell A is driving another clock gating cell B, and clock latency set on A is higher than the one set on B, the tool resolves it by overwriting the clock latency on A with the value set on B. If this fix is done, it is also informed by warning message PWR-746.

To remove the settings specified using **set_clock_gate_latency**, use **reset_clock_gate_latency**.

SEE ALSO

remove_clock_latency(2)
report_clock_gating(2)
reset_clock_gate_latency(2)

```
set_clock_gate_latency(2)
set_clock_latency(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)
```

balance_buffer

Builds a balanced buffer tree on user-specified nets and drivers.

SYNTAX

```
int balance_buffer  
[-from start_point_list]  
[-to end_point_list]  
[-net net_list] [-force]  
[-library library_name]  
[-prefer buffer | inverter | lib_cell_name]
```

Data Types

<i>start_point_list</i>	string
<i>end_point_list</i>	string
<i>net_list</i>	string

ARGUMENTS

-from *start_point_list*
Specifies a list of driver pins from which buffer trees are to be created.

-to *end_point_list*
Specifies a list of loads to buffer.

-net *net_list*
Specifies a list of nets to buffer.

-force
Forces **balance_buffer** implementation regardless of cost effect.

-library *library_name*
Specifies the library where the *lib_cell_name* is contained. You must specify this option when specifying *lib_cell_name*. If you do not specify *lib_cell_name*, this option is ignored.

-prefer buffer | inverter | *lib_cell_name*
Specifies the preferred cell type for the buffer tree. *buffer* creates a buffer tree using buffer-type library cells. *inverter* creates a buffer tree using inverter-type library cells. If *lib_cell_name* is specified, that library cell is the only cell used to form the buffer tree. You must specify *library_name* when specifying *lib_cell_name*. The library cell must be a buffer or an inverter library cell.

DESCRIPTION

The **balance_buffer** command creates DRC-free balanced buffer trees on user-specified nets and drivers. It removes existing buffer trees on the specified nets or drivers if any are present and builds a new DRC-free buffer tree. The buffer trees created in one hierarchy considers loads and drivers across hierarchies and will create

buffer trees at the fanout nets into these hierarchies.

The **balance_buffer** command generates a layered buffer tree, where each stage uses the same buffer or inverter and each gate of a given stage roughly drives the same load capacitance. The type of cells used are a combination of buffer and inverter cells available from the target library, unless the **-prefer** option is specified.

Note that any input port driving a net that is to be buffered by **balance_buffer** must have its drive value set. Otherwise, the default value specifies that the input port has infinite drive, and no buffer tree is created. Also note that **balance_buffer** uses the NLDM information specified in the target library for calculations on a buffer tree.

NOTE: **balance_buffer** does not take clock skew into account and, therefore, is not recommended for clock trees.

EXAMPLES

In the following example, a buffer tree is first built from port i0. Then, two additional buffer trees are built to drive load1 and load2.

```
prompt> read -f edif test.edif
prompt> set_driving_cell -cell IV i0
prompt> balance_buffer -from i0
prompt> balance_buffer -to {load1, load2}
```

SEE ALSO

compile(2)

balance_registers

Moves the registers of the current design to achieve a minimum cycle time.

SYNTAX

```
int balance_registers
```

DESCRIPTION

This command moves the registers of a design to obtain a minimum cycle time. You do not have to set the **balance_registers** attribute to invoke **balance_registers** outside **compile**. Using **balance_registers** outside of **compile** usually gives better control and better quality of results because the gate level netlist used as a starting point is defined exactly by the initial compile.

balance_registers has the following requirements:

- 1) All registers must be edge-triggered flip-flops clocked by the same phase of the same clock; or, all registers must be master-slave elements with the same signals for the master and slave pins.
 - 2) Flip-flops and master-slave elements cannot be present in the same design.
 - 3) Master and slave clock waveforms cannot overlap.
 - 4) The clock pins of all flip-flops in the design must be connected to the same clock port. The interconnection from the clock port to the clock pins can contain buffer and inverter cells having one input and one output pin. All paths from the clock port to the clock pins of the registers can contain either an even number (including zero) of inverters or an odd number of inverters. These paths cannot be comprised of both even and odd numbers of inverters in a single design.
- The outputs of the clock network can be connected to only sequential cells. The clock network of the retimed design will not contain any cells.
- 5) Only certain FPGA technologies are supported. See the DCFPGA documentation for more details.
 - 6) Flip-Flops that have asynchronous set or clear pins will not be moved.
 - 7) Designs cannot contain a combinational loop.
 - 8) The timing constraints for the design should be set in the following way: The external clock port of the design must have a clock constraint created by the `create_clock` command. This will be called the base clock. All primary inputs of the design should have a non-negative input delay relative to the base clock. All primary outputs of the design should have a non-negative output delay relative to the base clock. Negative input and output delays will be tolerated but the quality of the final retiming result may be worse. Input and output delays relative to virtual clocks or clocks other than the base clock will be ignored. Point-to-point timing exceptions as created by the `set_false_path`, `set_multicycle_path`, `set_max_delay` and `set_min_delay` commands will be tolerated but ignored. Case

analysis constraints will also be ignored.

9) Designs cannot contain tri-state cells or unmapped synthetic library components.

By default, **balance_registers** ungroups all instances in a design before moving registers. Avoid ungrouping by setting the **dont_touch** attribute on instances that are not to be ungrouped. The **balance_registers** command does not ungroup an instance that has the **dont_touch** attribute, and does not place any registers inside an instance that has the **dont_touch** attribute.

The **balance_registers** command includes a delay modeling capability. For the sake of predictability, the algorithm selects a flip-flop from the target library with small setup time, good drive at the outputs, and low input loading as compared with other registers in the library. This register is designated as the preferred flip-flop. Sequential mapping is invoked in the final step to improve the circuit by remapping the registers. The preferred flip-flop helps provide tighter bounds on the performance variation after the **balance_registers** sequence. To disable delay modeling, set the shell variable **balance_reg_delay** to 0.

As part of the delay modeling capability, **balance_registers** provides some statistics on the delays in the circuit. If the preferred flip-flop appears as a driver for a net, **balance_registers** analyzes the circuit to compute the clock-pin-to-next-state-pin delay for that net. The maximum of all these clock-pin-to-next-state pin delays is used as a reference for checking the input delays of the design. They should be larger than the average clock-pin-to-next-state pin delay. The clock-pin-to-next-pin-delay, the setup time of the preferred flip-flop and the clock uncertainty sum up to the clock correction. The sum of the clock correction and the maximum critical path length reported when balance registers has finished the register moving phase give an estimate for the clock period after retiming. However, the value of the clock correction does not influence the minimum clock period achieved. An example detailing the output of the delay modeling capability is shown in the EXAMPLES section. More information on the **balance_registers** command can be found in the Design Compiler reference manual.

EXAMPLES

The following example reads in the design `pipe_mult_32.db`, and moves the registers to achieve a minimum cycle time.

```
prompt> read pipe_mult_32.db
prompt> balance_registers
```

The output provided by the delay modeling capability shows that the sequential cell F611 from the target library is selected as the preferred flip-flop F611. The relevant delay information of the design is summarized in the two tables titled "Design Analysis" and "Clock-to-Q delay distribution."

```
prompt> balance_registers pipe_test
Loading design 'pipe_test'

Beginning retiming
-----
Retiming pipe_test
```


Preferred register is F611 with setup = 1.34

Design Analysis

	Worst	Best	Average
clock to Q	1.85	1.40	1.54

Clock to Q delay distribution

delay range	number of nets
[1.40 - 1.49]	23
[1.49 - 1.58]	33
[1.58 - 1.67]	14
[1.67 - 1.76]	4
[1.76 - 1.85]	4

Lower bound estimate = 5.40

Critical path length = 5.40

Clock correction = 3.19 (clock-to-Q delay = 1.85, setup = 1.34, uncertainty = 0.

00) Structuring 'pipe_test'

Mapping 'pipe_test'

Retiming complete

Transferring Design 'pipe_test' to database 'mult4_g.db'

1

SEE ALSO

current_design(2)

set_balance_registers(2)

set_dont_touch(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)

cell_of

Returns the cell objects for the specified pins in the current design.

SYNTAX

```
list cell_of  
[object_list]
```

Data Types

object_list list

ARGUMENTS

object_list
Specifies a list of pins in the current design whose cells are returned by the **cell_of** command.

DESCRIPTION

The **cell_of** command returns a list of cell objects that own the pins specified in the *object_list*. If no cells are found, the return value is an empty list ({}). Double quotation marks (" ") are needed for pin names that are ambiguous to the command parser.

EXAMPLES

The following example returns the cell of the specified pin in the current design:

```
prompt> cell_of U1/P  
  
{"U1"}
```

The following example returns the cell of pin "reg(8)/Q" in the current design:

```
prompt> cell_of "reg(8)/Q"  
{"reg(8) "}
```

The following example returns the list of cells for the given list of pins. The first cell is the owner cell of the first pin and the second cell is the owner cell of the second pin.

```
prompt> cell_of {U3/B U9/A}  
{"U3", "U9"}
```

The following example returns the cell of the given pin. The cell name is a hierarchical name at the current level, which can be found after ungrouping.

```
prompt> cell_of {U3/U1/B}  
{ "U3/U1" }
```

The following example returns a warning message when the pin does not exist in the current design:

```
prompt> cell_of U999/Z  
Warning: Can't find pin 'U999/Z' in design 'TOP'. (UID-95)  
{ }
```

The following example returns the cell of a hierarchical pin.

```
prompt> cell_of U1/U2/U3/A  
{ "U1/U2/U3" }
```

SEE ALSO

`all_connected(2)`

change_link

Changes the design to which a cell is linked.

SYNTAX

```
status change_link  
object_list  
design_name  
[-all_instances]  
[-force]
```

Data Types

<i>object_list</i>	list
<i>design_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.

-all_instances
Enables the command to accept instance cells in the *object_list*.

-force
Enables the command to allow mismatched pin counts in the *object_list*.

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.

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** command.

Use the **-all_instances** option when any cell in the *object_list* argument is an instance in the lower hierarchy and its parent cell is not unique. All similar instance cells under the same parent design are automatically linked to the new reference design. You do not have to change the current design to change the link for the instance cells in the lower design hierarchy. You do not need to use the **-all_instances** option if none of the cells in the *object_list* argument is an instance in the lower hierarchy or its parent cell is unique.

A common use of the **change_link** command for hierarchical cells is to rename designs and to change the cell links to the designs of the new names. In these cases, use the **rename_design** command with the **-update_links** option instead of the **copy_design**, **rename_design**, **change_link** and **remove_design** command set. The **rename_design** command provides faster runtime. For more information, see the **rename_design** command man page.

EXAMPLES

The following example shows a common use of the **change_link** command. The *ADDER* design is copied to a new design named *MY_ADDER*. The *U1* and *U2* cells are then linked from the current design to *MY_ADDER*.

```
prompt> copy_design ADDER MY_ADDER
Copying design 'ADDER' to 'MY_ADDER'

prompt> change_link {U1, U2} MY_ADDER
```

The following example shows an improper use of the **change_link** command. The specified cell does not have the same number of ports as the design to which it is being linked.

```
prompt> change_link U3 HALF_ADDER
Error: Number of pins on cell 'U3' don't equal design 'HALF_ADDER'
```

The following example uses the **-all_instances** option:

```
prompt> current_design
Current design is 'top'.
{top}

prompt> get_cells -hierarchical -filter "ref_name == bot"
{mid1/bot1 mid1/bot2}
1

prompt> get_cells -hierarchical -filter "ref_name == IVA"
{mid1/bot1/inv1 mid1/bot2/inv1 mid1/bot1/inv2 mid1/bot2/inv2 mid1/inv3}
1

prompt> change_link mid1/bot1/inv1 lsi_10k/IVAP -all_instances
Information: Changed link for all instances of cell 'inv1' in subdesign 'bot'.
(UID-193)
1
```

```
prompt> get_cells -hierarchical -filter "ref_name == IVAP"
{mid1/bot1/inv1 mid1/bot2/inv1}
1

prompt> get_cells -hierarchical -filter "ref_name == IVA"
{mid1/bot1/inv2 mid1/bot2/inv2 mid1/inv3}
1
```

In the previous example, the bot design has cells named inv1 and inv2 that were linked to lsi_10k/IVA. The mid1/bot1 and mid1/bot2 cells instantiate the bot design. The mid1/inv3 cell is also linked to lsi_10k/IVA. After running the **change_link** command in the example, the mid1/bot1/inv1 and mid1/bot2/inv1 cells are relinked to lsi_10k/IVAP, because they are the instances of the same inv1 cell in the bot design. The mid1/bot1/inv2 and mid1/bot2/inv2 cells are not relinked because they are not the instances of the inv1 cell in the bot design. The mid1/inv3 cell is not relinked because it is not in an instance that instantiates the bot design.

SEE ALSO

```
copy_design(2)
current_design(2)
link(2)
rename_design(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 ILM.  
  
-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 ILM view with the FRAM view for the mem_ctrl icc-ilm.

```
prompt> change_macro_view -reference mem_ctrl -view FRAM
```

Limitations

The command is only available in DC-Topographical mode.

SEE ALSO

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

change_names

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]
```

change_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(2)  
filter_collection(2)  
get_selection(2)  
query_objects(2)
```

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

```
compile(2)  
create_clock(2)  
current_design(2)  
group_path(2)  
link(2)  
read_sdf(2)  
remove_annotated_check(2)  
remove_annotated_delay(2)  
remove_constraint(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)
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)
set_wire_load_model(2)
set_wire_load_mode(2)
set_wire_load_selection_group(2)
set_wire_load_min_block_size(2)
uniquify(2)
```


check_bindings

Checks the bindings in a synthetic library module definition.

SYNTAX

```
integer check_bindings  
[-bindings binding_list]  
[-pin_widths pin_width_list]  
module_name
```

Data Types

<i>binding_list</i>	list
<i>pin_width_list</i>	list
<i>module_name</i>	string

ARGUMENTS

-bindings *binding_list*
Lists the bindings of the module to check. The default is to check all bindings.

-pin_widths *pin_width_list*
Lists the pin width that is checked for the *bound_operator*. The default is specified by the operator's **check_pin_widths** attribute. The **-pin_widths** option can be used only if all specified bindings bind to the same operator.

module_name
Specifies the name of the synthetic library module whose bindings you want to check. This parameter is required.

DESCRIPTION

This command ensures that bindings from a synthetic library module to synthetic library operators are correct.

By default, **check_bindings** checks every binding of the specified module. The **check_bindings** command verifies that a binding implements a particular operator correctly. The binding under test is correct if it generates a design that is functionally the same as the "golden" binding. All Synopsys operators are associated with thoroughly tested golden bindings.

To check a binding for a particular operator, that operator must be annotated with a module and binding known to be correct. Do this using the **check_module** and **check_binding** attributes. For example:

```
operator (ADD_UNSP_OP) {    check_module : "add";  
    check_binding : "b1";  
    check_pin_widths : "A=1,B=1,Z=1; A=3,B=4,Z=5";  
    /* other declaration go here */
```

```
}
```

By default, each binding to an operator is verified for a number of different operator pin widths. The default pin widths are specified by a **check_pin_widths** attribute on the operator. The **check_pin_widths** attribute is a list of pin-width specifications, separated by semicolons (;). Each pin-width specification is a comma (,) separated list of pin sizes. Pin sizes are specified by a pin name, followed by an equals sign (=), followed by a pin width.

The preceding example runs one verification for operators with all pins of width 1 and a second verification with the pin A pin of width 3, pin B of width 4, and pin Z of width 5.

LIMITATIONS

Because **check_bindings** is based on **check_design**, it inherits the same limitations. The **check_bindings** command does not verify noncombinational parts unless the flip-flop names are identical. The **check_bindings** command also does not check very large multipliers because the verification is too expensive.

EXAMPLES

The following example checks the bindings to the SYNTH_ADD module:

```
prompt> check_bindings SYNTH_ADD
```

The next example checks binds b1 and b2 for specific pin widths. Note that **-pin_widths** lists are specified in the list format of dc_shell, not as a single string (as used in the previous **check_pin_widths** attribute).

```
prompt> check_bindings  
-binding {b1 b2}  
-pin_widths {"A=2,B=2,Z=2" "A=31,B=31,Z=32"}  
SYNTH_ADD
```

SEE ALSO

```
check_implementations(2)  
synthetic_library(3)
```

check_bsd

Checks whether a design's boundary-scan implementation is compliant with IEEE Std 1149.1.

SYNTAX

```
int check_bsd  
[-verbose true | false]  
[-effort low | medium | high]  
[-infer_instructions true | false]
```

ARGUMENTS

-verbose true | false

If -verbose true, generates multiple warning or error messages for a set of similar violations during a single step. By default, a warning or error message is generated for only the first violating cell or port.

-effort low | medium | high

Controls the effort used to search for implemented instructions. The time taken increases exponentially for a sequential search on instruction registers (IRs) of length 8 or more. The *low* effort uses only heuristics. The **high** effort enforces sequential search. The default **medium** effort first uses heuristics and then random opcode generation for limited iterations.

-infer_instructions true | false

When false, (default) the **check_bsd** command uses the implemented instructions and their opcode values from the boundary-scan circuitry inserted design. This information is passed by the **insert_dft** command during boundary-scan insertion. If this information is not available then the tool looks for the instructions and opcodes specified by the **set_bsd_instruction** command.

DESCRIPTION

Checks whether the current design's boundary-scan implementation is compliant with IEEE Std 1149.1. You must specify the Test Access Ports (TMS, TCK, TRST, TDI, and TDO) using the **set_dft_signal -view existing** command. If violations to any of the IEEE Std 1149.1 rules are found, the appropriate messages are generated. Check the design for any unresolved references before using this command. Set system clocks using the **set_dft_signal** command in order to see correct functionality. Specify any compliance enable ports using the **set_bsd_compliance** command.

Fatal error messages indicate that your design violates an IEEE Std 1149.1 rule and the violation is serious enough that the compliance checker can no longer continue analysis. You cannot use subsequent tools (for example, the BSDL generator) until you correct the violation.

Error messages indicate that your design violates an IEEE Std 1149.1 rule, but it is still possible to analyze further and gather more information about the design. However, you cannot use subsequent tools (for example, the BSDL generator) until you correct the violation.

Warning messages indicate that your design violates an IEEE Std 1149.1 rule, but the violation does not prevent the BSDL generator from generating BSDL.

You can shorten the sequential search space by masking bits within the IR instruction opcodes using the following two environment variables:

The **test_cc_ir_masked_bits** variable specifies the IR bits to be masked. The compliance checker masks the IR bits that contain "1" in the binary equivalent to the decimal integer value assigned to this variable. For details, see the man page for the **test_cc_ir_masked_bits** variable.

The **test_cc_ir_value_of_masked_bits** variable specifies a predefined value to be forced onto the masked IR bits. The decimal integer value assigned to this variable is converted to binary for the bit width of the IR, and the masked bits are forced with the corresponding values during the search operation. For details, see the man page for the **test_cc_ir_value_of_masked_bits** variable.

The least significant bit of the IR is considered to be the bit closest to the TDO port.

By default compliance check is done in accordance with the IEEE1149.1-2001 standard version. You can switch back to the old IEEE1149.1-1993 standard version, by setting the **-ieee1149.1_1993** option of the **set_bsd_configuration** command.

For more details about the IEEE Std 1149.1 rules, refer to the *EEE Std Test Access Port and Boundary-Scan Architecture*.

EXAMPLES

The following command performs compliance checking for the current design, and illustrates the types of messages that are output.

```
prompt> current_design my_boundary_scan
prompt> set_dft_signal -view existing -port my_tdi_port -type TDI
prompt> set_dft_signal -view existing -port my_tdo_port -type TDO
prompt> set_bsd_compliance -name p1 \
    -pattern {my_compliance_port, 0}
prompt> set_dft_signal -view spec -port my_system_clock \
    -type ScanMasterClock -timing {1 2}
prompt> check_bsd -v true
    Loading target library 'lsi_10k'
    Loading design 'my_boundary_scan'
Warning: Undriven input port TRST is floating. When undriven,
this port should behave as though it was driven by logic one. (TEST-819)
Warning: Undriven input port TMS is floating. When undriven,
this port should behave as though it was driven by logic one. (TEST-819)
Warning: Undriven input port TDI is floating. When undriven,
this port should behave as though it was driven by logic one. (TEST-819)
Error: Illegal capture value of BYPASS register. (TEST-881)
```

IEEE 1149.1 Summary

4 state elements found in the TAP controller
5 cells found in the Instruction Register
5 standard instructions found.
11 user defined instructions found.
1 cells in BYPASS register
32 cells in DEVICE ID register
5 cells in boundary-scan register

IEEE 1149.1 Violation Summary

6 Violations found in extraction of TAP Controller
Violates Rules: 3.3.1b 3.6.1b
Violates Rules: 3.3.1b 3.6.1b
Violates Rules: 3.3.1b 3.6.1b
1 Violations found in extraction of BYPASS register
Violates Rules: 9.1.1b

SEE ALSO

set_bsd_compliance(2)
set_dft_signal(2)

check_budget

Checks that user-specified budgets and fixed delays are consistent with path constraints.

SYNTAX

```
status check_budget
[-verbose]
[-tolerance tolerance]
[-from object_list]
[-to object_list]
[-no_interblock_logic]
[cell_list]
```

Data Types

<i>tolerance</i>	float
<i>object_list</i>	list
<i>cell_list</i>	list

ARGUMENTS

-verbose
Shows the path segments that cause violation of timing constraints. By default, only the number of paths that violate timing constraints are shown.

-tolerance *tolerance*
Specifies that only paths with a slack greater than *tolerance* in absolute value are checked and reported. By default, all paths are checked and reported.

-from *object_list*
Shows only the paths from the named pins, ports, or endpoints clocked by named clocks. By default, all paths that violate timing constraints are shown.

-to *object_list*
Shows only the paths to the named pins, ports, or endpoints clocked by named clocks. By default, all paths that violate timing constraints are shown.

-no_interblock_logic
Specifies that interblock logic is not to be budgeted. By default, Design Budgeter considers interblock logic not fixed.

cell_list
Specifies a list of cells to budget. This option must be set only when the **check_budget** command is used before the **allocate_budget** command.

DESCRIPTION

The **check_budget** command checks that the user-specified budgets and fixed delays are consistent with path constraints. Inconsistencies exist when the sum of user-

specified budgets and fixed delays along a path exceed the value of the path constraint. By default, the report shows only the number of paths that violate the timing constraints. The **-verbose** option shows details about the violations. The *cell_list* option allows you to check user budgets on the specified cells before budget allocation. This option must not be used after budget allocation.

EXAMPLES

The following example shows a report using only the default values.

```
prompt> check_budget
```

```
*****
Check budget
*****
Design has 2 violations.
1
```

The following example shows a report with the verbose option.

```
prompt> check_budget -verbose
```

```
*****
Check budget
*****

Startpoint: C3/OUT_reg (rising edge-triggered flip-flop clocked by clk)
Endpoint: C3/OUT_reg (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
clock network delay (ideal)	0.00	0.00
C3/OUT_reg/CLK (fdmf1i2)	0.00	0.00 r
C3/OUT_reg/Q (fdmf1i2)	0.00 *	0.00 r
C3/OUT_reg/D1 (fdmf1i2)	0.00	0.00 f
data arrival time		0.00
clock clk (rise edge)	500.00	500.00
clock network delay (ideal)	0.00	500.00
clock uncertainty	-20.00	480.00
C3/OUT_reg/CLK (fdmf1i2)	0.00	480.00 r
library setup time	-504.81	-24.81
data required time		-24.81
data required time		-24.81
data arrival time		0.00
slack (VIOLATED)		-24.81

```

Startpoint: IN (input port clocked by clk)
Endpoint: C1/rOUT_reg
```

(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
clock network delay (ideal)	0.00	0.00
input external delay	250.00	250.00 f
C1/IN (D1)	0.00	250.00 f
C1/rOUT_reg/D0 (fdmf1c2)	0.00	250.00 f
data arrival time		250.00
clock clk (rise edge)	500.00	500.00
clock network delay (ideal)	0.00	500.00
clock uncertainty	-20.00	480.00
C1/rOUT_reg/CLK (fdmf1c2)	0.00	480.00 r
library setup time	-342.00	138.00
data required time		138.00

data required time		138.00
data arrival time		-250.00

slack (VIOLATED)		-112.00

Design has 2 violations.

1

SEE ALSO

`report_path_budget(2)`

`set_user_budget(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 ... } ]
```

```

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

```

check_ccs_lib

```
set driver_model_only no
set debug_correlation yes
set lsf_correlation yes
run correlation
```

SEE ALSO

`check_library(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\(2\)](#)

```
compile(2)
current_design(2)
set_boundary_optimization(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'
```

```
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_implementations

Checks the implementations in a synthetic library module definition.

SYNTAX

```
integer check_implementations  
[-implementations implementation_list]  
[-parameters parameter_list]  
module_name
```

Data Types

<i>implementation_list</i>	list
<i>parameter_list</i>	list
<i>module_name</i>	string

ARGUMENTS

-implementations *implementation_list*
Lists the implementations of the module to check. The default is all implementations.

-parameters *parameter_list*
Lists the module parameters to use when checking. The default is to use the parameters specified by the **check_params** attribute of the module.

module_name
Specifies the name of the synthetic library module whose implementations you want to check. This parameter is required.

DESCRIPTION

This command ensures that when you add a new implementation to an existing module, the new implementation works in the same manner as the old implementation.

Check implementations are provided by Synopsys for all Synopsys modules. They are never used to implement hardware, only for verification.

```
module (mod1) {      /* Other declarations go here */  
    check_implementation(check_add) {}  
    /* other implementations go here */  
}
```

By default, every implementation of a module is compared to the verify implementation. For parameterized modules, a verification is performed for each of the **check_params** that is specified for the module.

The **check_params** attribute is a list of parameter specifications, separated by semicolons (;), which specifies a test to be run. Each parameter specification is a comma (,), separated list of parameter names, followed by an equals sign (=),

followed by a value. For example, the specification below runs one verification with n=2 and m=3, and a second verification with n=5 and m=8 for each implementation:

```
<examplelast>
check_params : "n=2,m=3; n=5,m=8" ;
<body>

check_implementations -impl {rpl cla}add
```

LIMITATIONS

Because **check_implementations** is based on **check_design**, it inherits the same limitations. The **check_implementations** command does not verify non-combinational parts unless the flip-flop names are identical. The **check_implementations** command also does not check very large multipliers because the verification is too expensive.

EXAMPLES

The following example checks all implementations of the SYNTH_ADD module:

```
prompt> check_implementations SYNTH_ADD
```

This example checks the ripple and carry-lookahead implementations for parameter values of 33 and 63:

```
prompt> check_implementations -impl {rpl_add cla_add}

-param {"n=33" "n=63"} SYNTH_ADD
```

SEE ALSO

[check_bindings\(2\)](#)
[synthetic_library\(3\)](#)

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`

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_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.

check_mv_design

`-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)
compile(2)
remove_target_library_subset(2)

check_mv_design

```
set_target_library_subset(2)
```

check_rp_groups

Checks the relative placement constraints and reports the failures.

SYNTAX

```
collection check_rp_groups
{rp_groups | -all}
[-output filename]
```

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.

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_def

Allows scan chain structural consistency checking based on the scan chain information either stored in the current .ddc file as an attribute or an ASCII SCANDEF file.

SYNTAX

```
status check_scan_def
[-file file_name]
```

DESCRIPTION

The **check_scan_def** command performs a scan chain structural checking based on the scan chain information stored in the current DDC input. If the scan chain passes all structural checks, then the scan chain status will be "VALIDATED"/V. If the scan chain fails the structural checks, then the scan chain status will be "FAILED"/F.

This command is used in the dc_shell-based environment. This command presently does not support hierarchical flows where black box subdesigns with CTL model (ctlddc based flows) are used. If used in such flows, the **check_scan_def** command shows FAILED for all scan chains that have elements from black box subdesigns.

EXAMPLES

The following is an example of the result of **check_scan_def** command when no file is specified:

```
prompt> check_scan_def
Checking SCANDEF...
  Checking scan cell correspondance between SCANDEF and netlist...
  Checking scan chain checking...
All checks completed.
```

```
*****
```

```
Report : Scan DEF check
Design : design name
Version: software version
Date   : current date
```

```
*****
```

```
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 shows the result when a file is specified with the **check_scan_def** command:

```
prompt> check_scan_chain -file temp.def Reading DEF file temp.def Checking
SCANDEF... Checking scan cell correspondance between SCANDEF and netlist... Checking
scan chain checking... All checks completed.
```

```
*****
```

```
Report : Scan DEF check Design : design name Version: software version Date :
current date
```

```
*****
```

```
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

SEE ALSO

```
report_scan_chain(2)
write_scan_def(2)
```

check_synlib

Performs semantic checks on synthetic libraries.

SYNTAX

```
int check_synlib
```

ARGUMENTS

None.

DESCRIPTION

Synthetic libraries can contain errors that cannot be detected using the **read_lib** command. The **check_synlib** command performs checking on synthetic libraries found in the synthetic library search path (**synthetic_library**) and listed by the **link_library** variable. Note that **check_synlib** also implicitly checks against standard.sldb.

The **check_synlib** command returns one of the following two values:

```
0 : Errors were reported.  
1 : No critical errors were found.
```

Ideally, you should run **check_synlib** after you set the **synthetic_library** variable and the .sldb files exist, yet before commands such as **compile** or **replace_synthetic** use any of these libraries.

EXAMPLES

The following example checks for compatibility the synthetic libraries named in the **synthetic_library** variable. The two libraries are my_synlib_0.sldb and my_synlib_1.sldb.

```
prompt> synthetic_library = {"my_synlib_0.sldb", "my_synlib_1.sldb"}  
prompt> check_synlib
```

SEE ALSO

```
compile(2)  
read_lib(2)  
replace_synthetic(2)  
report_synlib(2)  
synthetic_library(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)  
compile(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)
compile(2)
compile_ultra(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)
```


clean_buffer_tree

Removes the buffer tree at the specified driver pin on a mapped design.

SYNTAX

```
int clean_buffer_tree  
[-from start_point_list]  
[-to end_point_list]  
[-net net_list]  
[-hierarchy]  
[-global]  
[-threshold integer_in_1]
```

Data Types

<i>start_point_list</i>	list
<i>end_point_list</i>	list
<i>net_list</i>	list
<i>integer_in_1</i>	integer

ARGUMENTS

-from *start_point_list*

Specifies the starting point of the buffer tree to remove. The *start_point_list* must contain only pins or ports. This option can be used in conjunction with the **-to** and **-net** options.

-to *end_point_list*

Specifies the endpoint of the buffer tree to remove. The buffer tree removal starts at the immediate buffer source of the specified pins or ports. The *end_point_list* must contain only pins or ports. This option can be used in conjunction with the **-from** and **-net** options.

-net *net_list*

Specifies the net where the driver is the starting point of the buffer tree removal. The *net_list* must contain only nets. This option can be used in conjunction with the **-from** and **-to** options.

-hierarchy

Removes the buffer tree across the hierarchy. By default, the buffer tree removal stops when it reaches the hierarchy boundary.

-global

Searches for high-fanout buffer tree root pins in the *net_list*. This option is only valid with the **-from**, **-to**, and **-net** options.

-threshold *integer_in_1*

Specifies the fanout threshold range that determines which buffer trees are cleaned up. If the number of sink pins driven from the primary root pin of the buffer tree is larger than the threshold, the buffer tree circuitry is cleaned up.

Valid **-threshold** values are from 1 to 1,000,000.

DESCRIPTION

The **clean_buffer_tree** command removes a buffer tree at a specified driver pin, load pin, or driver net on a mapped design. When given a driver pin or driver net, the buffer tree is removed with this driver as the root. When given a load pin, the buffer tree is removed with the immediate driver of the load pin as the root. By default, this command does not remove the buffer tree across the hierarchy. To enable buffer tree removal across boundaries, specify the **-hierarchy** option when running the command.

The buffer tree removal process starts at the source and removes all buffer and inverter cells in its transitive fanout cone until it finds nonbuffer cells or hierarchy boundaries. This does not apply if the **-hierarchy** option is specified.

Buffer and inverter cells that have the **dont_touch** attribute or that are connected to a net that has the **dont_touch** attribute are not removed, nor are any cells that follow these cells in the transitive fanout removed.

The **clean_buffer_tree** command adds an inverter driving all of the original buffer tree's inverted loads after it has removed all buffers and inverters in the tree.

This command accepts the final implementation of buffer tree removal, even if negatively impacts the timing or DRC violations in the design.

EXAMPLES

The following example removes a buffer tree without crossing hierarchical boundaries:

```
prompt> clean_buffer_tree -from cell1/O
```

The following example removes a buffer tree that crosses hierarchical boundaries:

```
prompt> clean_buffer_tree -hierarchy -from cell1/O
```

SEE ALSO

```
all_fanout(2)
balance_buffer(2)
report_buffer_tree(2)
report_cell(2)
report_constraint(2)
```

close_mw_lib

Closes the current Milkyway library.

SYNTAX

```
status close_mw_lib
```

ARGUMENTS

The **close_mw_lib** command has no arguments.

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).

```


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)
```

compile

Performs logic-level and gate-level synthesis and optimization on the current design.

SYNTAX

```
status compile
[-no_map]
[-map_effort medium | high]
[-area_effort none | low | medium | high]
[-incremental_mapping]
[-exact_map]
[-ungroup_all]
[-boundary_optimization]
[-auto_ungroup area | delay]
[-no_design_rule | -only_design_rule | -only_hold_time]
[-scan]
[-top]
[-power_effort none | low | medium | high]
[-gate_clock]
```

ARGUMENTS

-no_map

Specifies not to map the current design into the target technology library. With this option, generic Boolean equations and generic flip-flops represent the resulting design.

-map_effort medium | high

Specifies the relative amount of CPU time spent during the mapping phase of **compile**. Valid values are **medium**, and **high**. You can select one value. The default is **medium**.

-area_effort none | low | medium | high

Specifies the relative amount of CPU time spent during the area recovery phase of **compile**. Valid values are **none**, **low**, **medium**, and **high**. You can select one value. The default is the specified *map_effort*.

-incremental_mapping

Specifies to attempt only incremental improvements to the gate structure of a design. Portions of a design that are already mapped are exempt from logic level optimization, and the resulting design should be the same (if no improvements can be made) or better in terms of its design constraints. Implementations for DesignWare operators are reselected in an incremental compile run if the swap can improve the Optimization Cost based on the Optimization constraints on the design.

-exact_map

Specifies that the sequential elements in the final design must exactly match the descriptions specified in the HDL. SEQGEN is a technology-independent representation of a sequential element that is inferred by the HDL compiler from HDL descriptions of sequential circuits. When you specify this option,

the tool attempts to find sequential elements in the target technology library that match the behavior of only the SEQGEN element (that is, no surrounding logic is considered). In the event that an exact match for a SEQGEN is not available, a different sequential element is used. For improved sequential inference, use this option in conjunction with the HDL directives **sync_set_reset**, **sync_set_reset_local**, and **sync_set_reset_all**. The new attributes or directives tell HDL Compiler how to connect nets to the SEQGEN component. Use of the attributes/directives with the **-exact_map** option ensures that the results of **compile** are predictable.

-ungroup_all

Collapses all levels of hierarchy in a design, except those that have the **dont_touch** attribute set.

-boundary_optimization

Optimizes across all hierarchical boundaries in the design. This can change the function of the design such that it can operate only in its current environment. If input or output ports are complemented as a result of using this option, port names are changed according to the **port_complement_naming_style** variable.

-auto_ungroup area | delay

Automatically ungroups hierarchies in the design. A hierarchy will be considered for auto_ungrouping only if it satisfies all of the following criteria:

- It does not have the **dont_touch** or **ungroup** attribute set on it.
- There are no timing constraints or exceptions set on its pins.
- Its wire load model is the same as that of its parent, or **compile_auto_ungroup_override_wlm** is set to **true**.

- It has fewer child cells than the value of the **compile_auto_ungroup_area_num_cells** variable if you select **area**.

If you select **area**, all hierarchies that meet the above criteria will be ungrouped. The **-auto_ungroup area** option is generally used to ungroup small hierarchies in the design to improve area recovery. It does not have a significant impact on the timing of the design.

The **-auto_ungroup delay** option employs an intelligent ungrouping strategy that attempts to improve the overall timing of the design. It chooses hierarchies that are most likely to benefit from the extra boundary optimizations that ungrouping exposes. The algorithm emphasizes hierarchies containing paths that are either critical, or likely to become critical, after subsequent optimization steps.

Delay-driven auto-ungrouping offers a less CPU-intensive alternative to using the **-ungroup_all** option for improving design timing.

-no_design_rule

Determines, along with the **-only_design_rule** and **-only_hold_time** options, whether the command fixes design rule violations before exiting. The **-no_design_rule** option specifies for the command to exit before fixing design rule violations, thus allowing you to check the results in a constraint report before fixing the violations. The **-no_design_rule**, **-only_design_rule**, and **-only_hold_time** options are mutually exclusive. You can select only one. The

default is to perform both design rule fixing and mapping optimizations before exiting.

-only_design_rule

Determines, along with the **-no_design_rule** and **-only_hold_time** options, whether the command fixes design rule violations before exiting. The **-only_design_rule** option specifies for the command to perform only design rule fixing; that is, mapping optimizations are not performed. The **-no_design_rule**, **-only_design_rule**, and **-only_hold_time** options are mutually exclusive. You can select only one. The default is to perform both design rule fixing and mapping optimizations before exiting.

-only_hold_time

Determines, along with the **-no_design_rule** and **-only_design_rule** options, whether the command fixes design rule violations before exiting. The **-only_hold_time** option specifies for the command to perform only hold time fixing, ignoring other design rules. The **set_fix_hold** command must be specified for hold time fixing to be performed. The **-no_design_rule**, **-only_design_rule**, and **-only_hold_time** options are mutually exclusive. You can select only one. The default is to perform both design rule fixing and mapping optimizations before exiting.

-scan

Specifies that the command is to consider the impact of scan insertion on mission mode constraints during optimization. This option causes the command to replace all sequential elements during optimization. Some scan-replaced sequential cells might be converted to nonscan cells later in the test synthesis process because of test design rule violations or explicit user specifications. By accounting for the impact of internal scan insertion from the start of the design process, Test Ready Compile eliminates the need for future reoptimization.

-top

Fixes design rule and top-level timing violations for a design, but does not perform any mapping on the design. By default, this option fixes all design rule violations, but only those timing violations whose paths cross top-level hierarchical boundaries. If you want this option to fix timing violations for all paths, set the **compile_top_all_paths** variable to **true**.

-power_effort none | low | medium | high

Specifies the relative amount of CPU time spent during the power optimization phase of **compile**. Valid values are **none**, **low**, **medium**, and **high**. You can select one value. The default is the specified *map_effort*. Since power optimization in **compile** is enabled by power constraints, this option will be ignored if there is no power constraint on the design.

-gate_clock

Enables clock gating optimization: clock gates are automatically inserted or removed. If the variable **power_driven_clock_gating** is set to true, the optimization is based on the switching activity and dynamic power of the register banks. The **-gate_clock** option cannot be used in combination with the **-only_design_rule** option. When used in combination with the **-exact_map** option, it may not be possible to honor the **-exact_map** option for those registers that are involved with clock gating optimization.

DESCRIPTION

The **compile** command performs logic and gate-level synthesis and optimization on the current design. Optimization is controlled by user-specified constraints on the design. These constraints describe goals for the optimization process, such as make the smallest circuit possible, or try to make specified outputs arrive by a specified time. The optimization process trades off timing and area constraints to provide the smallest possible circuit that meets specified timing requirements. Values for the area and speed of components used in synthesizing and optimizing the design are obtained from user-specified libraries.

Constraints fall into one of two categories: Design Rule Constraints (DRC) and optimization constraints. Design Rule Constraints reflect technology-specific restrictions that must be met for a design to function correctly. Optimization constraints reflect less critical design goals and restrictions that are desirable, but not crucial for the operation of a design.

The Design Rule Constraints are **max_transition**, **max_fanout**, **max_capacitance**, and **min_capacitance**. All other constraints are optimization constraints. Examples include maximum delay and maximum area.

During optimization, both types of constraints are considered, but precedence is given to meeting Design Rule Constraints. The **min_delay** and **fix_hold** constraints are treated similarly to Design Rule Constraints except that they have lower priority than the maximum delay constraints. The priority given to various constraints can be modified by using the **set_cost_priority** command.

Often when a design read from an HDL description is optimized, new designs modeled from the synthetic library are generated. These designs are named according to the style specified in the **synthetic_design_naming_style** variable. For details, see the **compile_variables** man page.

When the current design is hierarchical, and there are multiple design instances of a subdesign, it is no longer necessary to run the **uniquify** command before running the **compile** command. The **uniquify** command can still be used on multiple design instances so that they can be individually optimized. For all designs marked with the **uniquify** attribute, **compile** copies and renames them according to the **uniquify_naming_style** variable.

The **compile** command does not optimize across hierarchical boundaries unless the **boundary_optimization** attribute is **true**. All synthetically generated designs are created with this flag set to **true**. If boundary optimization causes ports to be complemented, port names are changed according to the **port_complement_naming_style** variable.

To customize the way the **compile** command optimizes subdesigns, use compile directives. Commands for placing these directives include **set_flatten** and **set_structure**. Command line options pass global directives that affect how the entire design is optimized to the **compile** command.

If the current design or any of its subdesigns, is represented as a state table, the **compile** command automatically performs finite-state machine optimizations on these designs. Finite-state machine synthesis is also controlled with compile directives placed directly on these designs by using commands such as **set_fsm_encoding** and **set_fsm_minimize**.

If the current design or any of its subdesigns contains arithmetic operations (additions, subtractions, multiplications and comparisons), the **compile** command automatically transforms them into datapath blocks to be implemented by a datapath generator. Datapath optimization can be controlled by using the **hlo_disable_datapath_optimization** variable.

The **compile** command reports progress in real time by displaying a report. During optimization, the report shows incremental results each time a transformation is applied to the design. The default fields of the report are ELAPSED TIME, AREA, WORST NEG SLACK, TOTAL NEG SLACK, DESIGN RULE COST, and ENDPOINT.

ELAPSED TIME

Tracks the elapsed time since the beginning of the current **compile** or **reoptimize_design**.

AREA

Shows the area of the design during the optimization.

WORST NEG SLACK

Shows the worst negative slack (**max_path** violation) in all path groups.

TOTAL NEG

Shows the sum of the negative slack across all endpoints in the design.

DESIGN RULE COST

Measures the distance between the actual results and user-specified design rule constraints.

ENDPOINT

Shows the endpoint currently being worked on. When a delay violation is being fixed, the object for the **ENDPOINT** is a pin or a port. When a design rule violation is being fixed, the object for the **ENDPOINT** is a net.

You can specify other fields in addition to or instead of any of the default fields. For a list of available fields and other details about specifying the compile log format, see the **compile_log_format** variable man page.

To display the same log format as the 1998.02 version, set **compile_log_format** = "". The fields for the 1998.02 version are **TRIALS**, **AREA**, **DELTA DELAY**, **TOTAL NEGATIVE SLACK**, and **DESIGN RULE COST**. The new default does not have the **TRIALS** and **DELTA DELAY** fields.

TRIALS

Tracks the number of transformations that the optimizer tries before making the current selection.

DELTA DELAY

Shows the current max delay cost of the design, which is the sum of the worst negative slack (**max_path** violation) in each path group.

The log written by **compile** shows the different phases of optimization. In addition to the Resource Allocation and Mapping phases, there are four more phases. The first of these is the Delay Optimization phase, which fixes **max_path** violations. In the Phase 1 Design Rule Fixing Phase, design rule violations are fixed without creating any new **max_path** delay violations. In the Phase 2 Design Rule Fixing Phase, **max_path** delay constraints might be impacted while fixing design rule violations. Finally, there is an Area Recovery phase that attempts to reduce the area of the design while

keeping other constraints intact. The Area Recovery phase always does some minimal cleanup, but performs more CPU-intensive area recovery only if `max_area` constraints have been set on the design. If the **-only_design_rule** option is used, Delay Optimization Phase and Area Recovery Phase are skipped. If the **-no_design_rule** option is used, both Phase 1 Design Rule Fixing and Phase 2 Design Rule Fixing are skipped. If the **set_cost_priority** command with the **-delay** option has been used before using the **compile** command, Phase 2 Design Rule Fixing is skipped.

The **set_local_link_library** command sets the **local_link_library** attribute on a design. The **local_link_library** attribute contains a list of design and library files that are searched before the files are specified with the **link_library** variable whenever a link operation is performed. The **target_library** variable specifies a technology library or a list of technology libraries containing the components (usually from a specific ASIC supplier) to use during optimization. The **compile** command sets the **local_link_library** attribute of the top-level design to the value of the **target_library** variable.

If **compile** is interrupted by an interrupt signal during an interactive process, it displays the following menu:

```
Please type in one of the following option:
  1 to Write out the current state of the design
  2 to Abort optimization
  3 to Kill the process
  4 to Continue optimization
Please enter a number:
```

If you select option 1, the design is written out and the menu reappears.

If the process is running in the background, you cannot use the menu. The number of interrupt signals determines what action is performed. The sequence is as follows:

```
^C Information: Preparing to interrupt optimization... (INT-5)
^C Information: Aborting Optimization... (INT-6)
^C Information: Process terminated by interrupt. (INT-4)
```

The interrupt signal can be different from one machine to another.

EXAMPLES

The following examples apply to all modes.

The following command specifies that the current design be optimized without mapping the logic:

```
prompt> compile -no_map
```

The following command optimizes the design TEST twice; once for speed and once for area.

```
prompt> current_design TEST
```

```
prompt> set_max_delay 2.0 -to [all_outputs]
```



```
prompt> compile /usr/path/FAST
```

```
prompt> set_max_area 250
```

```
prompt> compile /usr/path/SMALL
```

The following example specifies that the command perform restricted optimizations across hierarchical boundaries:

```
prompt> compile -boundary_optimization
```

The following example demonstrates a simple optimization strategy. The commands in the example first define the current design. Design attributes and constraints are set. Optimization phases for **compile** are defined. The **compile** command is executed with the specified options.

```
prompt> current_design TRIAL  
prompt> set_wire_load_model "10x10"  
prompt> set_operating_conditions "WCCOM"  
prompt> create_clock -period 12 \  
-waveform {0 6} CLK  
prompt> set_input_delay 2 [all_inputs]  
prompt> set_output_delay 3 [all_outputs]  
prompt> set_max_area 580  
prompt> set_structure -boolean false  
prompt> set_flatten -phase true -effort medium  
prompt> compile -incremental_mapping \  
-no_design_rule  
prompt> report_constraint
```

SEE ALSO

```
alib_analyze_libs(2)  
alib_library_analysis_path(3)  
check_design(2)  
create_clock(2)  
report_compile_options(2)  
reset_design(2)  
set_boundary_optimization(2)  
set_cost_priority(2)  
set_critical_range(2)  
set_dont_touch(2)  
set_dont_touch_network(2)  
set_dont_use(2)  
set_drive(2)  
set_equal(2)  
set_fix_hold(2)  
set_fix_multiple_port_nets(2)  
set_flatten(2)  
set_fsm_encoding(2)  
set_fsm_encoding_style(2)  
set_fsm_order(2)  
set_fsm_state_vector(2)  
set_input_delay(2)  
set_load(2)
```

```
set_local_link_library(2)
set_logic_dc(2)
set_logic_one(2)
set_logic_zero(2)
set_max_area(2)
set_max_capacitance(2)
set_max_delay(2)
set_max_fanout(2)
set_max_transition(2)
set_min_delay(2)
set_operating_conditions(2)
set_opposite(2)
set_output_delay(2)
set_prefer(2)
set_register_type(2)
set_structure(2)
set_timing_ranges(2)
set_unconnected(2)
set_wire_load_mode(2)
set_wire_load_model(2)
set_wire_load_min_block_size(2)
set_wire_load_selection_group(2)
ungroup(2)
uniquify(2)
compile_auto_ungroup_area_num_cells(3)
compile_auto_ungroup_count_leaf_cells(3)
compile_auto_ungroup_override_wlm(3)
compile_log_format(3)
compile_top_all_paths(3)
ungroup_keep_original_design(3)
```

compile_partitions

Distributes compile jobs for a design.

SYNTAX

```
status compile_partitions  
-destination pass
```

Data Types

pass string

ARGUMENTS

-destination *pass*
 Specifies which Automated Chip Synthesis pass to use for compile jobs. The command uses the *pass* value to generate the subdirectory name.

DESCRIPTION

Distributes the compile jobs for a design by using the specified Automated Chip Synthesis destination pass directory. The command compiles the design by using the makefile located in *passn*/Makefile. The command uses the make utility tool specified by the **acs_make_exec** variable with optional arguments specified by using the **acs_make_args** variable. The **acs_num_parallel_jobs** variable specifies the number of compile jobs to run in parallel. The command saves a log of the run in the *passn*/logs/\$*acs_make_exec*.log file.

When using the Load Sharing Facility (LSF) software, you must define the root of the LSF installation location by using the **LSF_TOP** UNIX variable. This command requires that the LSF executable be located in \$*LSF_TOP*/\$*sh_arch*/bin/lsmake.

SEE ALSO

```
write_makefile(2)  
acs_make_args(3)  
acs_make_exec(3)  
acs_num_parallel_jobs(3)  
sh_arch(3)
```

compile_ultra

Performs a high-effort compile on the current design for better quality of results (QoR).

SYNTAX

```
status compile_ultra
[-incremental]
[-scan]
[-exact_map]
[-no_autoungroup]
[-no_seq_output_inversion]
[-no_boundary_optimization]
[-no_design_rule | -only_design_rule]
[-timing_high_effort_script | -area_high_effort_script]
[-top]
[-retime]
[-gate_clock]
[-check_only]
[-num_cpus n]
[-congestion]
```

Data Types

n integer

ARGUMENTS

- incremental**
Runs **compile_ultra** in incremental mode. In the incremental mode the tool does not run mapping or implementation selection stages.
- scan**
Enables the examination of the impact of scan insertion on mission mode constraints during optimization, as in a normal compile. Use this option to replace all sequential elements during optimization. Some scan-replaced sequential cells may be converted to nonscan cells later in the test synthesis process because of test design rule violations or explicit user specifications.
- exact_map**
Specifies that sequential cells are mapped exactly as indicated in HDL.
- no_autoungroup**
Specifies that automatic ungrouping is completely disabled. All user hierarchies are preserved unless otherwise specified.
- no_seq_output_inversion**
Disables sequential output inversion. The phase sequential of all sequential elements will be the same as the RTL. Without this option, **compile_ultra** is free to invert sequential elements during mapping and optimization. For more information, see the man page for the **compile_seqmap_enable_output_inversion**

variable.

-no_boundary_optimization

Specifies that no hierarchical boundary optimization is to be performed. By default, boundary optimization is turned on during **compile_ultra** activity.

-no_design_rule

Determines whether the command fixes design rule violations before exiting. The **-no_design_rule** option specifies for the command to exit before fixing design rule violations, thus allowing you to check the results in a constraint report before fixing the violations. The default is to perform both design rule fixing and mapping optimizations before exiting.

The **-no_design_rule** and **-only_design_rule** options are mutually exclusive. Use only one option.

-only_design_rule

Determines whether the command fixes design rule violations before exiting. The **-only_design_rule** option specifies for the command to perform only design rule fixing; that is, mapping optimizations are not performed. The default is to perform both design rule fixing and mapping optimizations before exiting.

The **-no_design_rule** and **-only_design_rule** options are mutually exclusive. Use only one option. The **-only_design_rule** option can be used only with the **-incremental** option.

-timing_high_effort_script

Runs a strategy intended to improve the resulting delay of the design, possibly at the cost of additional runtime. The strategy may make changes to variables or constraints that modify **compile_ultra** behavior and perform additional passes to achieve better delay.

-area_high_effort_script

Runs a strategy intended to improve the resulting area of the design, possibly at the cost of additional runtime. The strategy may make changes to variables or constraints that modify **compile_ultra** behavior and perform additional passes to achieve better area.

-top

Fixes design rule and top-level timing violations in a design. By default, this option fixes all design rule violations, but only those timing violations whose paths cross top-level hierarchical boundaries. If you want this option to fix timing violations for all paths, set the **compile_top_all_paths** variable to true.

-retime

Uses the adaptive retiming algorithm during optimization to improve delay. This option is ignored if the **-only_design_rule**, **-incremental**, or **-top** options are chosen at the same time.

-gate_clock

Enables clock gating optimization: clock gates are automatically inserted or removed. If the **power_driven_clock_gating** variable is set to true, the optimization is based on the switching activity and dynamic power of the register banks. The **-gate_clock** option cannot be used in combination with the **-only_design_rule** option. When used with the **-exact_map** option, it may not

be possible to honor the **-exact_map** option for those registers that are involved with clock gating optimization.

-check_only

Checks whether the design and libraries have all of the data that **compile_ultra** requires to run successfully. This option is available only in Design Compiler topographical mode.

-num_cpus n

This option is obsolete and will be removed in an upcoming release. Use **set_host_options -max_cores n** instead.

-congestion

Enables Design Compiler features to optimize for reduced routing-related congestion. This option is only supported in Design Compiler topographical mode.

DESCRIPTION

The **compile_ultra** command performs a high-effort compile on the current design for better quality of results (QoR). Like the **compile** command, optimization is controlled by constraints that you specify on the design. This command is targeted towards high performance designs with very tight timing constraints. It provides you with a simple push-button approach to achieve critical delay optimization. The **compile_ultra** command packages all of the Design Compiler Ultra features and have them on by default. It requires a Design Compiler Ultra license, plus a DesignWare Foundation license. This command provides the best strategy for optimum overall QoR and performance.

This command can be used in the same manner as the **compile** command.

By default, **compile_ultra** incorporates two ungrouping phases for user design hierarchies. The first phase is performed before "Pass1 Mapping" and attempts to ungroup small design hierarchies. This first ungrouping phase can be turned off using the following command:

```
set compile_ultra_ungroup_small_hierarchies false
```

The second ungrouping phase is performed during "Mapping Optimization" and applies a delay-based ungrouping strategy for user design hierarchies. You can set variables to control the second ungrouping phase in the same manner as with **compile -auto_ungroup_delay**; for example, with the **compile_auto_ungroup_delay_num_cells** variable. If you need to preserve all user design hierarchies, use the **-no_autoungroup** option.

By default, if **dw_foundation.sldb** is not in the **synthetic_library** list, and the DesignWare license is successfully checked out, **dw_foundation.sldb** is automatically added to the **synthetic_library** to utilize the QoR benefit provided by the licensed DesignWare architectures. This behavior occurs in the current command only, and it does not affect the user-specified **synthetic_library** and **link_library** list.

By default, all DesignWare hierarchies are unconditionally ungrouped in the second pass of compile. You can set the **compile_ultra_ungroup_dw** variable to control the

ungrouping process of DesignWare components.

By default, hierarchical boundary optimization is performed on the design. This can change the function of the design so that it can operate only in its current environment. If input or output ports are complemented as a result of this optimization, port names are changed according to the **port_complement_naming_style** variable. Use the **-no_boundary_optimization** option to turn off the boundary optimization feature.

Regardless of the options used, **compile_ultra** sets the value for the following environment variables:

```
hlo_resource_allocation = constraint_driven
hlo_resource_implementation = use_fastest
hlo_minimize_tree_delay = true
compile_use_low_timing_effort = false
compile_implementation_selection = true
```

The **-area_high_effort_script** and **-timing_high_effort_script** options run prepared scripts intended to improve the area or delay of the design. The scripts apply a compile strategy that may turn on or off different optimization features depending on the optimization goal, and may make temporary changes to optimization constraints. Therefore, using these scripts may override additional variables not mentioned in the previous paragraph. Some variable settings may persist after the **compile_ultra** command completes, so that subsequent incremental compiles run with the same settings. Because the scripts may perform additional compile passes, they may increase the runtime of the **compile_ultra** command. These two options cannot be used with the **-incremental**, **-no_design_rule**, or **-only_design_rule** option.

EXAMPLES

The following example runs the command with the first ungrouping phase for small user design hierarchies turned off and with changed setting for the second, delay-based automatic ungrouping phase.

```
prompt> set compile_ultra_ungroup_small_hierarchies false
prompt> set compile_auto_ungroup_delay_num_cells 300

prompt> compile_ultra
```

SEE ALSO

```
compile(2)
set_host_options(2)
```

connect_logic_one

Connects the logic one net to a load pin on a cell instance.

SYNTAX

```
status connect_logic_one
[-net_name net_name]
[load_pin_list]
[-reconnect_all]
[-mode power_intention | power_connection]
[-empty]
```

Data Types

<i>net_name</i>	string
<i>load_pin_list</i>	list

ARGUMENTS

-net_name *net_name*

Specifies the name of the Milkyway net that should be connected to the load pins. The net must be a scalar (single bit) net, it must be a power net, and it must exist in the Milkyway design. If this option is not specified, the **mw_logic1_net** variable is used to get the name of the net.

load_pin_list

Specifies a list of load pins on leaf cell instances to which the net is to be connected. The pins must be on cell instances that are linked to the link or target libraries, not to ports or hierarchical cell instances. The load pins may be at different levels of hierarchy. If a specified pin is already connected, the tool issues an error message. This argument is required unless **-reconnect_all** is used.

-reconnect_all

Reconnects all existing tie-high pins to proper power nets consistent with their cell instances' power connections in multivoltage designs. The power nets will be determined depending on the reconnection mode specified in **-mode**.

This option is off by default and cannot be used with any option other than **-mode** of the command.

-mode *power_intention* | *power_connection*

Specifies the mode for tie-off reconnection mode. The default mode is *power_intention*.

In *power_intention* mode, the design is required to have UPF or non-UPF power intention. For a given tie-high pin, the tool first identifies the related power pin defined in the logical lib_cell. The reconnection for the tie-high pin then utilizes the power net of this related power pin specified in the power intention. If logical lib_cells without power and ground pins are used in non-UPF mode, the tie-high pin of a regular cell will be connected to the primary power net of the cell's power domain, while the backup power net will be used for the tie-high pin on an always-on cell.

In power_connection mode, the reconnection follows the power connections in the database. For a tie-high pin, the power net connected to its related power pin is used for the reconnection if its cell is linked to a logical lib_cell with power and ground pins. If the logical lib_cell has no power and ground pin, the reconnection will work only if the cell's power pins are connected to one power net.

The option can only be used together with **-reconnect_all**.

-empty

Disconnects load pins on leaf cell instances and ports.

This option cannot be used with **-reconnect_all**.

DESCRIPTION

This command connects a logic one net to load pins in the current design and stores the connection persistently in the Milkyway database, as done in the **connect_pg_nets** command.

EXAMPLES

The following example uses the **connect_logic_one** to connect VDD to load pin U7/A. The **verbose_messages** variable is true, so that message follows the command. The **all_connected** command is then used to verify the Milkyway net connection.

```
prompt> connect_logic_one -net_name VDD [get_pin U7/A]
Connecting power net 'VDD' to pin 'U7/A'.
all_connected [get_pin U7/A]
{VDD}
```

The following example shows the error message generated when a pin is already connected:

```
prompt> connect_logic_one -net_name VDD [get_pin U7/A]
Object 'U7/A' is already connected to net 'signal25'. (UIED-21)
```

SEE ALSO

```
all_connected(2)
connect_logic_zero(2)
connect_power_domain(2)
create_net(2)
create_power_domain(2)
create_power_net_info(2)
current_design(2)
disconnect_net(2)
remove_net(2)
```

connect_logic_zero

Connects the logic zero net to a load pin on a cell instance.

SYNTAX

```
status connect_logic_zero
[-net_name net_name]
[load_pin_list]
[-reconnect_all]
[-empty]
[-mode power_intention | power_connection]
```

Data Types

<i>net_name</i>	string
<i>load_pin_list</i>	list

ARGUMENTS

-net_name *net_name*

Specifies the name of the Milkyway net that should be connected to the load pins. The net must be a scalar (single bit) net and a ground net, and must exist in the MW CEL. By default, the command uses the **mw_logic0_net** variable to get the name of the net.

load_pin_list

Specifies a list of load pins on leaf cell instances to which the net is to be connected. The pins must be on cell instances that are linked to the link or target libraries, not ports or hierarchical cell instances. The load pins can be at different levels of hierarchy. If a specified pin is already connected, the tool issues an error message. This argument is required unless you use the **-reconnect_all** option.

-reconnect_all

Allows you to reconnect all existing tie-low pins to proper ground nets consistent with their cell instances' power connections in multivoltage designs. The option requires power domains defined in the design. The proper ground net for a given tie-high pin is determined in the following manner: a tie-low pin follows the connection of its related ground pin if the `related_ground_pin` attribute is defined on the pin in the `lib_cell`; otherwise, the tie-low pin of a regular cell is connected to the primary ground net of the cell's power domain, while the backup ground net is used for the tie-low pin of an always-on cell. The **-reconnect_all** and **-empty** options are mutually exclusive; you can use only one. This option is mutually exclusive with all other options.

-empty

Allows you to disconnect load pins on leaf cell instances and ports. The **-reconnect_all** and **-empty** options are mutually exclusive; you can use only one.

DESCRIPTION

This command connects a logic zero net to load pins in the current design and stores the connection persistently in the Milkyway database, as is done in the **connect_pg_nets** command.

EXAMPLES

The following example uses the **connect_logic_zero** command to connect *VSS* to the load pin *U7/A*. The **verbose_messages** variable is **TRUE**, so that message follows the command. The **all_connected** command is then used to verify the Milkyway net connection:

```
prompt> connect_logic_zero -net_name VSS [get_pin U7/A]
Connecting ground net 'VSS' to pin 'U7/A'.
prompt> all_connected [get_pin U7/A]
{VSS}
```

The following example shows the error message generated when a pin is already connected:

```
prompt> connect_logic_zero -net_name VSS [get_pin U7/A]
Object 'U7/A' is already connected to net 'signal25'. (UIED-21)
```

SEE ALSO

```
all_connected(2)
connect_logic_one(2)
connect_power_domain(2)
create_net(2)
create_power_domain(2)
create_power_net_info(2)
current_design(2)
disconnect_net(2)
remove_net(2)
mw_logic0_net(3)
verbose_messages(3)
```

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.

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.

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
230

```
disconnect_power_net_info(2)  
report_power_pin_info(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

<i>supply_net_name</i>	string
<i>list</i>	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  
232
```

```
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)
```

context_check

Enables or disables the Syntax Checker's context_check mode which checks commands for context errors.

SYNTAX

integer **context_check**
true | false

ARGUMENTS

true | false
Indicates whether to enable or disable the context_check mode. A value of **true** enables the mode and **false** disables the mode. Set the related command **syntax_check** to **false** before setting **context_check** to **true**.

DESCRIPTION

This command, with **syntax_check**, comprises a pair of commands that enable or disable the context_check or syntax_check modes of the dc_shell Syntax Checker. In the context_check mode, the command interpreter checks each command for the correct context. The design is read in to check the validity of design or library objects, missing files or attributes, file existence or permission, and incorrect object types. To find out whether either mode is currently enabled, retrieve the value of the corresponding status variable (**syntax_check_status** or **context_check_status**) by running either the **printvar context_check_status** or the **printvar syntax_check_status** command.

Identify and correct context errors in a script file that you want to include by running the file using the context_check mode. As context errors are encountered (for example, invalid design objects or object types, or nonexistent library objects), the normal error messages are issued but context checking continues to the end of the file.

Note that the context_check mode does not check objects that appear as a result of transformation (for example, using **compile** or **characterize**). You might receive a false error message if an object expected to be created by transformation cannot be found.

The **context_check** command performs syntax checking in addition to context checking, because syntax errors also cause context errors. For details about the features of the context_check and the syntax_check modes, see the *Design Compiler Command-Line Interface Guide*.

EXAMPLES

The following example enables the context_check mode, checks the file named myfile.tcl for context errors, and performs any file redirection specified within the script file.

```
prompt> context_check true
Context checker on.
prompt> source myfile.tcl
```

In the following example, you determine whether **syntax_check** is enabled before attempting to enable **context_check**. The status of the variable shows that **syntax_check** is currently enabled, probably by a previous operation. Disable **syntax_check** and then enable **context_check**.

```
prompt> printvar syntax_check_status
true
prompt> syntax_false
Syntax checker off.
prompt> context_check true
Context checker on.
```

SEE ALSO

```
syntax_check(2)
context_check_status(3)
syntax_check_status(3)
```

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_design

Copies a design to a new design, or copies a list of designs to a new file in dc_shell memory.

SYNTAX

```
string copy_design  
design_list  
target_name
```

Data Types

```
design_list      string  
target_name     string
```

ARGUMENTS

design_list
Specifies a lists of designs to copy. If a design is specified, the *target_name* must be a file name. All of the listed designs are copied to the the specified file with their base design names.

target_name
Specifies the name of the new design to create or the name of the file to which multiple designs are copied.

DESCRIPTION

The **copy_design** command copies a list of designs either to a target file or to a newly created design.

In the first form, the **copy_design** command copies the contents of *design_name* to *target_name*. The attributes and constraints of *design_name* are copied with the design. A design cannot be copied over another design that already exists in dc_shell.

In the second form, **copy_design** copies the designs listed in *design_name* to the file specified by *target_name*.

To write these files to disk, use the **write** command.

EXAMPLES

This example copies the design TEST to the design named BILL:

```
prompt> copy_design TEST BILL  
Information: Copying design /usr/example/TEST.DB:TEST to /usr/example/  
TEST.db:BILL.  
(UIMG-45)
```

```
prompt> list_designs  
BILL      TEST.db (*)
```

This example copies design TEST to the new file my_test.db:

```
prompt> copy_design TEST /usr/example/my_test.db:TEST  
Information: Copying design /usr/example/TEST.db:TEST to /usr/example/  
my_test.db:TEST.  
(UIMG-45)
```

```
prompt> list_designs -show_file  
/usr/example/TEST.db  
TEST (*)
```

```
/usr/example/my_test.db  
TEST
```

In this example, the **get_designs** command is used with **copy_design** to copy all of the designs currently read into dc_shell to a single file:

```
prompt> list_designs -show_file  
/usr/example/TEST.db  
TEST (*)
```

```
/usr/example/ADDER.db  
ADDER
```

```
prompt> copy_design [get_designs *] /usr/dc/project/all.db  
Information: Copying design /usr/example/TEST.db:TEST to /usr/dc/project/  
all.db:TEST.  
(UIMG-45)  
Information: Copying design ADDER.db:ADDER to /usr/dc/project/all.db:ADDER.  
(UIMG-45)
```

```
prompt> list_designs -show_file  
/usr/example/TEST.db  
TEST (*)
```

```
/usr/example/ADDER.db  
ADDER
```

```
/usr/dc/project/all.db  
ADDER      TEST
```

If you attempt to copy multiple designs without specifying a target file name as the second argument of the **copy_design** command, an error is generated, as shown below:

```
prompt> copy_design {A, B, C} all.db:TEST3  
Error: Cannot copy multiple designs to a single design  
(UIMG-4)
```

SEE ALSO

`current_design(2)`
`get_designs(2)`
`remove_design(2)`
`rename_design(2)`
`write(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)`

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.

```
cputime  
242
```

```
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)

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]  
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.
- 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 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)  
set_cell_location(2)
```

create_bsd_patterns

Generates a set of functional patterns for a boundary-scan design.

SYNTAX

```
status create_bsd_patterns  
-output test_program_name  
[-effort low | medium | high]  
[-type pattern_type]
```

Data Types

<i>test_program_name</i>	string
<i>pattern_type</i>	string

ARGUMENTS

-output *test_program_name*
Specifies the name of the output test program. On completion, **create_bsd_patterns** writes out the Synopsys pattern database (*test_program_name.vdb*) file that contains the generated patterns and the fault status information. By default, the *test_program_name* is *design_name* and the file is named *design_name.vdb*. This file is placed in the current working directory.

-effort low | medium | high
Controls the effort used to search for implemented instructions. The time taken increases exponentially for a sequential search on instruction registers (IRs) of length 8 or more. The **low** effort uses only heuristics. The **high** effort enforces sequential search. The default value is **medium**, which first uses heuristics and then random opcode generation for limited iterations.

-type *pattern_type*
Generates vectors to test the various aspects of the boundary scan of the design.
This option allows you to generate vectors to test bsr, tap, tdr, and reset mechanisms of the boundary scan of the design separately. This option also allows generation of leakage vectors and all functional vectors (bsr, tap, tdr, and reset vectors). This option also allows generation of vectors to test ac receivers and transmitters.
You must specify one of the following values: **all** - generates everything (the default).
functional - generates {tap_controller, reset, bsr, tdr, ac_input_pulse, ac_input_train, ac_output_pulse, ac_output_train}.
dc_parametric - generates vectors to {bsr, leakage}. Can also be used to measure voltage and current levels for I/O.
tap_controller - generates vectors to test the tap finite state machine function.
reset - generates vectors to test the asynchronous and synchronous reset mechanism.
tdr - generates vectors to test that each instruction implemented selects the

corresponding test data register.

- bsr** - generates vectors to test the function of the boundary scan register.
- leakage** - generates vectors to test I/O leakage.
- ac_input_pulse** - generates vectors for ac input tests with EXTEST_PULSE instruction. These vectors test the preset/transition/single ended/dc behaviors of ac receivers and the dc behavior of the ac instruction.
- ac_input_train** - generates vectors for ac input tests with EXTEST_TRAIN instruction. These vectors test the preset/transition/single ended/dc behaviors of ac receivers and the dc behavior of the ac instruction.
- ac_output_pulse** - generates vectors for ac output tests with EXTEST_PULSE instruction. These vectors test the transition behavior of ac drivers with the ac instruction. The vectors also test ac/dc selection cells.
- ac_output_train** - generates vectors for ac output tests with EXTEST_TRAIN instruction. These vectors test the transition behavior of ac drivers with the ac instruction. The vectors also test ac/dc selection cells.

The default behavior is to generate both functional and DC parametric vectors. This is the same as running the command with the **-type all** option.

DESCRIPTION

The **create_bsd_patterns** command generates functional test patterns for a boundary-scan design. The command invokes the Synopsys ANSI/IEEE 1149.1 compliance checker under the hood. If a boundary-scan design is non-compliant, the **create_bsd_patterns** command does not generate any functional vectors. If a boundary-scan design is compliant, **create_bsd_patterns** generates functional test patterns to test the synchronous and asynchronous TAP reset, the TAP controller finite state machine, the implemented boundary-scan instructions and associated test data registers, and the boundary-scan register.

If the design contains IEEE 1149.6 logic, the generated functional vectors also include tests for ac receivers and transmitters. Note that these vectors are not generated if compliance checker is run prior to pattern generation.

The test patterns are stored in the Core Test Language (CTL) model for the design, which can be converted to different formats using the **write_test** command.

EXAMPLES

The following example performs functional testbench generation on the current design and writes out the bscan.vdb Synopsys pattern database:

```
prompt> create_bsd_patterns -out bscan

Loading db file '/u/root_dir/libraries/my_class.db'
Loading design 'M1'
...Starting IEEE 1149.1 Compliance Checking.
...Inferring the boundary scan protocol for the design 'M1'.
...Generating the BSD test patterns for the design 'M1'.
.....Generating vectors to test the asynchronous test logic reset.
.....Generating vectors to test the synchronizing sequence of 5 1's on tms.
.....Generating vectors to test the TAP FSM.
.....Generating vectors to test boundary scan instructions.
.....Generating vectors to test the 'BYPASS' instruction.
```

```
.....Generating vectors to test the 'EXTEST' instruction.  
.....Generating vectors to test the 'SAMPLE' instruction.  
.....Generating vectors to test the 'CLAMP' instruction.  
.....Generating vectors to test the 'HIGHZ' instruction.  
.....Generating vectors to test the 'IDCODE' instruction.  
.....Generating vectors to test the boundary scan register.  
...Writing test program bscan to file /tmp/bsdvec/bscan.vdb.
```

BSD test patterns are generated successfully.

1

SEE ALSO

```
check_bsd(2)  
write(2)  
write_test(2)
```

create_bus

Creates a port bus or a net bus.

SYNTAX

```
status create_bus  
object_list  
bus_name  
[-type type_name]  
[-sort]  
[-no_sort]  
[-start start_bit]  
[-end end_bit]
```

Data Types

<i>object_list</i>	list
<i>bus_name</i>	string
<i>type_name</i>	string
<i>start_bit</i>	integer
<i>end_bit</i>	integer

ARGUMENTS

object_list
Specifies a list of ports or nets to be put into a bus. If both ports and nets have the same names, the ports are put into the bus.
This argument is required.

bus_name
Specifies the name of the bus. This name cannot be the same as any other bus or object of the same type. 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.

-type *type_name*
Assigns the specified *type_name* to the bused port. Valid types are port or net. This name appears in port declarations when a design with the bused ports is written to a file in the VHDL format. All buses that are assigned the same type name must be the same width (containing the same number of members).

-sort
Sorts the bits of the bus specified in *object_list* in alphanumeric order. By default, bits sort in reverse alphanumeric order.

-no_sort
Specifies that the order of the bits in the bus should be the same as specified on the command line. By default, bits sort in reverse alphanumeric order.

-start *start_bit*
Specifies the start bit for the bus.

`-end end_bit`
Specifies the end bit for the bus.

DESCRIPTION

The **create_bus** command creates a bus object of the type port or net. The number of objects in the list determines the bus width. Buses appear as multibit ports on a design or as multiwire nets in a design. This command groups any number of ports or any number of nets into a bus object. Unless the **-sort** or **-no_sort** option is selected, the specified objects are sorted by reverse alphanumeric order of names before the bus is created.

When using Design Vision to create a design schematic, bused nets are inferred from bused ports or pins in the design. Bused nets only appear as nets in a schematic if they are connected to a bused port.

If the start bit and end bit for the bus are specified with the **-start** and **-end** options, the bus is created with these start and end indices. If only the start bit is specified, an upward going bus starting at that start bit is built. If only the end bit is specified, an upward going bus ending at that end bit is built.

EXAMPLES

This example groups the A1, A2, and A3 ports into a bus named A. The ports must already exist. The order in which the ports appear in the bus is A3, A2, A1.

```
prompt> create_bus {A1 A2 A3} A
```

This example groups the B1, B2, and B3 nets into a bus named B. The nets must already exist and there must not be any ports with the same names. The order in which the nets appear in the bus is B3, B2, B1.

```
prompt> create_bus {B1 B2 B3} B
```

This example groups the C1, C2, and C3 nets into a bus named C. The nets must already exist and because the object *net* is defined in the **get_nets** command, it does not matter if there are ports with the same name. The order in which the nets appear in the bus is C3, C2, C1.

```
prompt> create_bus [get_nets C1 C2 C3] C
```

This example groups the existing D1, D2, and D3 ports into a bus named D and assigns the 3-bit type to the bus:

```
prompt> create_bus {D1 D2 D3} D -type "3-bit"
```

This example groups the existing D1, D2, and D3 ports into a bus named D and assigns it the index range 6-to-4. Port D1 is assigned index 6, port D2 to index 5, and port D3 to index 4.

```
prompt> create_bus {D1 D2 D3} D -start 6 -end 4
```

This example groups the E1, E2, and GH ports into a bus named E. The ports must already exist. Because the **-sort** option is used, the first bit is port E1, the second bit is port E2, and the third bit is port GH.

```
prompt> create_bus {E2 GH E1} E -sort
```

This example groups the R1, R2, and GH ports into a bus named R. The ports must already exist. Because the **-no_sort** option is used, the first bit is port R1, the second bit is port R2, and the third bit is port GH.

```
prompt> create_bus {R1 R2 GH} R -no_sort
```

This example groups the ports in the MID subdesign into a bus named NEW. U1 is a unique instantiation of MID.

```
prompt> create_bus {U1/R1 U1/R2 U1/GH} NEW -no_sort
```

This example groups the nets in the MID subdesign into a bus named NET_BUS. U1 is a unique instantiation of MID.

```
prompt> create_bus [get_nets U1/R1 U1/R2 U1/GH] NET_BUS -no_sort
```

SEE ALSO

```
get_nets(2)  
remove_bus(2)  
report_bus(2)
```


create_cache

Populates the cache directories with instances of the requested synthetic modules.

SYNTAX

```
int create_cache  
-module module_list  
[-implementation implementation_list]  
[-parameters parameter_list]  
[-operating_condition operating_condition]  
[-wire_load list] [-report]
```

Data Types

<i>module_list</i>	list
<i>implementation_list</i>	list
<i>parameter_list</i>	list
<i>operating_condition</i>	string
<i>list</i>	list

ARGUMENTS

-module *module_list*
Lists the synthetic module names used to create and place in the cache. Strings in the list can contain the "*" wildcard character (for example: { DW_01 DW_* *_adders }).

-implementation *implementation_list*
Restricts the part creation to the implementations in the list. Strings in the list can contain the "*" wildcard character.

-parameters *parameter_list*
Restricts the part creation to instances of the parts that have the parameters in the list. **create_cache** displays warning or error messages when an incorrect number of parameters is specified; a list of the required parameters is also displayed. This argument is required unless the synthetic module is not parameterizable. For more information on *parameter_list*, see the DESCRIPTION section.

-operating_condition *operating_condition*
Optimizes and models the created parts using the specified *operating_condition*. If not specified, the default operating condition in the target library is used.

-wire_load *list*
Lists the wire load models to use to optimize and model the created parts. If no wire load is provided, the default wire load model in the target library is used.

-report
Generates a brief timing report that displays the longest path in the created part.

DESCRIPTION

Creates instances of the specified synthetic modules and places them in the cache. If no implementation name is given, an instance of all valid implementations of the synthetic module is created.

EXAMPLES

The following example shows the use of the **create_cache** command:

```
prompt>create_cache -module {DW01_add} \  
        -implementation {cla_rpl}  
        -parameters {"width = 8"}  
        -oper "Best_Case_Commercial"
```

The **-parameters** command argument has some additional features. The *parameter_list* is a space-separated list of quoted parameter specifications (for example: {"N=8,M=6" "N=3"}). Each quoted parameter specification is a comma-separated list of parameter settings. Parameter settings are a parameter name, followed by an equal sign (=), followed by a parameter value. Parameter settings can also support ranges for the parameter value: "width= [8;17)", where a square bracket ("[") means the end point is inclusive, and a parenthesis (")") means the end point is exclusive. Thus, '-parameters {"width = [8;17)"}' means create instances of the synthetic modules with parameter "width" set to 8 to 16.

The following example creates DW02 multipliers for all possible combinations of the A_width and B_width parameters in the given range - 12 to 16 X 8 to 16. The second set of parameters requests the creation of a 32x32 bit multiplier. Since no implementation name is provided, an instance of all valid implementations of the DW02_mult module is created.

```
prompt> create_cache -mod DW02_mult \  
        -wire_load {"70x70" "80x80"} \  
        -p {"A_width=(11;16], B_width=(7;16]", \  
        "A_width=32, B_width=32"}
```

SEE ALSO

```
remove_cache(2)  
report_cache(2)  
cache_read(3)  
cache_write(3)
```

create_cell

Creates cells in the current design or its subdesigns.

SYNTAX

```
int create_cell  
cell_list  
reference_name  
[-logic 0 | 1]  
[-only_physical]
```

Data Types

<i>cell_list</i>	list
<i>reference_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. You must specify the *reference_name* unless you use the **-logic** option. Ports on the reference determine the name, number, and direction of pins on the new cell.

-logic 0 | 1
Specifies that the new cell generates a logic 0 or logic 1 value. The logic value must be either **0** or **1**. The cell created when specifying **-logic** has a single output pin. By default, this option is off.

-only_physical
Creates the new physical cell using the reference from the physical library. By default, this option is off.

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, a library cell, a logic 0 generator, or a logic 1 generator.

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 a cell named a in the design corresponding to the m1/U1 and m2/U1 instances. The design corresponding to m1/U1 and m2/U1 must be unique.

```
prompt> create_cell {m1/U1/a m2/U1/a} -logic 0
```

The following example creates cells using existing designs as references.

```
prompt> create_design ALPHA  
prompt> current_design ALPHA  
prompt> create_cell U1 ADDER  
prompt> create_cell U2 SUBTRACTOR
```

The following example creates logic 1 and logic 0 cells.

```
prompt> create_cell LOGIC_ONE -logic 1  
prompt> create_cell LOGIC_ZERO -logic 0
```

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
```

The following example creates physical cells using physical library cells as references.

```
prompt> create_cell -only_physical {U3 U4} lib.pdb:my_lib/pad0  
prompt> create_cell -only_physical phys_inst1 test_lib.pdb:mylib/PAD0
```

SEE ALSO

```
current_design(2)  
remove_cell(2)  
report_cell(2)  
set_cell_location(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)
- compile(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_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_design

Creates a design in dc_shell memory.

SYNTAX

```
status create_design  
design_name  
[file_name]
```

Data Types

```
design_name      string  
file_name       string
```

ARGUMENTS

design_name
Specifies the name of the design created. A *design_name* must be unique within the default file unless *file_name* is specified.

file_name
Specifies the name of the file in which the new design is placed. The *file_name* argument is optional. By default, the design is placed in *design_name.db* in the current directory. The file is not written by **create_design**. The design is placed in a default file in dc_shell.

DESCRIPTION

This command creates a new design in dc_shell. The new design created is empty. A new design contains no subobjects. Subobjects can be created with **create_bus**, **create_cell**, **create_net**, and **create_port**. To make the new design the working design in dc_shell, use **current_design**. To remove designs from dc_shell, use **remove_design**.

EXAMPLES

The following example uses **create_design** to create a design:

```
prompt> create_design ADDER  
  
prompt> list_designs  
  
prompt> current_design ADDER
```

The following example specifies the default file for the new design:

```
prompt> create_design SUBTRACTOR "/tmp/blat.db"  
  
prompt> list_designs -show_file
```

In the following example, **create_design** specifies a nonunique design name:

```
prompt> create_design example
```

```
prompt> create_design example
```

SEE ALSO

```
create_bus(2)  
create_cell(2)  
create_net(2)  
create_port(2)  
current_design(2)  
list_designs(2)  
remove_bus(2)  
remove_cell(2)  
remove_design(2)  
remove_net(2)  
remove_port(2)
```

create_die_area

Specifies the die area as rectangular or polygonal. This command is supported only in Design Compiler topographical mode.

SYNTAX

```
status create_die_area  
-coordinate {llx lly urx ury}  
-polygon {{x_0 y_0} {x_1 y_1} ... {x_n y_n}}
```

ARGUMENTS

```
-coordinate {llx lly urx ury}  
    Specifies a rectangular die area, with the lower-left and upper-right  
    coordinates.  
  
-polygon {{x_0 y_0} {x_1 y_1} ... {x_n y_n}}  
    Specifies a die area as rectangular or polygonal, with rectilinear  
    coordinates. The coordinates may be ordered clockwise or counter-clockwise.
```

DESCRIPTION

The **create_die_area** command defines the die area for the current design. The command overwrites the previous die area.

The core area is inferred from the die area and site rows:

- If both the die area and site rows are specified, the core area is computed by shrinking the die area until the site rows are reached.
- If only the die area is specified, the core area will have the same shape and size.
- If only site rows are specified, then the die area and core area are computed as the bounding box of the site rows.
- If there is no die area and site rows, the core area is determined by the **set_placement_area** command.

EXAMPLES

The following example shows how to set a die area of polygon L shape:

```
prompt> create_die_area -polygon {{0 0} {0 400} {200 400} {200 200} \  
      {400 200} {400 0} {0 0}}
```

SEE ALSO

`extract_physical_constraints(2)`

```
report_physical_constraints(2)
write_physical_constraints(2)
```

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_ilm

Creates an interface logic model (ILM) for the current design. After command execution, the design in memory is the interface logic model.

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]
[-include_all_logic]
[-verbose]
```

Data Types

<i>port_list</i>	collection
<i>levels</i>	int

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. After this operation is completed, the design in memory is still the original, complete design. 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. After this operation is completed, the design in memory is the original complete design. 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.

-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.

-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 design in memory is the ILM. You can save the ILM design in memory to the ILM view by using the **write_milkyway** command.

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 **physopt** command 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 save an ILM to a file in .ddc format. The example creates an ILM for the current design with the specification 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}]
prompt> write -format ddc -hier -output foo_ilm.ddc
```

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}]
prompt> write_milkyway -output my_ilm
```

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  
prompt> write_milkyway -output my_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  
prompt> write_milkyway -output my_ilm
```

SEE ALSO

```
all_scenarios(2)  
create_clock(2)  
create_generated_clock(2)  
extract_rc(2)  
propagate_constraints(2)  
read_sdc(2)  
set_case_analysis(2)  
write(2)  
write_milkyway(2)  
ilm_ignore_percentage(3)
```

create_multibit

Creates a multibit component for the specified list of cells or cell instances in the current design.

SYNTAX

```
status create_multibit  
object_list  
[-name multibit_name]  
[-sort]  
[-no_sort]
```

Data Types

<i>object_list</i>	list
<i>multibit_name</i>	string

ARGUMENTS

object_list
Specifies a list of cells in the current design for which a multibit component has to be created.

-name *multibit_name*
Specifies the name to be given to the new multibit component. If this argument is omitted, then the name of the first cell in the component is used to generate a name for the multibit component. If a multibit component by the name *multibit_name* already exists in the current design, then an error is issued and the command is aborted.

-sort
Sorts the cells specified in *object_list* in alphanumeric order. By default, cells are sorted in reverse alphanumeric order of their names.

-no_sort
Specifies that the order of the cells in the multibit component should be the same as specified on the command line. By default, cells are sorted in reverse alphanumeric order of their names.

DESCRIPTION

This command creates a new multibit component and each cell or instance in *object_list* is inserted into that multibit component. If a name is provided, the new multibit component gets that name, otherwise a name is generated internally. If a cell instance is given, then the parent of the cell instances should be same and the parent has to be unique.

If the cells in the list are MUX_OP cells, then a separate multibit component is created for each cell.

The list of cells is sorted in a reverse alphanumeric order by default. Use the -

sort option to sort in alphanumeric order, or the **-no_sort** order to order cells in the same way as specified on the command line.

This command requires the design to be linked. If an attempt to link the design fails, the command is aborted. If any of the cells in *object_list* already belong to a multibit component, then an error is issued and the command is aborted.

EXAMPLES

In the following example, a multibit component is created for four registers:

```
prompt> report_multibit
```

```
*****
Report : multibit
Design : subtest
Version: 1998.02
Date   : Fri Aug 29 10:01:19 1997
*****
```

Attributes:

- b - black box (unknown)
- h - hierarchical
- n - noncombinational
- r - removable
- u - contains unmapped logic

Total 0 Multibit Components

```
prompt> create_multibit -name y_reg {y_reg*}
```

1

```
prompt> report_multibit
```

```
*****
Report : multibit
Design : subtest
Version: 1998.02
Date   : Fri Aug 29 10:01:19 1997
*****
```

Attributes:

- b - black box (unknown)
- h - hierarchical
- n - noncombinational
- r - removable
- u - contains unmapped logic

Multibit Component : y_reg

Cell	Reference	Library	Area	Width	Attributes
y_reg[3]	**SEQGEN**		0.00	1	n, u

y_reg[2]	**SEQGEN**	0.00	1	n, u
y_reg[1]	**SEQGEN**	0.00	1	n, u
y_reg[0]	**SEQGEN**	0.00	1	n, u

Total 4 cells		0.00	4	

Total 1 Multibit Components

In the next example, the command is executed on a list of registers and uses the **-no_sort** option:

```
prompt> create_multibit -name y_reg -
no_sort {y_reg[0] y_reg[2] y_reg[1] y_reg[3]}
```

1

```
prompt> report_multibit
```

```
*****
Report : multibit
Design : subtest
Version: 1998.02
Date   : Fri Aug 29 10:01:19 1997
*****
```

Attributes:

- b - black box (unknown)
- h - hierarchical
- n - noncombinational
- r - removable
- s - synthetic operator
- u - contains unmapped logic

Multibit Component : y_reg

Cell	Reference	Library	Area	Width	Attributes

y_reg[0]	**SEQGEN**		0.00	1	n, u
y_reg[2]	**SEQGEN**		0.00	1	n, u
y_reg[1]	**SEQGEN**		0.00	1	n, u
y_reg[3]	**SEQGEN**		0.00	1	n, u

Total 4 cells			0.00	4	

Total 1 Multibit Components

In the next example, the command is executed on a list of MUX_OP cells:

```
prompt> create_multibit {U4, U7}
```

1

```
prompt> report_multibit
```

```
*****
Report : multibit
Design : subtest
Version: 1998.02
Date   : Fri Aug 29 10:01:19 1997
*****
```

Attributes:

- b - black box (unknown)
- h - hierarchical
- n - noncombinational
- r - removable
- s - synthetic operator
- u - contains unmapped logic

Multibit Component : y_reg

Cell	Reference	Library	Area	Width	Attributes
U4	*MUX_OP_4_2.2		0.00	2	s, u
Total 1 cells			0.00	4	

Multibit Component : U7_multibit

Cell	Reference	Library	Area	Width	Attributes
U7	*MUX_OP_4_2.2		0.00	2	s, u
Total 1 cells			0.00	2	

Total 2 Multibit Components

In the next example, the command is executed using cell instances U1/foo1 and U1/foo2. In this case the instance U1 corresponding to design MID is unique.

```
prompt> create_multibit {U1/foo1 U1/foo2} -name mult
```

1

```
prompt> report_multibit
```

```
*****
Report : multibit
Design : subtest
Version: 1998.02
Date   : Fri Aug 29 10:01:19 1997
*****
```

Attributes:

- b - black box (unknown)
- h - hierarchical
- n - noncombinational
- r - removable
- s - synthetic operator
- u - contains unmapped logic

Multibit Component : mult

Cell	Reference	Library	Area	Width	Attributes
U1/foo1	**SEQGEN**		0.00	1	n, u
U1/foo2	**SEQGEN**		0.00	1	n, u
Total 2 cells			0.00	2	

Total 1 Multibit Components

SEE ALSO

current_design(2)
remove_multibit(2)
report_cell(2)
report_compile_options(2)
report_multibit(2)
report_reference(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_net

Creates nets in the current design or its subdesign.

SYNTAX

```
status create_net  
    | | | net_list
```

Data Types

net_list list

ARGUMENTS

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.

To bundle scalar nets into buses, use the **create_bus** command.

Nets connect pins and ports in a design. When you create nets with **create_net**, they are not connected. To establish this connection, use the **connect_net** 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> 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> current_design
prompt> create_net {mid1/N1 mid1/N2 mid1/N3 mid1/N4}
prompt> get_nets mid1*
```

SEE ALSO

```
all_connected(2)
connect_net(2)
current_design(2)
disconnect_net(2)
remove_net(2)
remove_bus(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
    | [-length real]
[-width real]
[-path_type square | round | extend_half_width | octagon]
-layer layer
-net net_name
[-vertical]
[-route_type route_type]
[-datatype int]
[-quiet]
[-net_type string]
```

Data Types

<i>point</i>	point
<i>list_of_points</i>	list of points
<i>rect</i>	rectangle
<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.

-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:
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 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

<i>user_enter</i>	User entered
<i>signal_route, signal_route_detail</i>	Detail routing
<i>signal_route_global</i>	Global routing
<i>pg_ring</i>	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.

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_placement_blockage(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_pass_directories

Creates the directory structure required for storing Automated Chip Synthesis data.

SYNTAX

```
int create_pass_directories  
pass_list
```

Data Types

pass_list list

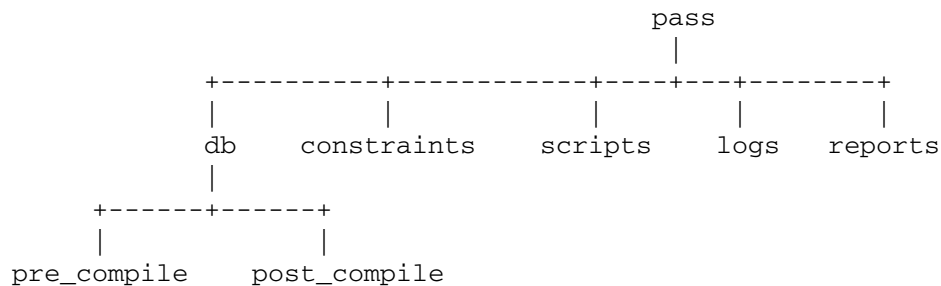
ARGUMENTS

pass_list
Specifies a list of directory names that specify the passes that require data directories.

DESCRIPTION

The **create_pass_directories** command creates a directory structure to store pass data for each specified pass. The pass data directories are subdirectories to the Automated Chip Synthesis working directory, specified by the **acs_work_dir** variable.

The directory structure for each pass data directory is:



If any part of the data directory structure for a specified pass does not exist, Automated Chip Synthesis creates it.

The @ character is reserved to specify pass-dependent directory names. It cannot be used as a literal character in Automated Chip Synthesis directory paths.

SEE ALSO

```
remove_pass_directories(2)  
acs_work_dir(3)
```


create_placement_blockage

Creates a new placement blockage.

SYNTAX

```
status create_placement_blockage
-bbox rectangle
[-type hard | soft | partial]
[-blocked_percentage percentage]
[-name blockage_name]
```

Data Types

<i>rectangle</i>	x1 y1 x2 y2
<i>percentage</i>	integer
<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 as follows:

- **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 placer tries not to place standard cells or hard macros in the specified region but will do so if the congestion is too high.

- **partial** for partial blockage

For a partial blockage the amount of area used to place cell will be limited to the specified percent of the blockage area.

The default value is **hard**.

-blocked_percentage *percentage*

Specifies the percentage blockage for a partial blockage.

This option can only be used with the **partial** blockage type.

-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_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.

To bundle scalar ports into buses, use **create_bus**.

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_bus(2)
remove_bus(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.

-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_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

report_power_switch(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_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]
```

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.

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.

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)
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.

Creation and usage of scenarios is available with Design Compiler Graphical.

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_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

extract_physical_constraints(2)
report_physical_constraints(2)
write_physical_constraints(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

`report_supply_port(2)`

create_test_protocol

Creates a test protocol based on user specification.

SYNTAX

```
int create_test_protocol  
[-infer_asynch]  
[-infer_clock]  
[-capture_procedure single_clock | multi_clock]
```

ARGUMENTS

-infer_asynch
If this option is specified, infers asynchronous set and reset signals in the design.

-infer_clock
If this option is specified, infers test clocks in the design.

-capture_procedure single_clock | multi_clock
Specifies the capture procedure type. The default value is **multi_clock**. **multi_clock** creates a protocol file which uses generic capture procedures for all capture clocks. **single_clock** creates a protocol file which uses the legacy 3-vector capture procedures for all capture clocks.

DESCRIPTION

This command creates a test protocol for the current design based on user specifications issued prior to running this command. Such specifications are made using commands such as **set_dft_signal**.

This command removes any protocol that is present in memory due to a previous execution of **create_test_protocol**. However, if the protocol is present in memory due to a previous execution of **read_test_protocol**, it issues a warning and does not create a new test protocol.

This command checks whether the user-specified values are consistent with each other. If they are not, it issues an error and does not generate a protocol.

If **-infer_asynch** is specified, **create_test_protocol** infers asynchronous set and reset signals in the design, and places them at off state during scan shifting.

If **-infer_clock** is specified, **create_test_protocol** infers test clock pins from the design, and pulses them during scan shifting. The timing of the test clock is based on the **test_default_period**, **test_default_delay**, **test_default_strobe**, and **test_default_strobe_width** variables.

Both **-infer_asynch** and **-infer_clock** take previous user specifications into account.

When **-capture_procedure multi_clock** is specified, which is also default, the protocol file contains four capture procedures: **multiclock_capture**,

allclock_capture, allclock_launch and allclock_launch_capture procedure. This single protocol file can be used for ATPG stuck-at, transition delay and path delay testing.

The **create_test_protocol** command automatically generates a master_observe procedure for LSSD designs in STIL format. However, to use this feature, you must set the scan style to LSSD design by using the **set_scan_configuration -style** command.

The **create_test_protocol** command should be executed before running the dft_drc command because design rule checking requires a test protocol.

If the interface of a design changes, for example, a port is created or removed, after a test protocol is created, you don't need to rerun **create_test_protocol** because the interface change is automatically taken into account in the protocol.

However, if you change any test specification of the design, for example, you use **set_dft_signal** to specify a test clock, the protocol present in the memory is deleted. Therefore, you need to rerun **create_test_protocol** to create a test protocol. The commands that change test specification of the existing design include set_dft_signal and set_scan_path.

The **insert_dft** command automatically updates the protocol after inserting scan circuitry into the design, and dft_drc can be executed afterward without rerunning **create_test_protocol**.

EXAMPLES

The following example creates a test protocol in memory for the current design:

```
prompt> create_test_protocol
```

The following example creates a test protocol in memory for the current design, inferring both asynchronous set and reset signals as well as test clocks:

```
prompt> create_test_protocol -infer_clock -infer_asynch
```

SEE ALSO

```
current_design(2)
set_scan_configuration(2)
set_scan_path(2)
set_dft_signal(2)
read_test_protocol(2)
write_test_protocol(2)
remove_test_protocol(2)
```

create_voltage_area

Creates a voltage area at the specified region for providing placement constraints of cells associated with the region. This command is supported only in topographical mode.

SYNTAX

```
int create_voltage_area  
modules -name voltage_area_name  
    | -power_domain power_domain_name  
-coordinate llx1_lly1_urx1_ury1_...  
[-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 area 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

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.

-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
```

SEE ALSO

```
remove_voltage_area(2)  
report_voltage_area(2)  
create_bounds(2)
```

create_wiring_keepouts

Creates a wiring keepout.

SYNTAX

```
status create_wiring_keepouts  
-name name  
-layer layer  
-coordinate list_of_float_values
```

Data Types

<i>name</i>	string
<i>layer</i>	string
<i>list_of_float_values</i>	list

ARGUMENTS

-name *name*
Specifies the name of the wiring keepout to be created.

-layer *layer*
Specifies the name of the layer on which the routing obstruction is to lie.

-coordinate *list_of_float_values*
Specifies a list of float values that serve as the coordinates of the wiring keepout. These coordinates are relative to the design origin at (0 0) and are in units of micron.

DESCRIPTION

The **create_wiring_keepouts** command creates a wiring keepout.

A wiring keepout is not created if the given keepout name matches with any other existing wiring keepout name. However, a wiring keepout is created if its name matches the name of an existing placement keepout.

EXAMPLES

The following is an example of the **create_wiring_keepouts** command:

```
prompt> create_wiring_keepouts -name "my_keep1" -layer "METAL1" \  
-coord {12 12 100 100}
```

SEE ALSO

```
extract_physical_constraints(2)  
report_physical_constraints(2)  
reset_physical_constraints(2)
```

create_wiring_keepouts

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_dft_partition

Sets or gets the design partition for the current specification.

SYNTAX

```
status current_dft_partition  
[design_partition_label]
```

Data Types

design_partition_label string

ARGUMENTS

design_partition_label
Specifies the working design partition for many commands. If the *design_partition_label* is not specified, the tool uses the last design partition specified. If the *design_partition_label* specifies a partition name that is not defined in the CTL model for the current design and that has not been specified previously, using the **define_dft_partition** command, an error is issued and the working test mode remains unchanged.

DESCRIPTION

The **current_dft_partition** command sets the design partition for many specification commands, such as **set_dft_signal**, **set_scan_configuration**, **set_scan_compression_configuration**, and **set_scan_path**. When no arguments are specified, **current_dft_partition** returns the name of the current DFT partition.

To display the names of all of the design partitions in the test model for the current design, use the **report_dft_partition** command.

EXAMPLES

The following example sets the required number of scan chains to 3 and 2 to the partitions part1 and part2, respectively:

```
prompt> define_dft_partition part1 -include {U1 U2}  
prompt> define_dft_partition part2 -include {U3 U4}  
prompt> current_dft_partition part1  
prompt> set_scan_configuration -chain_count 3  
prompt> current_dft_partition part2  
prompt> set_scan_configuration -chain_count 2  
prompt> preview_dft  
1
```

SEE ALSO

`define_dft_partition(2)`

`current_dft_partition`
326

```
insert_dft(2)
preview_dft(2)
remove_dft_partition(2)
report_dft_partition(2)
set_dft_signal(2)
set_scan_compression_configuration(2)
set_scan_configuration(2)
set_scan_path(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)  
find(2)  
list_designs(2)  
list_instances(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.

Creation and usage of scenarios is available with Design Compiler Graphical.

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)
```

current_test_mode

Sets or gets the working test mode for the current design.

SYNTAX

```
int current_test_mode  
[test_mode_label]
```

ARGUMENTS

test_mode_label
Specifies the working test mode for many commands. If the *test_mode_label* is not specified, `dc_shell` uses the last test mode specified. If *test_mode_label* specifies a mode name that is not defined in the CTL model for the current design and that has not been specified previously using the **define_test_mode** command, an error is issued and the working test mode remains unchanged.

DESCRIPTION

The **current_test_mode** command sets the working mode for many commands like **dft_drc** and **report_test_mode**. When no arguments are specified, **current_test_mode** returns the name of the current working test mode.

To display the names of all the modes in the test model for the current design, use the **report_test_model** command.

To display the names of all the modes of a design defined with the **define_test_mode** command, use the **list_test_modes** command.

EXAMPLES

The following example sets the test mode to "Internal_scan" for the current design.

```
prompt> current_test_mode Internal_scan  
1
```

The following example gets the test mode for the current design.

```
prompt> current_test_mode  
Current test mode is 'Internal_scan'.  
1
```

SEE ALSO

```
dft_drc(2)  
current_design(2)  
list_test_modes(2)  
report_test_model(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

dc_allocate_budgets

Allocates budgets to specified cells.

SYNTAX

```
string dc_allocate_budgets
[cell_list]
[-write_script]
[-file_format_spec format_spec]
[-format dctcl | dcsh]
[-separator hierarchy_separator]
[-mode rtl | gate | mixed]
[-budget_design_ware]
[-levels budget_levels]
[-no_interblock_logic]
[-min_register_to_output minimum_budget]
[-min_input_to_output minimum_budget]
[-min_input_to_register minimum_budget]
```

Data Types

<i>cell_list</i>	list
<i>format_spec</i>	string
<i>hierarchy_separator</i>	string
<i>budget_levels</i>	string
<i>minimum_budget</i>	float

SYNTAX

```
string dc_allocate_budgets
[cell_list]
[-write_script]
[-file_format_spec format_spec]
[-format dctcl | dcsh]
[-separator hierarchy_separator]
[-mode rtl | gate | mixed]
[-budget_design_ware]
[-levels budget_levels]
[-no_interblock_logic]
```

ARGUMENTS

cell_list

Specifies the list of cells to budget. The list of cells is required when you do not use the **-levels** option.

-write_script

Specifies that the context of each budgeted cell or design is to be written into a constraint file.

`-file_format_spec format_spec`
 Specifies the format for the names of constraint files generated for each budgeted cell, if the **-write_script** option is used. The default is `/%C.dctcl`. For details, see the `DESCRIPTION` section.

`-format dctcl | dcsh`
 Specifies the output format for script files. Allowed values are **dctcl** (the default) or **dcsh**.

`-separator hierarchy_separator`
 Specifies a single character to use as a separator in deriving the names of constraint files generated for each cell. All characters are allowed except `/` (slash), `%` (percent), and `e\` (double backslash). The default is the `@` (at sign).

`-mode rtl | gate | mixed`
 Specifies whether the design to be budgeted is an RTL or a gate-level design, or a design that is mixed with RTL and gates. The default is **gate**. For details on the **-mode** option, see the *Budgeting for Synthesis User Guide*.

`-budget_design_ware`
 Specifies that DesignWare components are to be budgeted. By default, Design Budgeter considers these components fixed.

`-levels budget_levels`
 Specifies the number of levels of hierarchy for which to allocate budgets starting from the specified cells. Allowed values are `all`, `1`, `2`, `3...`, and `n`. The default is **all**, which allocates budgets down all levels of hierarchy.

`-no_interblock_logic`
 Specifies that interblock logic is not to be budgeted. By default, the design budgeter considers interblock logic not fixed.

`-min_register_to_output minimum_budget`
 For XG and DB (Tcl) modes only: a positive floating point number that specifies the minimum budget for any path segment from an internal register to an output pin of a budgeted cell.

`-min_input_to_output minimum_budget`
 For XG and DB (Tcl) modes only: a positive floating point number that specifies the minimum budget for any path segment from an input pin to an output pin of a budgeted cell.

`-min_input_to_register minimum_budget`
 For XG and DB (Tcl) modes only: a positive floating point number that specifies the minimum budget for any path segment from an input pin to an internal register of a budgeted cell.
 The **-min_register_to_output**, **-min_input_to_output**, and **-min_input_to_register** options allow you to avoid overconstraining noncompressible path segments, such as segments to registered output pins or segments from registered input pins. Use these options carefully to avoid creating mutually inconsistent constraints. If necessary, the budgeter assigns path delays less than these minimum budget values to prevent negative slack in the budget.

DESCRIPTION

The **dc_allocate_budgets** command distributes the timing constraints among the specified cells. The command allows you to specify cells by either non-hierarchical budgets allocation or the hierarchical budgets allocation.

The non-hierarchical budgets allocation is the default method. Only cells specified with *cell_list* are budgeted. Designs are not budgeted.

The hierarchical budgets allocation is enabled by the **-levels** option. When activated, the tool allocates budgets hierarchically. If you specify *cell_list*, budgets are hierarchically allocated starting from these cells. If you do not specify *cell_list*, budgets are allocated starting from cells in the top level of the design. If a design has several instances, a constraint file can be written out for each instance for cell-based file format spec (%C). For design-based file format spec (%D), the last instance of the design is used to generate the constraint file.

The **-file_format_spec** option specifies the format for the names of constraint files generated for each budgeted cell. The format contains the directory in which files are to be written, and the prefix, the suffix, and the extension to use. The format specification must contain only one conversion specification: either %D, which indicates a design name, or %C, which indicates a cell name.

For the %D conversion specification, the constraint file name is obtained by replacing the conversion specification by either the name of the cell design if the design is referenced once (uniquified design), or by the concatenation of the design name and the full cell name.

For the %C conversion specification, the constraint file name is obtained by replacing the conversion specification by the full cell name.

The default value of **-file_format_spec** depends on the **-format** option.

The default value is `./%C.dctcl` if the **-format** option is not specified, `./%C.dctcl` if the **-format** option value is **dctcl**, or `./%C.dcsh` if the **-format** option is **dcsh**.

By default, budget allocation is performed automatically. You can optionally specify user budgets or budget ratios using the **set_user_budget** command.

EXAMPLES

The following command allocates budgets to cells *I1* and *I2*, and writes a constraint file in *dcsh* format to files named *I1.dcsh* and *I2.dcsh*. The report shows the results.

```
prompt> dc_allocate_budgets -write_script
        -file_format_spec %C.dcsh -format dcsh {I1 I2}
```

```
prompt> report_path_budget
```

```
    Loading design 'budtest1'
```

```
*****
```

```
Report : budget
```

```

        -path full
        -delay max
        -max_paths 1
Design : budtest1
Version: 2000.11-PROD
Date   : Mon Oct 16 11:12:16 2000
*****

```

Operating Conditions:
Wire Load Model Mode: top

Startpoint: I1/reg3 (rising edge-triggered flip-flop clocked by clk)
Endpoint: I3/reg1 (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
clock network delay (ideal)	0.00	0.00
I1/reg3/CP (FD1)	0.00	0.00 r
I1/reg3/Q (FD1)	0.85 *	0.85 f
I1/out2 (desA)	0.30	1.15 f
I2/in2 (desB)	0.00	1.15 f
I2/out2 (desB)	1.75	2.90 f
I3/in2 (desC)	0.00	2.90 f
I3/reg1/D (FD1)	0.30	3.20 f
data arrival time		3.20
clock clk (rise edge)	4.00	4.00
clock network delay (ideal)	0.00	4.00
I3/reg1/CP (FD1)	0.00	4.00 r
library setup time	-0.80	3.20
data required time		3.20

data required time		3.20
data arrival time		-3.20

slack (VIOLATED: increase significant digits)		0.00

1

SEE ALSO

```

report_path_budget(2)
set_user_budget(2)

```

define_design_lib

Maps a design library to a UNIX directory.

SYNTAX

```
int define_design_lib  
  library_name  
  -path directory
```

Data Types

<i>library_name</i>	string
<i>directory</i>	string

ARGUMENTS

library_name
Specifies the library to be mapped.

-path *directory*
Specifies the directory to which the library is mapped.

DESCRIPTION

The **define_design_lib** command maps a design library to a UNIX directory. The directory is used to store intermediate representations of designs.

When **define_design_lib** is used with the **design_library_file** command, the file is automatically reread when it is changed. As a result, if the file contains a library definition that was manually defined with **define_design_lib**, the manual definition is overridden.

EXAMPLES

The following example uses **define_design_lib** to map the *MY_LIB* library to the *~library* directory:

```
prompt> define_design_lib MY_LIB -path ~/library
```

SEE ALSO

```
analyze(2)  
elaborate(2)  
get_design_lib_path(2)  
report_design_lib(2)  
write_design_lib_paths(2)
```


define_dft_design

Characterizes a design as a DFT design to be used for DFT insertion.

SYNTAX

```
status define_dft_design  
-design_name design_name  
[-type design_type]  
[-interface access_list]  
[-test_model model_path_name]  
[-params param_list]
```

Data Types

<i>design_name</i>	string
<i>design_type</i>	string
<i>access_list</i>	list
<i>model_path_name</i>	string
<i>param_list</i>	list

ARGUMENTS

-design_name *design_name*
Identifies the design to be instantiated during DFT insertion.
This argument is required.

-type *design_type*
Specifies the type of the DFT design to which the design corresponds. Valid design types are:

DECODE	CONTROL_FORCE	Z_CONTROL_FORCE	OBSERV_DGEN
SHIFT_REG	BC1	BC2	BC3
BC4	BC5	BC7	BC8
BC9	BC10	WC_D1	WC_D1_S
WC_S1	WC_S1_S	INSTREG	TAP
TAP_UC	LBIST_CONTROLLER	LBIST_CODEC	LBIST_XCONTROLLER
LBIST_XCODEC	CLK_MUX	CLK_CHAIN	MBIST_CONTROLLER
MBIST_WRAPPER		PAD	AC1
AC2	AC7	AC_SELU	AC_SELX

There is no default value for this option. When the option is not specified, the tool creates a new DFT design type from the specified design name and the specified interface access list.

-interface *access_list*
Specifies the list of signal mapping triplets relating a signal type to a pin of the design specified with **-design_name** *design_name*. The valid format of a signal mapping triplet is as follows:
signal_type_name *pin_name* *pin_polarity*
The *signal_type_name* specifies the signal type name of the latest of DesignWare version of the DFT design pin name. The *pin_name* specifies the pin name of the design specified with **-design_name** *design_name*. The *pin_polarity* specifies the polarity of the specified design pin with respect to the *signal_type_name*.

Valid values for *pin_polarity* are **h** for normal polarity and **l** for negative polarity.

There is no default value for this option. When the option is not specified, it is assumed that the design does not have interface access pins for use with DFT insertion.

-test_model *model_path_name*

Identifies the path name of the test model for the specified DFT design. There is no default value for this option.

-params *param_list*

Specifies the list of parameter triplets for the specified DFT design. The valid format of a parameter triplet is as follows:

param_name param_type param_value

The *param_name* specifies the name of the parameter, *param_type* specifies the type of the parameter, and *param_value* specifies the value of the parameter. Valid values for *param_type* are **integer**, **float**, **boolean**, and **string**.

These parameters are stored on the specified DFT design for use with DFT insertion.

There is no default value for this option.

DESCRIPTION

The **define_dft_design** command characterizes a design as a DFT design of a certain type for use with DFT insertion. Ensure that the specified design functionality matches the standard design of the specified type. The specified design is used to instantiate an IP instead of a DesignWare IP during DFT insertion.

EXAMPLES

The following example instructs **insert_dft** to use the *my_des* design as a WC_D1 for wrapper insertion:

```
prompt> read_db my_des.db
```

```
1
```

```
prompt> define_dft_design -design_name my_des -type WC_D1 \  
-interface {shift_clk capture_clk h capture_en capture_en \  
h shift_en shift_dr h cti si h cto so h cfi data_in \  
h cfo data_out h}
```

```
1
```

The following example instructs **insert_dft** to use the *my_des1* design as a BC4 for BSD insertion:

```
prompt> read_db my_des1.db
```

```
1
```

```
prompt> define_dft_design -design_name my_des1 -type BC4 \  
-interface {capture_clk cap_clk h capture_en cen h \  
shift_dr shift h si ti h so to h data_in di h data_out do h}
```

```
1
```

The following example instructs **insert_dft** to use the *ip_pad_des1* design as an input pad design for BSD insertion:

```
prompt> read_db ip_pad_des1.db
1

prompt> define_dft_design -design_name ip_pad_des1 -type PAD \
    -interface {port di h data_out do h} \
    -params {$pad_type$ string input}
1
```

The following example instructs **insert_dft** to use the *op_pad_des1* design as a tristate output pad design with an inverted enable pin for BSD insertion:

```
prompt> read_db op_pad_des1.db
1

prompt> define_dft_design -design_name op_pad_des1 -type PAD \
    -interface {port do h data_in di h enable en l} \
    -params {$pad_type$ string tristate_output}
1
```

The following example instructs **insert_dft** to use the *bidi_pad_des1* design as a bidirectional differential pad design library cell with observe/high/low pins for BSD insertion:

```
prompt> read_db bidi_pad_des1.db
1

prompt> define_dft_design -design_name bidi_pad_des1 -type PAD \
    -interface {port ZP h port ZM l data_in di h \
    data_out do h observe IE h high TM h low TE h} \
    -params {$pad_type$ string bidirectional \
    $differential$ string true $lib_cell$ string true}
1
```

The following example instructs **insert_dft** to use the *ip_op_pad_des1* design as an differential input as well as an differential output pad design for BSD insertion:

```
prompt> read_db ip_op_pad_des1.db
1

prompt> define_dft_design -design_name ip_op_pad_des1 -type PAD \
    -interface {port dop h port don l data_in di h} \
    -params {$pad_type$ string output $differential$ string true}
1
prompt> define_dft_design -design_name ip_op_pad_des1 -type PAD \
    -interface {port dip h port din l data_out do h} \
    -params {$pad_type$ string input $differential$ string true}
1
```

The following example instructs **insert_dft** to use the *ip_op_pad_des1* design as a differential hybrid pad design for BSD insertion:

```
prompt> read_db ip_op_pad_des1.db
```

1

```
prompt> define_dft_design -design_name ip_op_pad_des1 -type PAD \  
-interface {si si_pin H so so_pin H shift_dr shift_dr_pin H} \  
-params {$pad_type$ string hybrid $differential$ string true \  
$diff_port_pairs$ list string \  
"dop don" "dip din" $end_list$ \  
$bsr_segment$ list string \  
"0 BC1 - dop X -" \  
"1 BC4 dip - X -" "2 BC2 dip - X -" "3 BC4 din - X -" \  
$end_list$}
```

1

SEE ALSO

```
insert_dft(2)  
preview_dft(2)  
remove_dft_design(2)  
report_dft_design(2)
```

define_dft_partition

Defines a design DFT partition to be created during DFT synthesis.

SYNTAX

```
status define_dft_partition  
design_partition_label  
[-include list_of_cells_or_references]  
[-clocks list_of_clocks]  
[-rising_edge_clocks list_of_clocks]  
[-falling_edge_clocks list_of_clocks]  
[-default define_default_partition]
```

Data Types

<i>design_partition_label</i>	string
<i>list_of_cells_or_references</i>	list
<i>list_of_clocks</i>	list

ARGUMENTS

design_partition_label
Specifies a user-defined text string containing only alphanumeric characters (a-z, A-Z, 0-9) and the underscore (_).

-include *list_of_cells_or_references*
Specifies the list of cells or design references that belongs to the defined partition.

-clocks *list_of_clocks*
Specifies the list of clocks that belongs to the defined partition.

-rising_edge_clocks *list_of_clocks*
Specifies the list of rising clocks that belongs to the defined partition.

-falling_edge_clocks *list_of_clocks*
Specifies the list of falling clocks that belongs to the defined partition.

DESCRIPTION

The **define_dft_partition** command allows you to define a specific design partition during DFT synthesis. The partition is referenced to the top level in a top-down flow.

The partition information is then used during the DFT architect and insertion process, while running the **preview_dft** or the **insert_dft** command. Each design partition will be DFT inserted based on user constraints. DFT insertion is performed only once for all partitions previously defined by users.

EXAMPLES

The following example defines 2 design partitions for use in a DFT insertion. The design contains at the top level 4 hierarchical cells named U1, U2, U3 and U4, respectively.

```
prompt> define_dft_partition part_1\  
        -include {U1 U2}  
  
prompt> define_dft_partition part_2\  
        -include {U3 U4}
```

The following example defines 2 design partitions for use in a DFT insertion. The design contains at the top level 4 hierarchical cells. The cells U1 and U2 are instantiated from the ref1 design reference and cells U3 and U4 are instantiated from design reference ref2:

```
prompt> define_dft_partition part_1\  
        -include {ref1}  
  
prompt> define_dft_partition part_2\  
        -include {ref2}
```

SEE ALSO

```
current_dft_partition(2)  
insert_dft(2)  
preview_dft(2)  
remove_dft_partition(2)  
report_dft_partition(2)  
set_dft_signal(2)  
set_scan_compression_configuration(2)  
set_scan_configuration(2)  
set_scan_path(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 a 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
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_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_test_mode

Defines a test mode to be created during DFT synthesis.

SYNTAX

```
status define_test_mode  
test_mode_label  
[-usage mode_purpose]  
[-encoding {port_name 0 | 1, ...}]  
[-inherit parent_mode_name]
```

Data Types

<i>test_mode_label</i>	string
<i>mode_purpose</i>	string
<i>parent_mode_name</i>	string

ARGUMENTS

test_mode_label
Specifies a user-defined text string containing only alphanumeric characters (a-z, A-Z, 0-9) and the underscore (_).

-usage *mode_purpose*
Specifies the mode type for a test mode. Substitute a mode type for *mode_purpose*. The supported mode types are: **scan_compression**, **scan**, **wrp_if**, **wrp_of**, **wrp_safe**, **bist**, **bist_controller** and **bist_scan**.
scan_compression - This is the DFT MAX ScanCompression_mode of operation. In this mode, the internal scan chains are driven by the load compressors and the scan chains drive the unload compressors. This mode is used for testing all logic internal to the core.
scan - This is the basic scan mode operation. The scan chains are driven directly by top-level scan-in ports and they in turn drive top-level scan-out ports. This mode is used for testing all logic internal to the core. This is the default mode if none is specified.
wrp_if - This is the inward facing, or **INTEST** mode of wrapper operation. This mode is used for testing all logic internal to the core. In this mode, wrappers are enabled and configured to drive and capture data within the design in conjunction with the internal scan chains.
wrp_of - This is the outward facing, or **EXTEST** mode of wrapper operation. This mode is used for testing all logic external to the design. Wrappers are enabled and configured to drive and capture data outside of the design. In this mode the internal chains are disabled.
wrp_safe - This is the safe wrapper mode. In this mode the internal chains are disabled and the internal core is protected from toggle activity. This mode is optional and provides isolation of the core while other cores are being tested. When active, safe mode enables driving steady states into or out of the design.
bist - In this mode, the internal scan chains are driven by SoCBIST PRPG and these chains drive the MISR. It is the main test mode used by SoCBIST. This mode is used for testing all logic internal to the core.
bist_controller - This mode of SoCBIST operation is for testing the BIST

controller logic. When in this mode, all wrapped cores are in the EXTEST mode.

bist_scan - This mode of SoCBIST operation is a traditional basic scan mode operation. The scan chains are driven directly by top-level scan-in ports and they in turn drive top-level scan-out ports. This mode is used for testing all logic internal to the core.

-encoding {port_name 0 | 1, ...}

Specifies a list of ports and the binary values that select a particular test mode. The encoding argument consists of a port name and a binary value pair that specify the test mode port and the bit values to be used in a particular test mode. The encoding port pairs for a test mode must contain all pairs to be used in the design. See the EXAMPLES section for more information.

Before a port can be used for a test mode port, it must be defined as a test mode port using the **set_dft_signal** command:

set_dft_signal -type TestMode

DESCRIPTION

The **define_test_mode** command allows you to define a test mode for DFT synthesis. Inference for existing scan multimode designs is not supported at this time.

The information associated with a *spec* mode is interpreted as a specification of a test mode to be synthesized. This information is used during the DFT architect and insertion process, while running the **preview_dft** or the **insert_dft** command. The specification information for a mode consists of protocols, port attributes, and the configuration information specific to the DFT to be inserted.

Perform multimode specifications in the following order:

1. Define TestMode signals with **set_dft_signal -test_mode all**.
2. Define all test modes, their usage, and their encoding with **define_test_mode**.
3. Define clocks, asynchronous signals, and constants common to all test modes with **set_dft_signal -test_mode all**.
4. Define mode-specific signals with **set_dft_signal -test_mode test_mode_name**.
5. Specify scan paths to be used in all test modes with **set_scan_path -test_mode all**.
6. Specify scan paths to be used in specific test modes with **set_scan_path -test_mode test_mode_name**.

Note that when the **define_test_mode** command is used, it will set the **current_test_mode**. Then if any of the commands listed below are used without the **-test_mode** option, the spec will not be applied to all test modes. Instead, the spec will be applied only to the mode that was last defined with **define_test_mode**. The affected commands are

- **set_scan_configuration**
- **set_scan_path**
- **set_dft_signal**
- **set_scan_configuration_compression**

define_test_mode

Be sure to use the **-test_mode** option with all of these commands once the **define_test_mode** command has been issued. If the spec is to be applied to all modes use **-test_mode all**. If the spec is to be applied to a specific test mode, use **-test_mode test_mode_name**.

Use the **remove_test_mode** command to remove a test mode and all associated information.

EXAMPLES

The following example defines 3 modes for use in a DFT MAX insertion. The modes are ScanCompression_mode, Internal_scan1, and Internal_scan2. The encoding values are also specified.

```
prompt> set_dft_signal -view spec -type TestMode \  
        -port [list i_trdy_de i_trdy_dd i_cs]  
  
prompt> define_test_mode ScanCompression_mode \  
        -usage scan_compression \  
        -encoding {i_trdy_de 0 i_trdy_dd 0 i_cs 1}  
  
prompt> define_test_mode Internal_scan1 -usage scan \  
        -encoding {i_trdy_de 0 i_trdy_dd 1 i_cs 0}  
  
prompt> define_test_mode Internal_scan2 -usage scan \  
        -encoding {i_trdy_de 0 i_trdy_dd 1 i_cs 1}
```

The following example defines 9 wrapper modes:

```
prompt> set_dft_signal -view spec -type TestMode \  
        -port [list i_trdy_de i_trdy_dd i_cs]  
  
prompt> define_test_mode burn_in -usage scan \  
        -encoding {i_trdy_de 0 i_trdy_dd 0 i_cs 0 i_wr 1}  
  
prompt> define_test_mode domain -usage scan \  
        encoding {i_trdy_de 0 i_trdy_dd 0 i_cs 1 i_wr 0}  
  
prompt> define_test_mode my_wrp_if -usage wrp_if \  
        encoding {i_trdy_de 0 i_trdy_dd 0 i_cs 1 i_wr 1}  
  
prompt> define_test_mode my_wrp_if_delay -usage wrp_if \  
        -encoding {i_trdy_de 0 i_trdy_dd 1 i_cs 0 i_wr 0}  
  
prompt> define_test_mode my_wrp_if_scl_delay -usage wrp_if \  
        -encoding {i_trdy_de 0 i_trdy_dd 1 i_cs 0 i_wr 1}  
  
prompt> define_test_mode my_wrp_of -usage wrp_of \  
        -encoding {i_trdy_de 0 i_trdy_dd 1 i_cs 1 i_wr 0}  
  
prompt> define_test_mode my_wrp_of_delay -usage wrp_of \  
        -encoding {i_trdy_de 0 i_trdy_dd 1 i_cs 1 i_wr 1}
```

```
prompt> define_test_mode my_wrp_of_scl_delay -usage wrp_of \  
         -encoding {i_trdy_de 1 i_trdy_dd 0 i_cs 0 i_wr 0}  
  
prompt> define_test_mode my_wrp_safe -usage wrp_safe \  
         -encoding {i_trdy_de 1 i_trdy_dd 0 i_cs 0 i_wr 1}
```

SEE ALSO

```
current_test_mode(2)  
insert_dft(2)  
list_test_modes(2)  
preview_dft(2)  
remove_test_mode(2)  
set_dft_signal(2)  
set_scan_compression_configuration(2)  
set_scan_configuration(2)  
set_scan_path(2)
```

define_user_attribute

Defines a new user-defined attribute.

SYNTAX

```
int define_user_attribute  
-type data_type  
-classes class_list  
[-quiet]  
attr_name  
[-range_max max]  
[-one_of values]  
[-range_min min]  
[-import]
```

Data Types

<i>data_type</i>	string
<i>class_list</i>	values
<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**.

-classes *class_list*
Specifies the list of class names for the user-defined *attr_name* attribute. Valid classes are design, port, cell, net, etc.

-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_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
```

SEE ALSO

```
get_attribute(2)  
list_attributes(2)  
remove_attribute(2)  
set_attribute(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)`
`MasterInstance(3)`
`target_library(3)`

dft_drc

Checks the current design against test design rules.

SYNTAX

```
int dft_drc  
[-pre_dft] [-verbose]  
[-coverage_estimate] [-sample percentage]
```

ARGUMENTS

-pre_dft

If specified only pre-DFT rules (D rules) are checked. By default, the state of the design determines the rules that are checked. The state of the design is automatically determined using attributes. For scan routed designs, post-DFT rules are checked otherwise pre-DFT rules are checked.

-verbose

Controls the amount of detail when displaying violations. If specified, every violation instance is displayed. By default, only the first instance and the number of instances are displayed.

-coverage_estimate

Generates test coverage estimate at the end of design rule checking.

-sample *percentage*

Specifies a sample percent of faults to be considered when estimating test coverage. Must be used in conjunction with the **-coverage_estimate** option.

DESCRIPTION

Checks the current design against the test design rules of the scan test implementation specified by the **set_scan_configuration** (**-style** option) command.

Also, if there are unconnected test modes and functional Autofix clocks, then appropriately constrains them.

If design rule violations are found, the appropriate messages are generated.

You should perform test design rule checking on a design before any other DFT Compiler operations such as **insert_dft**.

This command requires the existence of a valid test protocol. The test protocol can be generated by **create_test_protocol** command or read using **read_test_protocol** command.

The severity of violations falls under one of the following categories:

1. Information - no action required
2. Warning - You should analyze the violations. These might violate sequential cells

resulting in their exclusion from scan chains. However they do not prevent you from running certain DFT Compiler commands.

3. Fatal - These violations will stop you from proceeding. Certain DFT Compiler commands cannot be run.

If you start with a design with pre-existing scan structures, the scan info for inference should be specified with `-view` as `existing_dft` for `set_dft_signal` & `set_scan_path` commands and execute `create_test_protocol`.

The `-coverage` option generates test coverage statistics for the current design. The option does not allow you to save or write out test patterns. The number generated here is only an estimate and might be different from the one generated by an ATPG tool.

If you are running the command under the Design Vision environment, you can use the violation browser to analyze violations.

EXAMPLES

The following command performs test design rule checking for the current design, and illustrates the types of messages that are printed.

```
prompt> current_design des1

prompt> set_scan_configuration -methodology full_scan\
        -style multiplexed_flip_flop

prompt> set_dft_signal -view existing_dft -type Constant -port TM1 -
active_state
Accepted dft signal specification.
1

prompt> set_dft_signal -view existing_dft -type Constant -port TM2 -
active_state
Accepted dft signal specification.
1

prompt> set_dft_signal -view existing_dft -type MasterClock -port CLK1 -
timing [list 45 55]
Accepted dft signal specification.
1

prompt> create_test_protocol -infer_asynch
In all_dft mode...

Information: Starting test protocol creation. (TEST-219)
...reading user specified clock signals...
Information: Identified system clock port CLK1 (45.0,55.0). (TEST-265)
...inferring asynchronous signals...
Information: Inferred active low asynchronous control port as1. (TEST-261)
Information: Inferred active low asynchronous control port as2. (TEST-261)
1
```

```

prompt> dft_drc -ver
In all_dft mode...
  Loading test protocol
  Pre-DFT DRC enabled

Information: Starting test design rule checking. (TEST-222)
  ...basic checks...
  ...basic sequential cell checks...
    ...checking for scan equivalents...
  ...checking vector rules...
  ...checking pre-dft rules...

-----
Begin Pre-DFT violations...

  Warning: Clock input CP of DFF ff2 was not controlled. (D1-1)

Pre-DFT violations completed...
-----

-----
DRC Report

  Total violations: 1

-----

1 PRE-DFT VIOLATION
  1 Uncontrollable clock input of flip-flop violation (D1)

Warning: Violations occurred during test design rule checking. (TEST-124)

-----
Sequential Cell Report

  1 out of 2 sequential cells have violations

-----

SEQUENTIAL CELLS WITH VIOLATIONS
  *   1 cell has test design rule violations
      ff2
SEQUENTIAL CELLS WITHOUT VIOLATIONS
  *   1 cell is a valid scan cell
      ff1

Information: Test design rule checking completed. (TEST-123)

```

SEE ALSO

```

current_design(2)
preview_dft(2)
insert_dft(2)
create_test_protocol(2)
read_test_protocol(2)

```

```
set_dft_signal(2)
set_scan_path(2)
target_library(3)
```

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

<i>net</i>	Specifies the net name or net instance name to disconnect. A net must exist in the current design.
<i>object_list</i>	Specifies the pins and ports disconnected from the net. Only pins and ports existing in the current design are specified. Either <i>object_list</i> or -all must be specified.
-all	Specifies to break all connections on the net. Either -all or <i>object_list</i> 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.

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:

```
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)
```

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)`

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

elaborate

Builds a design from the intermediate format of a Verilog module, a VHDL entity and architecture, or a VHDL configuration.

SYNTAX

```
status elaborate
design_name
[-library library_name
 | -work library_name]
[-architecture arch_name]
[-parameters param_list]
[-file_parameters file_list]
[-update]
[-ref]
```

Data Types

<i>design_name</i>	string
<i>library_name</i>	string
<i>arch_name</i>	string
<i>param_list</i>	list
<i>file_list</i>	list

ARGUMENTS

design_name
Specifies the name of the design to build. The design can be a Verilog module, a VHDL entity, or a VHDL configuration.

-library *library_name*
Specifies the library name that **work** is to be mapped to. By default, **elaborate** looks in the **work** library for the design to be built. Specifying **-library** allows you to temporarily change the library that **work** is mapped to.

-work *library_name*
Specifies an alias used for **-library**.

-architecture *arch_name*
Specifies the name of the architecture. In VHDL, the default architecture is the most recently analyzed architecture.

-parameters *param_list*
Specifies a list of design parameters enclosed in quotes. Parameters within the list must be separated by commas. A specification can be based on parameter order (for example, "8,7,5") or on parameter names (for example, "N=>8,M=>6"). It is acceptable to mix ordered and named parameter specifications, as long as the ordered parameters are listed first. The full *param_list* syntax is summarized in the DESCRIPTION section below.

-file_parameters *file_list*
Specifies a list of files that contain parameter specifications. The

specifications are appended to the parameters listed in the **-parameters** option (if present). The syntax of the parameter file is identical to the syntax between the parameter-delimiters (#) in the **-parameters** option, except that the backslashes (\\) at the end of lines and the hashes (#) must be omitted. The specified files are searched for in the `search_path`.

-update

Reanalyzes out-of-date intermediate files if the source can be found.

-ref

Elaborates several reference designs that contain bottom design instantiations in the elaborated module, using a hierarchical elaboration flow. When the top design is instantiated, it includes all the linkage information, including interfaces, modports, and parameters, and the logic for the lower-level designs.

DESCRIPTION

The **elaborate** command builds a design from its intermediate representation using the specified parameters.

The syntax of the parameter specifications includes a subset of VHDL syntax plus the 'h constant format of Verilog. VHDL accepts all but the 'h format. Verilog accepts strings, integers, and 'h values. VHDL also allows the concatenation operator ampersand (&) between two arrays or strings. Concatenating an array with a scalar is not supported. Define each parameter only once; an error message is generated if a parameter is defined more than once.

Parameters are not case sensitive, so NUM is the same as num.

For the **-parameters** option (but not in files specified by **-file_parameters**), the parameter specification must be specially delimited. The parameter specification can be enclosed in double quotation marks ("), but this is not advised if the specification itself includes string values. Instead, use the special parameter-delimiter, which brackets the parameter specification syntax. The parameter-delimiter starts and ends with '#' (hash). For multi-line strings put backslashes (\\) as the last character in the line. This is because "#" is properly recognized as a string constant.

Spaces could be inserted in the middle of the parameter string for user's script easy of readenness, but then the parameter string must be enclosed in double quotation marks ("), or the spaces should be properly escaped using backslashes (\\).

In the following example, the same parameter is passed three times, one without intermediate spaces and two using the appropriate syntax. All three cases are equivalent.

```
prompt> elaborate TOP -param width=>32,ports=>8
prompt> elaborate TOP -param "width => 32, ports => 8"
prompt> elaborate TOP -param #width\\ =>\\ 32,\\ ports\\ =>\\ 8#
```

In the following example, the second # in the first line does not terminate the

parameter.

```
prompt> elaborate MY_DES -param #words=>4,str => "#",\
      name=>"my_name",\
      matrix=>"1010101010001"&\
      "0001010101010"#
```

The formal syntax for the parameter specification is as follows:

```
parameter_list:  parameter_spec [, parameter_spec]*

parameter_spec:  parameter_name => value_spec
                  parameter_name = value_spec
                  value_spec

value_spec:      " string_value "
                  ' character_value '
                  integer_value
                  enum_value
                  ( value_spec [, value_spec]* )
                  array_value_spec & array_value_spec
                  integer_value 'h' hex_value
```

When the **-ref** option is specified, the **elaborate** command scans the design and groups the bottom design instantiations into several reference designs. Each reference design contains all of the instantiations of a bottom design type.

For example, if a 'top' design contains several instances of bottom designs, name 'bot1' and 'bot2', the **-ref** option generates two reference designs, 'top_bot1' and 'top_bot2', that contain all of the instantiations of each type of lower design, respectively. The instance parameters and SV interface specializations for SystemVerilog designs are preserved in the reference design information.

Later, the reference design can be loaded and linked, forcing the elaboration of all of the specialized information required from the top design. The already elaborated bottom designs can be used in a hierarchical elaboration flow.

For more information about the hierarchical elaboration flow for SystemVerilog designs with interfaces, see the *SystemVerilog User Guide*.

EXAMPLES

The following example builds the design *mult* with *N* parameter set to 8, and *M* set to 3. The **-update** switch causes all out-of-date HDL files to be automatically reanalyzed.

```
prompt> elaborate -update mult -parameters "N=8,M=3"
Information: Re-analyzing the out of date package 'USER_TYPES'. (LBR-15)
Information: Re-analyzing the out of date entity 'MULT'. (LBR-15)
Information: Re-analyzing the out of date architecture 'STRUCTURE(MULT)'. (LBR-15)
```



```
prompt> write -f ddc -hier -o bottom.ddc {bottom_M64 bottom_N32_M16}
Writing ddc file 'bottom.ddc'.
1
```

SEE ALSO

```
analyze(2)
define_design_lib(2)
list_designs(2)
read_ddc(2)
report_design_lib(2)
write(2)
hdlin_auto_save_templates(3)
template_naming_style(3)
template_parameter_style(3)
template_separator_style(3)
```

encrypt_lib

Encrypts a VHDL source library file.

SYNTAX

```
int encrypt_lib  
file_name  
[-output encrypted_file]  
[-format {vhdl | tf | mwlib}]  
[-key string]
```

Data Types

<i>file_name</i>	string
<i>encrypted_file</i>	string

ARGUMENTS

file_name
Specifies the name of the VHDL file to encrypt.

-output *encrypted_file*
Specifies the output filename to which the encrypted output is to be written. By default, the output file is named *file_name.E* and is written to the current directory.
NOTE: If you specify an output filename that already exists, the command overwrites the file without warning.

DESCRIPTION

This command encrypts a VHDL source library file to protect ASIC vendors' VHDL models. The encrypted file is read and simulated by **vhdlan** and **vhdlsim**, respectively, in the same way as a nonencrypted file.

EXAMPLES

The following example encrypts the technology library *my_file.vhd*. The output file is named *my_file.E* and is written to the current directory.

```
prompt> encrypt_lib my_file.vhd
```

SEE ALSO

```
read_lib(2)  
write_lib(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

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)
```

extract_physical_constraints

Extracts physical constraints information from one or more Design Exchange Format (DEF) files.

This command is supported only in topographical mode.

SYNTAX

```
status extract_physical_constraints  
def_files  
[-verbose]  
[-no_incremental]  
[-exact]
```

Data Types

def_files list

ARGUMENTS

def_files
Specifies one or more DEF files from which the physical constraints information is extracted. This argument is required.

-verbose
Provides more detailed output during the process of extracting the physical constraints.

-no_incremental
Indicates that the command runs in non-incremental mode. By default, the command runs in incremental mode.
In incremental mode, the placement area is extracted based on the current core area and the site rows in the DEF file; while in non-incremental mode, the placement area is extracted based on the site rows in the DEF file.
In non-incremental mode when there are no site rows in the DEF file, other constraints are not extracted, except for pre-routes.

-exact
Specifies that the exact match is performed between the netlist and DEF file for cell and port names. By default, a fuzzy match is performed, with the underlying fuzzy match rule determined by the **set_fuzzy_query_options** command.

DESCRIPTION

The **extract_physical_constraints** command extracts the physical constraints information from one or more DEF files, and applies these constraints to the design. The applied constraints are saved in .ddc format and do not need to be reapplied in a new topographical mode session when the .ddc file is read in. The saved constraints can only be recognized by Design Compiler topographical mode. The saved constraints in .ddc have no effect in other tools, such as Design Compiler wireload

mode, Physical Compiler, or IC Compiler.

Note that not all information in DEF files is extracted and applied to the design. Only the following types of information are extracted and applied:

- Placement area

Placement Area is computed based on site array definition and applied to the design.

- Port location

Each port with location and port shape in the DEF file has the location and port shape set on the corresponding port in the design. Ports that do not exist in the design or without locations have no effect on the design.

- Cell location

Each cell with a location and the FIXED attribute in the DEF file has the location set on the corresponding cell in the design. Cells without a location or the FIXED attribute, or that do not exist in the design have no effect on the design.

- Placement blockage

Placement blockages are extracted as placement keepouts and set on the design. For a single rectangle placement blockage in the DEF file, one placement keepout is created on the design. For placement blockages with multiple rectangles, one placement keepout is created for each rectangle. All created placement keepouts are hard keepouts.

- Site rows

The site rows commands are extracted and applied if they exist in the DEF file.

- Bounds

If regions are defined in the DEF file, the placement bounds are extracted. Also, if there are cells in the related group attached to the region, then the cells are fuzzy matched with the ones in design, and the matched cells are attached to the bounds in the following ways:

- If there are regions in the design with the same name in the DEF file, in incremental mode, the cells in the related group are attached to the region.
- Otherwise, the region is created with the same name as in the DEF file, the matched cells in the related group are also attached.

- Wiring keepouts

Wiring keepouts are extracted in a manner similar to how placement blockages are extracted.

- Pre-routes

Pre-routes are extracted from DEF and applied by default.

To check which physical constraints were applied to the design, use the

report_physical_constraints command. To generate a Tcl script of the physical constraints use the **write_physical_constraints** command

EXAMPLES

The following example shows how to extract the physical constraints from the DEF file named in.def

```
prompt> extract_physical_constraints in.def
```

SEE ALSO

```
create_bounds(2)  
create_net_shape(2)  
create_placement_blockage(2)  
create_site_row(2)  
create_wiring_keepouts(2)  
report_physical_constraints(2)  
set_fuzzy_query_options(2)  
write_physical_constraints(2)  
update_bounds(2)
```

extract_rc

Executes 2.5D extraction for routes in a design.

SYNTAX

```
status extract_rc  
[-estimate]
```

ARGUMENTS

-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.

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.

Limitations

The command is only available in DC-Topographical mode.

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
```

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)`

filter

Returns a list of design objects that satisfy a conditional attribute expression.

SYNTAX

```
list filter  
object_list  
expression  
[-regexp]  
[-nocase]
```

Data Types

<i>object_list</i>	list
<i>expression</i>	string

ARGUMENTS

object_list
Specifies a list of design or library objects to filter.

expression
Specifies a conditional expression used to filter the object list in which attribute names are preceded by @ (ampersand). For examples of typical expressions, refer to the EXAMPLES section. For a complete description of conditional expressions, refer to the dc_shell manual page.

-regexp
Indicates that the real regular expressions are to be used by the =~ and !~ filter operators. By default, the =~ and !~ filter operators use simple wildcard pattern matching with the '*' and '?' wildcards.

-nocase
When used with the **-regexp** option, the pattern matching is case-insensitive. This option cannot be used without the **-regexp** option.

DESCRIPTION

The **filter** command selectively searches a group of objects and returns only those objects for which the conditional expression is true. The return of an empty list indicates that there are no objects in *object_list* for which the conditional expression is true. To use **filter** with Boolean values, the correct syntax is **true** and **false**. For a list of Synopsys attributes that can be used with **filter**, refer to the *Design Compiler Command-line Interface Guide*.

The basic form of the conditional expression is a series of relations joined together with && and || operators. Parentheses are also supported. The basic relation contrasts an attribute name with a value through a relational operator. For example:

```
is_hierarchical == true && area <= 6
```

The relational operators are as follows:

```
==    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 as follows:

- String attributes can be compared with any operator.
- Numeric attributes cannot be compared with pattern match operators.
- Boolean attributes can only be compared with == and !=. The value can be only true or false.

Existence relations determine if an attribute is defined or not defined for the object. For example:

```
sense == setup_clk_rise && defined(@sdf_cond)
```

The existence operators are as follows:

```
defined
undefined
```

These operators apply to any attribute as long as it is valid for the object class.

For a complete description of conditional expressions, refer to the *PrimeTime Reference Manual*.

Regular expression matching is the same as in the Tcl **regexp** command. When using **-regexp**, use care when you quote the filter *expression*. Using rigid quoting around regular expression pattern is recommended. See the EXAMPLES section for more details. 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.

EXAMPLES

The following example uses the **filter** command to return the bidirectional ports of the current design:

```
prompt> filter [get_ports *] \
           "@port_direction == inout"
{B1 B2}
```

The following example finds all of the designs in the system that are programmable logic arrays or state machines:

```
prompt> filter [get_designs *] \
        "@design_type == pla || @design_type == fsm"
{CONTROL COUNTER}
```

The following example finds all of the nets for which **dont_touch** is set to **true**:

```
prompt> filter [get_nets *] \
        "@dont_touch == true"
{N1 N9}
```

The following example finds all ports from in[18] to in[23]. The placement of double quotes and backslashes is necessary to avoid syntax errors.

```
prompt> filter [get_ports] \
        -regexp {@name=~ "in\[([1][8-9]|[2][0-3])\]" }
{in[23] in[22] in[21] in[20] in[19] in[18]}
```

The following example finds the U2 and U4 cells, using the **-nocase** option:

```
prompt> filter [get_cells] -regexp \
        -nocase {@name =~ u[2,4]}
{U2 U4}
```

The following example uses wildcard characters to match the name. Wildcard characters can be used only with the =~ and !~ operators.

```
prompt> filter [get_cells] {@name =~ U?}
{U1 U2 U3 U4}
```

The following example finds all cells in the system that have the ****logic_0**** reference name. The ***** and **?** characters have special meanings when used with the =~ and !~ operators, allowing you to use the == operator for an exact match.

```
prompt> filter [get_cells] {@ref_name == **logic_0**}
```

SEE ALSO

```
dc_shell(1)
current_design(2)
filter_collection(2)
get_attribute(2)
remove_attribute(2)
set_attribute(2)
verbose_messages(3)
```

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)`

find

Finds a design or library object.

SYNTAX

```
list find  
type  
[name_list]  
[-hierarchy]  
[-flat]
```

Data Types

<i>type</i>	one-of-string
<i>name_list</i>	list

ARGUMENTS

type

Specifies the type of object to be found. The value of *type* can be any of the following:

design	clock
port	reference
cell	net
pin	cluster
rp_group	library
lib_cell	lib_pin
multibit	operator
module	implementation
file	

The value can also be **physnet**, **physcell**, or **physport** for physical objects.

name_list

Specifies a list of names of design or library objects in dc_shell to find. If you do not specify *name_list*, all objects of the specified type are returned.

-hierarchy

Specifies to return all objects matching *type* and *name_list* within the current design hierarchy. The full name of the object at a particular level must match the patterns. The search is similar to the UNIX **find** command. For example, if there is a cell named *block1/adder*, a hierarchical search finds it using *adder*. If you use the **-hierarchy** option, the *type* must be either **design**, **lib_cell**, **net**, **cell**, or **pin**.

-flat

Specifies that the command finds only objects in leaf cells. You must specify **-flat** with the **-hierarchy** option.

DESCRIPTION

The **find** command locates and returns a list of design or library objects in `dc_shell`. The command returns all of the design or library objects of the given *type* that match the names specified in the *name_list*. If no objects are found, the return value is an empty list (`{}`).

In Tcl-based `dc_shell`, the **find** command returns a collection.

The **find** command is generally used as a parameter to another `dc_shell` command. In this case, the parameters of **find** must be enclosed in parentheses to separate them from the parameters of the other command.

The **find** command is used to resolve name conflicts for commands that accept objects of more than one type. For example, the **set_dont_touch** command accepts design cells, nets, references, clusters, and library cells as arguments. If the current design contains both a net and cell named *interrupt* the name *interrupt* does not specify a unique object. Depending on the type of object expected, *interrupt* can mean the net *interrupt* or the cell *interrupt*. Resolve this explicitly with the following command:

```
prompt> set_dont_touch find (net, interrupt)
```

Naming Conventions

A consistent naming convention is used to refer to design and library objects by all of the commands in `dc_shell`. In most cases, the referenced object exists in the current design and can be identified by using its simple name. More complex cases can be handled by including a library specification prefix. This prefix allows you to reference objects in any library and objects in the current design. The design specification prefix is separated from the simple object name with a / (slash).

In the following example, the *MUX51HP* cell in the *lsi_10k* library is returned:

```
prompt> find cell lsi_10k/MUX51HP
{A}
```

You can uniquely identify any design or library object in `dc_shell` by also specifying the design file in the design specification prefix. This portion of the name is delimited by a : (colon). Do this only when you have two designs (or libraries) with the same name that originated from different file names, as in the following example:

```
prompt> find port /project/ADDER.db:ADDER/A
{A}
```

Objects are subordinate to a design or library object. The types of objects that are contained in a design object are ports, references, clocks, cells, nets, pins, clusters, and Relative Placement (RP) groups. The types of objects contained in a library object are `lib_cells`, modules, implementations, and pins.

To find an object that might exist in a design or a library, a search order is

implied. The following example shows the search procedure that **find** uses to locate objects.

```
prompt> find cell ADDER/A
```

To determine the object that **find** returns depends on a two step search:

1. Search the current design for a cell named *ADDER/A*. If one exists, return it.
2. Extract the library specification and search for the library *ADDER*. Search the *ADDER* library for a cell named *A*. If one exists, return that cell name. Otherwise, return an empty list.

You can use the wildcard character ***** (asterisk) to match any number of characters in a name, or you can use **?** (question mark) to match a single character. For example, *u** specifies all object names that begin with the letter *u*. Specifying *u*z* returns all object names that begin with the letter *u* and end with the letter *z*. Specifying *u?v* returns all object names that begin with the letter *u* and end with letter *v* matching any single character in the middle.

Wildcard characters must be escaped using double backslashes (****) to remove their special regular expression meaning. For example, *u*z* specifies any object with the name *u*z*. For more details about escaping wildcards refer to the **wildcards** variable man page.

Implicit Find Command

Commands that accept design or library object arguments perform an implicit **find** for each argument that is not already an explicit find. For example, the **set_input_delay** command accepts only port objects. Therefore, the following two commands are equivalent:

```
prompt> set_input_delay IN1 10
```

```
prompt> set_input_delay find (port, IN1) 10
```

The implicit **find** operation is different for commands that accept more than one object type. In this case, each object type is searched (in order) for a match to the name. The search ends when a match is found. For example, the following command first looks for a cell named *X*, then a net named *X*, next a reference named *X*, and finally a library cell named *X*:

```
prompt> set_dont_touch X
```

The implicit **find** operation is different for commands that operate only on bus objects. In this case, each object type is searched (in order) for a match to the name. It first searches ports, and then searches nets. The search of object types ends when a match is found. If one of the objects found is a bus object, it is among the objects returned in the list. If one of the objects found is not a bus object, it is among the objects returned only if it is not a member of a bus object. For example, the following command first looks for ports with names that begin with the letter *X*, then for nets with names that begin with the letter *X*.

```
prompt> remove_bus X*
```

If there are ports named XJ1, XJ2, XK1, XK2, XL1, and XL2, and there are port buses named XJ (with members XJ1 and XJ2) and XK (with members XK1 and XK2), the objects returned are ports XL1, and XL2, and the port buses XJ and XK. Nets are not searched for because ports are found.

Hierarchical Find Command

The **-hierarchy** option to **find** returns all objects within the hierarchy of the current design that meet the *type* and *name_list* criteria.

The impact of each allowed *type* used in conjunction with the **-hierarchy** option is described as follows:

```
find design -hierarchy
    Returns all design objects referenced within the hierarchy of the current
    design.

find lib_cell -hierarchy
    Returns all lib_cell objects referenced within the hierarchy of the current
    design.

find net -hierarchy
    Returns all instances of nets within the hierarchy of the current design.

find cell
    Returns all instances of cells within the hierarchy of the current design.

find pin
    Returns all instances of pins within the hierarchy of the current design.
    Adding a name_list in each of the previous cases further restricts the
    returned objects to a given name.
```

EXAMPLES

The following example finds the *N1* net in the current design:

```
prompt> find net N1
      {N1}
```

The following example uses the * (asterisk) wildcard character to find all cells in the current design that begin with *U*:

```
prompt> find cell U*
      {U0, U1, U2, U3, U4, U5, U6, U7}
```

The following example finds the *CACHE/MEM* cluster in the current design:

```
prompt> find cluster CACHE/MEM
      {MEM}
```

The following example uses a library prefix to find the *IVA* cell in the *tech_lib* library and sets the **dont_use** attribute:

```
prompt> set_dont_use tech_lib/IVA
```

The following example finds all pins of the *U8* cell:

```
prompt> find pin U8/*
      {U8/A, U8/Z}
```

The following example sets the **dont_touch** attribute for design reference *adder* of the current design:

```
prompt> set_dont_touch find (reference, adder)
```

The following example sets the **dont_touch** attribute for *datapath* cluster of the current design:

```
prompt> set_dont_touch find (cluster, datapath)
```

In the following example, the specified object is not found and a warning is returned:

```
prompt> find port A1
Warning: Can't find port 'A1' in design 'adder'
      {}
```

If you specify multiple names, **find** returns all of the objects found, and a warning is returned for those not found.

```
prompt> find design {add, hadd, fadd}
Warning: Can't find design 'hadd'
      {add, fadd}
```

The following example shows how to prefix a name with a full file specification:

```
prompt> find cell /project/ripple/lookahead/TOP.db:MULTX
      {MULTX}
```

The following examples show the use of **find** where *type* has the value of **file**:

```
prompt> list -files
File                                     Path
----
```

```

        adder.db                /remote/usr4/joeuser/designs
        clock.db                /remote/usr4/joeuser/designs
1

```

```

prompt> find file "adder.db"
        {"adder.db"}

```

```

prompt> find file "mux.db"
        Error: Can't find file 'mux.db'. (UID-109)
        {}

```

```

prompt> remove_design find(file,"clock.db")
        Removing file 'clock.db'
        design 'TOP'
        design 'MUX'
        design 'TIME_BLOCK'
        design 'TIME_STATE_MACHINE'
        design 'TIME_COUNTER'
        design 'TIME_COUNTER_DW01_inc_n6_0'
        design 'ALARM_BLOCK'
        design 'ALARM_COUNTER'
        design 'ALARM_COUNTER_DW01_inc_n6_0'
        design 'ALARM_STATE_MACHINE'
        design 'ALARM_SM_2'
        design 'CONVERTOR_CKT'
        design 'CONVERTOR'
        design 'HOURS_FILTER'
        design 'COMPARATOR'
1

```

If you specify **-hierarchy** using **design** as the value of **type**, all designs referenced within the current design are returned as shown in the following example:

```

prompt> find design -hierarchy
        {"HALF_SUBTRACTOR", "FULL_SUBTRACTOR", "SUBTRACTOR"}

```

The following example returns all instances of cells that begin with the letter *U* in the hierarchy of the current design:

```

prompt> find -hierarchy cell "U*"
        {"U1", "MY_CELL/U1", "MY_CELL/U2", "U1/EXAMPLE/U33"}

```

The following example returns all instances of leaf cells that begin with *U1* in the hierarchy of the current design:

```

prompt> find -flat -hierarchy cell "U1*"
        {"U4/U10", "U4/U11", "U4/U12", "U1/U2/U132", "U1/U2/U131", "U6/U10"}

```

The following example returns all multibit components that begin with *U1* in the current design:


```
prompt> find multibit "U*"
{"U2", "U4_multibit"}
```

The following example in Tcl mode returns and displays a collection using the implicit query property:

```
prompt> find cell U*
{"U1", "U2", "U345"}
```

The following example in Tcl mode sets variable `a` to contain a collection. The implicit query property of **find** in Tcl mode returns the collection, even though the name assigned to the collection is `-sel1`, as shown by using the **echo** command:

```
prompt> set a [find cell U*]
{"U1", "U2", "U345"}
prompt> echo $a
"_sel1"
```

SEE ALSO

```
collections(2)
current_design(2)
get_attribute(2)
get_cells(2)
get_clocks(2)
get_clusters(2)
get_designs(2)
get_generated_clocks(2)
get_lib_cells(2)
get_lib_pins(2)
get_libs(2)
get_multibits(2)
get_nets(2)
get_pins(2)
get_ports(2)
get_references(2)
remove_attribute(2)
report_lib(2)
set_attribute(2)
find_allow_only_non_hier_ports(3)
find_converts_name_lists(3)
wildcards(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
```

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 a list of design or library objects.

SYNTAX

```
list get_attribute  
object_list  
attribute_name  
[-bus]  
[-quiet]
```

Data Types

<i>object_list</i>	list
<i>attribute_name</i>	string

ARGUMENTS

object_list
Specifies a list of the design or library objects for which the attribute value is to be returned. For details on searching for design and library object names, see the man page for the **find** command.

attribute_name
Specifies the name of the attribute whose value is to be returned.

-bus
Returns the value of the attribute that is on the bus as a whole, and not on each individual bus member.

-quiet
Turns off the warning message otherwise issued if the attribute or objects are not found.

DESCRIPTION

This command searches the *object_list* for the specified attribute and returns a list of attribute values. An empty list is returned if the attribute is not found on any of the specified objects.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example shows how a command is used to find the *IN1* port and obtain its rise-drive value.

```
prompt> get_attribute get_ports IN1 rise_drive  
{0.25}
```

The following example shows the warning that is issued because the cell *U9* is not found:

```
prompt> get_attribute get_cells {U1, U9} dont_touch  
Warning: Can't find object 'U9' in design 'example'  
{true}
```

The following example shows how a command is used to obtain the load value of ports whose names begin with *OUT*:

```
prompt> get_attribute get_ports OUT* load  
{1.0, 2.0, 2.5, 1.0}
```

The following example shows how a command returns area values for gates *G1* and *G2* in the library *tech_lib* are returned:

```
prompt> get_attribute {tech_lib/G1, tech_lib/G2} area  
{1.0, 2.0}
```

SEE ALSO

```
collections(2)  
define_user_attribute(2)  
find(2)  
foreach_in_collection(2)  
list_attributes(2)  
remove_attribute(2)  
set_attribute(2)
```

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_cells

Creates a collection of cells from the current design, relative to the current instance.

SYNTAX

```
collection get_cells
[-hierarchical]
[-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

-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* and **-of_objects** 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 named pin or net, a pin collection, a net collection, or a power domain. When the object is a power domain, the cells returned are the root cells of the power domain.

The *patterns* and **-of_objects** 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.

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* or *objects* argument and pass the filter, if specified. If no objects match the criteria, an empty string is returned.

If *patterns* and *objects* 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.

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 removes the wire load model from the i1P and i2 cells.

```
prompt> remove_wire_load_model [get_cells {i1 i2}]  
Removing wire load model from cell 'i1'.  
Removing wire load model from cell 'i2'.  
1
```

SEE ALSO

```
collections(2)  
filter_collection(2)  
get_pins(2)  
list_attributes(2)  
query_objects(2)  
win_select_objects(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_clusters

Creates a collection of one or more clusters.

SYNTAX

```
collection get_clusters
[-quiet]
[-regexp [-nocase]]
[-exact]
[-filter expression]
[-hier]
[-flat]
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. By default, this option is off.

-regexp

Views the *patterns* option 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** options are mutually exclusive; you can choose only one. By default, this option is off.

-nocase

Makes matches case-insensitive when combined with **-regexp**. You can use the **-nocase** option only when you also use **-regexp**. By default, this option is off.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. The **-regexp** and **-exact** options are mutually exclusive; you can choose only one. By default, this option is off.

-filter *expression*

Filters the collection with the value of *expression*. For any clusters that match the *patterns* value, the expression is evaluated based on the attributes of the cluster. If the expression evaluates to *true*, the design is included in the result. By default, this option is off.

-hier

Specifies hierarchical. By default, this option is off.

-flat

Specifies flat. By default, this option is off.

patterns

Matches cluster names against patterns. Patterns can include the wildcard character ***** (asterisk) and **?** (question mark), or regular expressions based on the **-regexp** option. By default, this option is off.

-of_objects *objects*

Get cells related to these objects. By default, this option is off.

DESCRIPTION

This command creates a collection of clusters that match certain criteria. It returns a collection handle (identifier) if any clusters match the *patterns* or *objects* and pass the filter (if specified). If no objects match your criteria, the command returns an empty string.

You can use the **get_clusters** 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_clusters** result to a variable.

When issued from the command prompt, the **get_clusters** 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 using the **collection_result_display_limit** variable.

The "implicit query" property of **get_clusters** provides a fast, simple way to display clusters 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_clusters** as an argument to the **query_objects** command. Add the **-query** option to emulate the behavior of **query_objects**.

For information about collections and querying objects, see the **collections** man page.

Multicorner-Multimode Support

This command is not supported in a multi-scenario design flow.

EXAMPLES

The following example queries the clusters that begin with "out". Although the output looks like a list, it is only a display.

```
prompt> get_clusters out*
{"out_reg"}
```

SEE ALSO

collections(2)


```
filter_collection(2)
query_objects(2)
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_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

```
analyze(2)  
define_design_lib(2)  
elaborate(2)  
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]
[-regexp]
[-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.

-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.

-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_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
[-nocase]
[-exact]
[-regexp]
```

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_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. 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.

-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. This option works the same as the **filter** command in dc_shell-t.

-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_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_multibits

Creates a collection of one or more multibits loaded into `dc_shell`. You can assign these multibits to a variable or pass them into another command.

SYNTAX

```
string get_multibits  
[-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
Uses the *patterns* argument as real regular expressions rather than simple wildcard patterns.

-nocase
Makes matches case-insensitive.

-filter *expression*
Filters the collection with *expression*. For any multibits that match *patterns*, the expression is evaluated based on the multibit's attributes. If the expression evaluates to true, the multibit is included in the result. This option works the same as the **filter** command in `dc_shell`.

patterns
Matches multibit 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_multibits** command creates a collection of multibits from those currently loaded into `dc_shell` that match certain criteria. The command returns a collection handle (identifier) if any multibits match the *patterns* and pass the filter (if specified). If no objects matched 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; 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_multibits** 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_multibits** result to a variable.

When issued from the command prompt, **get_multibits** 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_multibits** provides a fast, simple way to display multibits 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_multibits** as an argument to **query_objects**.

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following example queries the multibits that begin with *out*:

```
prompt> get_multibits out*
{out_reg}
```

The following example queries the multibits that begin with *foo* in the design instance *mid1/bot1*:

```
prompt> get_multibits mid1/bot1/foo*
{mid1/bot1/foo_reg}
```

SEE ALSO

```
collections(2)
filter(2)
filter_collection(2)
query_objects(2)
collection_result_display_limit(3)
wildcards(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]
patterns | -of_objects objects
```

Data Types

<i>expression</i>	string
<i>patterns</i>	list
<i>objects</i>	list

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.

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* and **-of_objects** options 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, pin collection, port collection, or cell collection. The *patterns* and **-of_objects** options arguments are mutually exclusive; you can specify only one. You cannot use the **-hierarchical** option with the **-of_objects** option.

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* or *objects* and pass the filter (if specified). If no objects matched the criteria, the command returns an empty string.

If *patterns* and *objects* 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_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 in **dc_shell-t** and only for collections that contain a single object. If the collection contains more than one object, the command issues an error message.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

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_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. For XG mode only: 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. For XG mode only: 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**. For XG mode only: 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_hierarchy

Returns a list of hierarchical cells that are physical hierarchies in the current design.

SYNTAX

integer **get_physical_hierarchy**

DESCRIPTION

The **get_physical_hierarchy** command returns a list of names of hierarchical cells in the current design that are physical hierarchies. These hierarchical cells are set as physical hierarchies through the **set_physical_hierarchy** command.

EXAMPLES

The following example first shows how to set the U1 and U2 hierarchical cells as physical hierarchies and then shows how they are returned by the **get_physical_hierarchy** command.

```
prompt> set_physical_hierarchy {U1 U2}
```

```
prompt> get_physical_hierarchy  
{U1, U2}
```

SEE ALSO

`set_physical_hierarchy(2)`

get_pins

Creates a collection of pins that match the specified criteria.

SYNTAX

```
collection get_pins
[-hierarchical]
[-filter expression]
[-quiet]
[-regex [-nocase] | -exact]
[patterns | -of_objects objects [-leaf]]
```

Data Types

<i>expression</i>	string
<i>patterns</i>	list
<i>objects</i>	collection

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.

-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 **-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* and **-of_objects** arguments are mutually exclusive; you can specify only one.

-of_objects *objects*

Creates a collection of pins connected to the specified objects. Each object is a named cell or net, cell collection, or net collection. The *patterns* and **-of_objects** 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.

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* or *objects* 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* and *objects* 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_ports

Creates a collection of ports from the current design.

SYNTAX

```
collection get_ports
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-filter expression]
patterns | -of_objects objects
[-hierarchical]
```

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 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* and **-of_objects** 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* and **-of_objects** arguments are mutually exclusive; you can specify only one of these arguments.

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* 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*. 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]
[patterns]
[-hierarchical]
```

Data Types

<i>expression</i>	string
<i>patterns</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.
- 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.
- 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.

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 tools 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)
report_power_switch(2)
```

get_references

Creates a collection of one or more references loaded into memory.

SYNTAX

```
collection get_references
[-hierarchical]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-filter expression]
patterns
```

Data Types

<i>expression</i>	string
<i>patterns</i>	list

ARGUMENTS

-hierarchical

Searches for hierarchical cells whose references match the pattern. The full name of the object at a particular level must match the patterns. The use of this option does not force an autolink.

-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.

-filter *expression*

Filters the collection with *expression*. For any references that match *patterns*, the expression is evaluated based on the attributes of the reference. If the expression evaluates to true, the design is included in the result. This option works the same as the **filter** command in **dc_shell**.

patterns

Matches reference 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_references** command creates a collection of cells whose references match certain criteria. If no patterns 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; 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_references** 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_references** result to a variable.

When issued from the command prompt, **get_references** behaves as though **query_objects** has been executed 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_references** provides a fast, simple way to display references 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_references** 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 references that begin with "mpu". Although the output looks like a list, it is just a display.

```
prompt> get_references mpu*
{mpu_0_0 mpu_0_1 mpu_1_0 mpu_1_1}
```

SEE ALSO

```
collections(2)
filter_collection(2)
query_objects(2)
collection_result_display_limit(3)
wildcards(3)
```

get_rp_groups

Creates a collection of relative placement groups.

SYNTAX

```
collection get_rp_groups
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-ignored]
[-top]
[patterns | -of_objects objects]
```

Data Types

<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.

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 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  
[-test_mode test_mode]
```

Data Types

<i>chain_name</i>	string
<i>test_mode</i>	string

ARGUMENTS

```
-chain chain_name  
    Specifies the name of the scan chain.  
  
-test_mode test_mode  
    Specifies a mode name. The command returns only scan cells of this mode name  
    in the Tcl collection. By default, this option is off.
```

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  
[-test_mode test_mode]
```

Data Types

test_mode string

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_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]
[patterns]
[-hierarchical]
```

Data Types

<i>expression</i>	string
<i>patterns</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.
- 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.

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 tools 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)  
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]]
[patterns]
[-hierarchical]
```

Data Types

<i>expression</i>	string
<i>patterns</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.

patterns

Matches supply port names against patterns. Patterns can include the * (asterisk) and ? (question mark) wildcard characters or regular expressions, based on the **-regexp** option. For more details about using and escaping wildcards, see the **wildcards** man page.

-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.

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)
report_supply_port(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_though**, 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_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)

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)  
set_fsm_state_vector(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)
```

group_variable

Adds a variable to the specified variable group. This command is typically used by the system administrator only.

SYNTAX

```
int group_variable  
group_name variable_name
```

Data Types

```
group_name          string  
variable_name      string
```

ARGUMENTS

group_name
Specifies the name of the group that the specified variable is added to. The existing variable group_names are

```
compile_variables  
edif_variables  
hdl_variables  
insert_dft_variables  
io_variables  
multibit_variables  
plot_variables  
schematic_variables  
synlib_variables  
system_variables  
suffix_variables  
view_variables  
vhdl_io_variables  
write_test_variables  
Additional groups can be created with this command.
```

variable_name
Specifies the name of the variable to add.

DESCRIPTION

Adds a variable to a variable group. For descriptions of all of the current variables and variable groups, refer to the individual variable group man pages, which you can access online from dc_shell by entering

```
prompt> help group_name
```

To list all of the currently defined variables, use the command **print_variable_group all**. To list all of the variables in a group, use the command **print_variable_group**

group_name.

EXAMPLES

The following example adds the variable "current_design" to the "system" group:

```
prompt> group_variable system current_design
```

SEE ALSO

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)
insert_dft(2)
remove_clock_gating(2)
report_clock_gating(2)
rewire_clock_gating(2)
set_clock_gating_style(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  
536
```

```

#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)

```

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]  
object_list  
buffer_lib_cell
```

Data Types

<i>new_net_names</i>	list
<i>new_cell_names</i>	list
<i>number</i>	integer
<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.

-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.

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**,

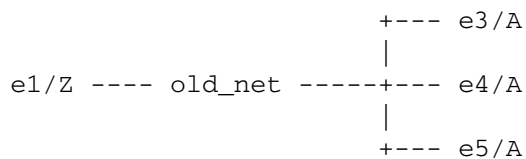
disconnect_net, and **connect_net**, but the **insert_buffer** command provides a more efficient and fail safe way to insert buffers.

The **insert_buffer** command uses the following basic rules to check its arguments for validity:

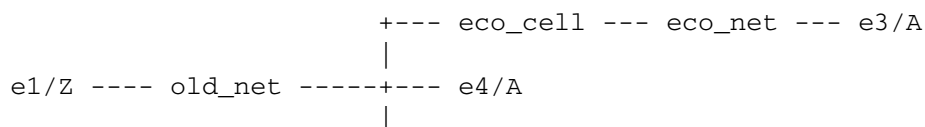
- The pin or port must be in scope at or below the current instance. For a description of some special scoping rules, see the Buffering Inside Boundary Pins section.
- The pin or port must be connected to a net.
- Bidirectional pins cannot be buffered.
- The *lib_cell* cannot be sequential.
- The *lib_cell* 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.



+--- e5/A

- 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.

```

                                     +--- e3/A
                                     |
e1/Z --- old_net --- eco_cell --- eco_net -+--- e4/A
                                     |
                                     +--- e5/A
```

- 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)
remove_buffer(2)
report_cell(2)
report_net(2)
size_cell(2)

```

insert_clock_gating

Performs clock gating on an appropriately-prepared GTECH netlist.

SYNTAX

```
status insert_clock_gating  
[-regular_only]  
[-global]  
[-no_hier]
```

ARGUMENTS

-regular_only
Disables enhanced clock gating during register-based clock gate insertion.

-global
Performs hierarchical clock gating on all subdesigns as one global step.

-no_hier
Limits clock gating to the top level of the current design.

DESCRIPTION

This command performs clock gating on an appropriately-prepared GTECH netlist. To insert or optimize clock gating on a mapped netlist, use the **compile** or **compile_ultra** command with the **-gate_clock** option. You can optionally use the **set_clock_gating_style** command beforehand to specify the structure of the clock-gating circuitry to be inserted. If no style is explicitly set, a default clock-gating style is applied. See the **set_clock_gating_style** command man page for more details on the default clock-gating style.

To control the names of modules, cells and gated clock nets created by this command, you can set different naming style variables listed at the end.

In addition to standard bank clock gating, the **insert_clock_gating** command searches for more clock gating opportunities by combining different register banks that share common enable conditions. This type of enhanced clock gating can result in extra clock gating cells and/or gated registers.

The **insert_clock_gating** command performs clock gating at all levels of hierarchy in the design. By using the **-no_hier** option, the clock gating is limited to only the top level. If you want to perform clock gating on some, but not all, levels of hierarchy, the other levels can be excluded by setting **dont_touch** attributes on those designs or instances.

When the **-no_hier** option is omitted, clock gating is performed on all subdesigns. Every subdesign is processed independently. With the **-global** option, the design and its subdesigns are processed in one single global step. The advantage of the one-step hierarchical clock gating is twofold. First, the subdesigns are analyzed in their context, which is similar to boundary optimization during **compile** command. Second, the clock gates can be inserted at higher levels in the design, allowing for

sharing of registers among different subdesigns. When a clock gate is inserted in a different hierarchical level than the registers that are being gated, new ports will be added to connect the gated clock and enable nets.

Important: when using the **-global** option, the operation is no longer context independent. The functionality of the subdesigns instantiated in the design can be modified in context of the current design. The functionality of the top level (current_design) is preserved. However, when certain subdesigns are also instantiated elsewhere (in a design different from the current design), the functionality of those designs can change. Therefore, it is not recommended to change the current design to that of a lower level and then run **insert_clock_gating -global**. If you choose to do so, you must uniquify the original design before changing the current design and running the **insert_clock_gating -global** command on a subdesign to guaranteed that the functionality of the original design is not altered.

During test (scan shifting), the clock gate has to be surpassed so that each clock pulse is passed onto the register. This is facilitated with a control point insertion. See the **set_clock_gating_style** man page for details. If the clock gates have control points, all of them must be hooked up correctly to the top level test_mode or scan_enable pins using the **insert_dft** command.

EXAMPLES

The commands in the following example show the typical flow for using the **insert_clock_gating** command. After reading or elaborating the design and defining the clock ports, you set the clock-gating style to specify how to perform clock gating. The **insert_clock_gating** command inserts clock gating for the registers.

```
prompt> read_verilog design.v
prompt> current_design top
prompt> link
prompt> set_clock_gating_style -sequential latch
prompt> create_clock clk
prompt> insert_clock_gating
prompt> propagate_constraints -gate_clock
prompt> compile
```

The following example specifies using an integrated cell for the clock-gating circuitry. The **target_library** environment variable also contains the name of the library that contains the integrated cell. This example results in a mapped netlist that has integrated clock-gating cells.

```
prompt> set_clock_gating_style -sequential latch \
    -pos integrated -neg integrated -control_point before
prompt> set target_library {$target_library integrated_cell_library.db}
prompt> read_verilog design.v
prompt> current_design top
prompt> link
prompt> insert_clock_gating
prompt> uniquify
prompt> propagate_constraints -gate_clock
```

```
prompt> compile
```

SEE ALSO

```
compile(2)  
compile_ultra(2)  
create_clock(2)  
insert_dft(2)  
propagate_constraints(2)  
set_clock_gating_registers(2)  
set_clock_gating_style(2)  
uniquify(2)  
power_cg_gated_clock_net_naming_style(3)  
power_cg_cell_naming_style(3)  
power_cg_module_naming_style(3)  
target_library(3)
```

insert_dft

Adds scan circuitry (either internal scan or boundary scan) to the current design.

SYNTAX

```
int insert_dft
```

DESCRIPTION

This command adds internal scan or boundary scan circuitry to the current design.

Scan Synthesis Description

Scan circuitry is chosen from the technology library specified in the **target_library** variable. You must define a target library before issuing the **insert_dft** command.

The **insert_dft** command supports top-down, bottom-up, and middle-out scan insertion methodologies. In the process of adding test circuitry, testable versions of the subdesigns and top-level design are created. Write all new designs to a disk using the **write** command. For testable subdesigns, new designs are created in the database that are distinct from the original subdesigns. The new subdesigns are named according to the value of the **insert_test_design_naming_style** variable. Note that the name of the top-level design is not changed during the **insert_dft** execution.

The **insert_dft** command adds scan signals that use existing ports identified by using the **set_dft_signal** command.

If no such ports are found, new ports are added to the design. The new ports are named according to the following variables:

```
test_mode_port_naming_style  
test_mode_port_inverted_naming_style  
test_clock_port_naming_style  
test_scan_clock_a_port_naming_style  
test_scan_clock_b_port_naming_style  
test_scan_clock_port_naming_style  
test_scan_enable_inverted_port_naming_style  
test_scan_enable_port_naming_style  
test_scan_in_port_naming_style  
test_scan_out_port_naming_style
```

These variables are described on the **insert_dft_variables** man page and on the appropriate individual variable man pages.

The **insert_dft** command processing is described below. First, the command populates a flattened representation of the entire design. Existing logic is marked so scan insertion can detect whether a particular piece of logic has been added. The command also selects a wire load model if you have not yet specified one.

By default, **insert_dft** performs only scan insertion and routing.

If previously generated test design rule checking (DRC) information is not available, **insert_dft** invokes DRC for the specified scan style. DRC generates the clock information that scan insertion retrieves. DRC also violates cells that cannot belong to scan chains.

Next, **insert_dft** architects scan chains into the design. Use the **preview_dft** command to view the proposed architecture without synthesizing it. By default, **insert_dft** constructs as many scan chains as there are clocks and edges. By setting the **-clock_mixing** option of **set_scan_configuration**, the number of scan chains created based on clock can be controlled. set the **-clock_mixing** option to **no_mix**, **mix_edges**, or **mix_clocks**. If you want **insert_dft** to build more scan chains, set the **-chain_count** option of **set_scan_configuration** command. Multiple scan chains are also constructed to reflect the assignment of scan cells to scan chains by using the **set_scan_path** command.

When **insert_dft** constructs multiple scan chains, scan cells are allocated to scan chains based on the following criteria:

- The **set_scan_path** command.
- Clock signals in the design.
- Subdesign scan chains.
- Design hierarchy.
- Names of individual scan cells.

Scan cells are ordered on scan chains based on the following criteria:

- The **set_scan_path** command position.
- Scan.
- Clock trigger times.
- Scan clock domains.
- Scan cell names.

The **insert_dft** command overrides the **set_scan_path** command positions to ensure functional chains if **set_scan_path** is not inserting lock-up latches.

The **insert_dft** command applies a scan equivalence process to all other cells. Scan equivalents are initially selected based on library function identifiers. If no scan equivalent for a storage element is found using this behavior, **insert_dft** uses constraint-based sequential mapping techniques. These techniques consider the logic function implemented by each flip-flop instance to be scan replaced, along with immediately surrounding logic. This consideration generates a set of possible alternatives, from which the best replacement is selected based on **scan_style**, design rule considerations (for example, connection classes and **max_fanout**), and design area. To disable sequential mapping based scan equivalence, set the **insert_test_map_effort_enabled** environment variable to **false**.

With clock gating complete, **insert_dft** ensures that the three-state buses, which do not drive pads, are disabled during scan shift and that bidirectional ports are properly configured during scan shift. **insert_dft** disregards this step if the **set_scan_configuration** command is executed with a **false -disable** option.

The **insert_dft** command avoids adding redundant logic by ignoring buses that already have exactly one active driver and bidirectional ports that are properly configured. It takes into account port **set_test_hold** conditions as well as signal types specified using **set_dft_signal** and options for ports and nets specified using **set_scan_bidi** and **set_scan_tristate**.

Having identified candidate buses, **insert_dft** identifies the disabling logic required. The command iterates through all three-state bus drivers and computes the logic values that must be applied to input pins to disable the bus. It does not succeed in the following situations, when:

- A bus driver belongs to a design that has a **dont_touch** attribute.
- Any of the three-state drivers cannot be disabled.
- Conflicting logic values are required to disable three-state drivers.
- Disabling values conflict with those of previous buses.

The **insert_dft** command then adds generic disabling logic where necessary. It finds and gates all pins that do not hold the values they require during scan shift.

The command processes all bidirectional ports. It discards the following:

- Ports used to apply scan enable signals to the design.
- Degenerated bidirectional ports that are configured as inputs or outputs by constant logic values.
- Ports that do not have exactly one three-state driver that is not in a **dont_touch** design.

The **insert_dft** command establishes the direction to which other ports must be configured during scan shift. Scan outputs are turned outwards, and other scan signals are turned inwards. By default, all other bidirectional ports are turned inwards. To turn bidirectional ports outwards, use the **set_scan_bidi** command with the **-bidi_mode** option.

The **insert_dft** command computes the logic values that must be applied to the driver input pins to configure each port again. It fails if required logic values conflict with previous logic values. The command then adds generic configuration logic where necessary. It finds all pins that do not hold the values they require during scan shift and gates them.

Having disabled three-state buses and configured bidirectional ports, **insert_dft** builds the scan chains if the **set_scan_configuration** command has not been executed with a **-route false** option. For each chain, **insert_dft** determines the scan-in pin. It first looks for **ScanDataIn** signals specified by using the **set_dft_signal** commands. It then looks for design ports with **ScanDataIn** *signal_type* attributes. If it finds no such ports, it creates a dedicated scan-in port for the chain.

Having found the scan-in pin, **insert_dft** jumps over the connected pad if it is not in a **dont_touch** design. It first tries finding a true or inverting path through the pad that is sensitized by test hold, scan enable, and bidirectional control signals, and does not need any additional disabling. If not successful, it gets the vector required to enable a path through the pad and the resulting hookup pin. It finds all pins that do not hold the necessary values during scan shift and gates the pins using generic logic again. Note that **insert_dft** does not have to add three-state disabling logic to ensure that the pad is in input mode: It has been done earlier.

The **insert_dft** command then jumps through input buffers. Starting at the load pin, it jumps over single-fanout inverters and buffers to find a better hookup pin. Scan sense is toggled if the jump causes inversions.

Given the best hookup pin, scan insertion builds the rest of the chain. The preferred routing order, specified by using the **set_scan_path** command, is used to route between serial signals and scan cells. If no preferred order is specified, it uses a default ordering scheme.

If the **set_scan_configuration** command has not been executed with an **-add_lockup false** option, the **insert_dft** command inserts lock-up latches at clock domain boundaries. This compensates for clock skew and ensures a glitch-free scan shift. You can specify to **insert_dft** exactly to add lock-up latches at the end of chain by using the **set_scan_configuration** command with **-insert_terminal_lockup**.

The **insert_dft** command multiplexes into functionally-connected scan segment **ScanDataIn** access pins. It does not recognize a pin as functionally connected if the following conditions are true:

- The pin is not connected to a net.
- The pin has been created by **insert_dft**.
- The pin is a driver and does not have a load, or the pin is connected to a subdesign output port that **insert_dft** created.
- The pin is a load and does not have a driver, or the pin is connected to a subdesign input port that **insert_dft** created.

The **insert_dft** command does not touch optional methodology signal pins that are functionally connected. It reconnects mandatory methodology signal pins that are functionally connected. A warning message appears in this case.

The **insert_dft** command selects the scan-out driver for each cell by using the following criteria:

- It identifies the scan-out signal by looking at `signal_type` attributes. If pins with both `ScanDataIn` and `ScanDataIn` with inverted attributes are present, it picks the noninverted pin, unless the other has a routing position.

- It considers all the equal and opposite drivers that fanout from the scan-out signal and chooses the driver with the most slack. If equal, it chooses the driver with greatest drive strength.

You can set the `test_disable_find_best_scan_out` environment variable to `true` to disable this behavior.

To finish construction, `insert_dft` locates the scan-out port for the chain. If you did not specify a scan-out port, it creates a new one. Otherwise, it avoids inserting redundant multiplexing logic. It first establishes whether there is already a signal path between the last scan-out pin and the scan-out port when scan enable, test hold, and bidirectional control conditions are asserted. If the scan out port is connected to a three-state driver or a pad, it uses the reverse of the method described earlier for input ports to jump back through three-state drivers, pads, buffers, or inverters to a hookup pin. If the this pin is already functionally connected, `insert_dft` adds multiplexing logic and connects the best last scan cell driver to the multiplexer input. If the scan out is on the same net as another port, it isolates the scan out by using a buffer. The reason for this isolation is because some downstream tools cannot handle nets that fanout to multiple ports.

Finally, `insert_dft` routes global signals (including either or both scan enable and test clocks) and applies a default technology mapping to all new generic logic (including disabling logic and multiplexers). It introduces dedicated test clocks for `clocked_scan`, `lssd`, and `aux_clock_lssd` scan styles.

In `lssd`, `aux_clock_lssd` designs, single latches and flip-flops are replaced with their scan equivalences with the "test_scan_clock_b" pins unconnected. If the `test_use_dual_latch_slave_clock` variable is set to `true`, these unconnected "test_scan_clock_b" pins are connected to the slave clocks distributed to the dual latches if they have the same master clocks.

Typically, at this point, the `insert_dft` command has violated compile design rules and constraints and now begins minimizing these violations. If it is executed without the `set_dft_optimization_configuration -none` switch and the design has constraints, it first optimizes the design. Optimizations chosen depend on the setting of the `-map_effort` option using the command `set_dft_insertion_configuration`.

Note that `insert_dft` does not optimize the scan enable net to meet timing constraints.

For `-map_effort low` of command `set_dft_insertion_configuration`, the `insert_dft` command resizes only the logic it added to the design.

For `-map_effort medium` of command `set_dft_insertion_configuration`, `insert_dft` first resizes the logic it added. If constraints are still violated, it applies a set of heuristic optimizations to all critical points in the design. The heuristics include:

- 1) *Sizing*: Size one or more drivers or loads for appropriate drive strength.

- 2) *Phasing*: Replace the driver with one that has an opposing phase.
- 3) *Buffering*: Insert buffers or inverter pairs; transform inverter pairs to buffers, or transform buffers to inverter pairs or buffer inverters.
- 4) *Down-sizing*: Down-size and swap out loads.
- 5) *Isolation*: Isolate noncritical loads using inverter pairs or buffers.
- 6) *Off-loading*: Offload noncritical loads to other driver outputs, use a driver output with a greater drive strength, use drivers with more outputs, or move loads to nets with earlier arrival times.
- 7) *Balancing*: Redistribute loads among drivers, balance buffers, or balance loads across driver outputs.
- 8) *Splitting*: Duplicate driver, distributing loads or dedicating one driver to the critical path.

For **-map_effort_high** option of command **set_dft_insertion_configuration**, the **insert_dft** command resizes the logic it added. If constraints are still violated, it performs critical-path optimization on the entire design. If constraints are now still violated, it applies sequential mapping to sequential elements on critical paths. It then attempts to reduce design area by downsizing cells that are not on a critical path and eliminating extra inverters. It attempts to eliminate any remaining constraint violations by using normal critical-path optimization.

If the **insert_dft** command is not executed with the **set_dft_optimization_configuration -none** the command fixes design rule violations, removing capacitance, transition, and fanout violations. Finally, it completes a last pass at meeting constraints, if specified.

At this point, some cleanup is necessary. The **insert_dft** command removes connection class violations, replaces and merges logic constants, then uniquifies the design. This is because you might want to write out subdesigns that **insert_dft** has modified as instances of new designs. In some cases, where test modifies instances of the same design differently, **insert_dft** produces more than one new design. You might want **insert_dft** to do this only when necessary. It uses a new technique called selective uniquification to produce new designs only when necessary. This technique generates a canonical netlist ID that does not depend on net-names or cell-names. The canonical netlist ID works only on gate-type, net connections, and port-names.

In case of non-uniquified hierarchical design, forced uniquification is performed to record the changes to the design. This is because you might want to write out subdesigns that **insert_dft** has modified as instances of new designs. In some cases, where test modifies instances of the same design differently, **insert_dft** produces more than one new design. You might want **insert_dft** to do this only when necessary.

BSD Synthesis Description

The **insert_dft** command synthesizes ANSI/IEEE Std 1149.1/1149.6-compliant boundary scan circuitry using DesignWare macro cells when you specify the **-bsd enable** option of the **set_dft_configuration** command.

Synthesis is done at the gate level. The newly-synthesized BSD circuitry is mapped. The BSR cells added during BSD insertion are not uniquified. You must run the **uniquify** command after BSD insertion if you need a uniquified design. The **insert_dft** command can use a design with or without core logic, but the pads on the design ports must be present.

If you set **set_bsd_configuration -asynchronous_reset false**, you must specify the four mandatory test access ports (TAPs); otherwise, you must specify all five ports.

Depending on the instructions you selected to implement by using **set_bsd_instruction**, by default the instruction register (IR) is synthesized with the minimum number of bits to accommodate all the instructions. However, you can enforce one-hot encoding of the IR by using **set_bsd_configuration -instruction_encoding one_hot**. The **insert_dft** command synthesizes BSR cells using DesignWare BC_1 on all the design output ports except on bidirectional ports, where a BC_7 is synthesized. On all input ports and for all control points, BC_2 BSR cells are synthesized. If you set system clocks, a BC_4 is synthesized on these design ports. If you used some design ports other than the actual controlling ports to control the pads' controlling logic (for example, an extra controlling port to disable all three-state outputs together), logic is synthesized to allow the actual controlling signal to do the controlling during the boundary scan testing. This removes the need for subsequently setting the extra controlling ports as compliance-enable ports.

It is not possible to synthesize boundary scan circuitry without pads. If the design does not have pads, the command terminates.

Error messages can result from missing TAP ports, missing PADs, or missing functionality of pads. These messages indicate an inability to proceed, and the command exits.

By default, synthesis of boundary scan circuitry is done in compliance with the IEEE1149.1-2001 standard. You can switch back to the IEEE1149.1-1993 standard by setting the **-std {ieee1149.1_1993}** option of the **set_bsd_configuration** command. To implement the IEEE1149.6-2003 standard, specify the **-std {ieee1149.6_2003}** option of the **set_bsd_configuration** command.

By default, synthesis of boundary scan circuitry is done using the new DesignWare tap, DW_tap_uc. To use the old DesignWare tap, DW_tap, instead, set the **bsd_use_old_tap** variable to true.

To use user-defined BSR/TAP designs in BSD synthesis, use the **define_dft_design** command to specify the user-defined designs.

For more details on the IEEE Std 1149.1/1149.6 rules, refer to the *IEEE Std Test Access Port and Boundary-Scan Architecture*.

insert_dft command is also supported in multicorner-multimode (MCMM) technology which is available with Design Compiler Graphical.

Use the **set_dft_location** command to add all DFT logic into a user specified hierarchy during DFT insertion.

EXAMPLES

The following example shows the usage syntax for the **insert_dft** command on the design **WC66** without fixing constraint violations and compile design rules.

```
prompt> current_design WC66
prompt> insert_dft
```

The following command performs synthesis of boundary scan circuitry in compliance with ANSI/IEEE Std 1149.1 on the current design, and illustrates the types of messages that are output.

```
prompt> current_design M1
prompt> set_dft_configuration -scan disable -bsd enable
prompt> set_dft_signal -type tdi -port my_tdi
prompt> set_dft_signal -type tck -port my_tck
prompt> set_dft_signal -type tms -port my_tms
prompt> set_dft_signal -type trst -port my_trst
prompt> set_dft_signal -type tdo -port my_tdo
prompt> create_clock -p 10 my_system_clock
prompt> insert_dft

Loading design 'M1'
...
Generating the TAP
...
Generating the BSR cells
...
Generating the control logic
Writing implemented instructions information to the design
Creating Hierarchy for Boundary Scan Logic ...
...
1
```

SEE ALSO

```
create_test_protocol(2)
current_design(2)
dft_drc(2)
preview_dft(2)
set_dft_configuration(2)
set_dft_insertion_configuration(2)
set_scan_configuration(2)
set_dft_location(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)
```

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  
560
```

```
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)
```

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  
[-remove_sub_design]
```

ARGUMENTS

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 = {. synopsis_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)
```


list_attributes

Lists the currently defined attributes.

SYNTAX

```
status list_attributes
[-application]
[-class class_name]
```

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 **all**, **bound**, **cell**, **clock**, **design**, **ilms**, **lib**, **lib_cell**, **lib_pin**, **net**, **phys_cluster**, **phys_design**, **phys_layer**, **physcell**, **physnet**, **physport**, **pin**, **placement_bound**, **port**, **power_domain**, **power_switch**, **region**, **supply_net**, and **supply_port**.

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 : attribute definition
Design :
```

Attributes:

r - read-only
u - user-defined

Attribute name	Class	Type	Attributes
DDB_EXTRACT_HIGH_FAN_OUT_THRES_ATTR	lib_cell	integer	u
DDB_HIER_TOTAL_UNGROUP_AREA_ATTR	lib_cell	integer	u
DDB_REAL_METALFILL_EXTR_ATTR	lib_cell	integer	u
DDB_VIR_SHIELD_EXTR_ATTR	lib_cell	boolean	u
worst_slack	port	float	u
write_port_name	cell	string	u
write_sched_step	cell	integer	u

1

The following example adds application-defined attributes, but limits the listing to net attributes only:

```
prompt> list_attributes -application -class net
```

Report : attribute definition
-application

Design :

Attributes:

a - application-defined
r - read-only
u - user-defined

Attribute name	Class	Type	Attributes
actual_fall_transition_max	net	float	a
actual_fall_transition_min	net	float	a
actual_max_net_capacitance	net	float	a
actual_min_net_capacitance	net	float	a
actual_rise_transition_max	net	float	a
actual_rise_transition_min	net	float	a
worst_length_attr	net	float	a
x_length	net	integer	a
y_length	net	integer	a

1

SEE ALSO

define_user_attribute(2)
get_attribute(2)
remove_attribute(2)
report_attribute(2)

list_attributes

```
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_duplicate_designs

Lists designs that have the same design name in dc_shell.

SYNTAX

```
int list_duplicate_designs
```

DESCRIPTION

Displays a list of designs currently in dc_shell that have the same name as other designs also currently in dc_shell.

Returns '1' if multiple designs with the same name are detected. Returns '0' if no designs in dc_shell memory share the same design name.

EXAMPLES

In the following example, **list_duplicate_designs** lists two designs that have the same name in dc_shell memory.

```
prompt> list_duplicate_designs
Warning:  Multiple designs in memory with the same design name.

      Design      File      Path
      -----      -
      seq2         A.db      /usr/joe/design
      seq2         B.db      /usr/joe/design
1
prompt>
```

In the following example, **list_duplicate_designs** reports that there are no designs with the same name in dc_shell memory.

```
prompt> list_duplicate_designs
No duplicate designs found.
0
prompt>
```

SEE ALSO

current_design(2)
list_designs(2)
find(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_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  
580
```


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
```

```
Licenses in use:
```

```
    DC-Expert (3)  
    DC-Ultra-Features (3)  
    DC-Ultra-Opt (3)  
    Design-Compiler  
    DesignWare
```

```
1
```

SEE ALSO

```
check_license(2)  
get_license(2)  
license_users(2)  
remove_license(2)
```

list_test_models

Lists the designs with test models available in dc_shell.

SYNTAX

```
status list_test_models
```

ARGUMENTS

The **list_test_models** command has no arguments.

DESCRIPTION

Displays the designs currently in dc_shell that have an associated test model.

EXAMPLES

```
prompt> list_test_models
      des_unit                               /remote/my_dir/des_unit_xtol/db/des_unit.db
      des_unit_SCCOMP_COMPRESSOR             /remote/my_dir/des_unit_xtol/
des_unit_SCCOMP_COMPRESSOR.db
      des_unit_SCCOMP_DECOMPRESSOR           /remote/my_dir/des_unit_xtol/
des_unit_SCCOMP_DECOMPRESSOR.db
1
```

SEE ALSO

list_test_modes

Displays all of the test modes that are defined for the current design.

SYNTAX

```
int list_test_modes
```

ARGUMENTS

None.

DESCRIPTION

The **list_test_modes** command displays all of the modes defined for the current design. The **define_test_mode** command is used to define a mode of operation for a design.

If a design does not have any test modes defined, it displays the message "Design has no test modes."

EXAMPLES

The following is an example of the test modes report.

```
prompt> list_test_modes
Test Modes
=====

      Name: ScanCompression_mode
      Type: InternalTest
      Focus:
      Core des_unit_U_decompressor in decompress mode
      Core des_unit_U_compressor in compress mode

      Name: Internal_scan
      Type: InternalTest
      Focus:

      Name: Mission_mode
      Type: Normal
```

1

SEE ALSO

```
current_design(2)
current_test_mode(2)
define_test_mode(2)
insert_dft(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 $11 {a[1] b[1]}]  
a a* b c
```

SEE ALSO

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. (UPF-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. (UPF-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)
```

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_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 name_list
```

Data Types

```
switch_name      string  
domain_name     string
```

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 name_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)`

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)
```


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
```

102400

SEE ALSO

`cputime(2)`

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)  
saif_map(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:

```
prompt> name_format -isolation_prefix "MY_ISO" -level_shift_prefix "MY_LS"
```

SEE ALSO

```
set_isolation(2)  
set_level_shifter(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_registers

Performs retiming of sequential cells (edge-triggered registers or level-sensitive latches) on a mapped gate-level netlist; determines the placement of sequential cells in a design to achieve a target clock period; and minimizes the number of sequential cells while maintaining that clock period.

SYNTAX

```
int optimize_registers
[-minimum_period_only]
[-no_compile]
[-sync_transform multiclass | decompose | dont_retime]
[-async_transform multiclass | decompose | dont_retime]
[-check_design [-verbose]] [-print_critical_loop]
[-clock clock_name [-edge rise | fall]]
[-latch]
[-justification_effort low | medium | high]
[-only_attributed_designs]
```

Data Types

clock_name string

ARGUMENTS

- minimum_period_only**
Indicates that only the minimum period step of the retiming algorithm (minimum clock period retiming) and not the minimum area (sequential cell count optimization) step is to be executed. By default, both the minimum period and minimum area optimization steps are executed. This option is useful if you want a fast turnaround when trying to optimize the design's timing. The runtime is reduced, but your area results will not be optimal. After you are satisfied with the timing, you can attempt to reduce area by running the retiming command without this option.
- no_compile**
Indicates that the default incremental logic synthesis step, normally performed after computation of the optimal sequential cell locations, is to be omitted. If you specify this option, no design rule fixing will be performed. Generic sequential cells may remain in the design if the **-no_compile** option is specified. Using the **-no_compile** option, you can choose a logic compilation script that is adapted to your design instead of relying on the default used internally by **optimize_registers**. It is important to perform a logic synthesis step after sequential cell retiming to obtain the best possible timing results.
- sync_transform multiclass | decompose | dont_retime**
Specifies which transformation method is used for synchronous sequential cells in the design. An edge-triggered register is synchronous if none of its input pins change the outputs asynchronously. A level-sensitive latch is considered synchronous if none of its input pins can change the outputs during the clock phase where the latch is not transparent. Selecting *multiclass*

transformation specifies that the identifiable synchronous clear, set, and enable functionality is moved with the synchronous sequential cells if they are moved during retiming. sequential cells are classified according to their set, clear, and enable connections. The class of the sequential cells at the fanin or fanout of a combinational cell determines whether retiming across this cell can be performed. Selecting the *decompose* transformation specifies that synchronous sequential in the design are transformed into an instance of a D-flip-flop respectively D-latch and additional combinational logic to create the necessary synchronous functionality. Only the D-flip-flop/D-latch instance can be moved during retiming. Specifying *dont_retime* specifies that synchronous sequential cells will not be moved during retiming. Their mapping might still be changed to a different flip-flop from the technology library. The **dont_touch** attributes and retiming transformation attributes set on individual sequential cells override the value set in this option. To set retiming transform attributes, use the **set_transform_for_retiming** command. The default argument for this option is *multiclass*.

-async_transform multiclass | decompose | dont_retime

Specifies which transformation method is used for asynchronous sequential cells in the design. An edge-triggered register is asynchronous if at least one of its input pins changes the outputs asynchronously. A level-sensitive latch is asynchronous if at least one of its inputs can change the outputs during the clock phase where the latch is not transparent. Selecting the *multiclass* transformation specifies that the identifiable asynchronous clear and set functionality, as well as any synchronous set, clear, and enable functionality, is moved with the asynchronous sequential cells, if they are moved during retiming. sequential cells are classified according to their set, clear, and enable connections. The class of the sequential cells at the fanin or fanout of a combinational cell determines whether retiming can be performed across this cell. Selecting the *decompose* transformation specifies that asynchronous sequential cells in the design are transformed into an instance of a flip-flop respectively latch with asynchronous set and clear inputs, as necessary, and additional combinational logic to create the necessary synchronous functionality. Only the flip-flop/latch instances can be moved during retiming. They will be classified according to their synchronous set, clear, and enable functionality. Selecting the *dont_retime* value specifies that asynchronous sequential cells will not be moved during retiming. Their mapping might still be changed to a different flip-flop from the technology library. The **dont_touch** attributes and retiming transformation attributes set on individual sequential cells override the value set in this option. To set retiming transform attributes, use the **set_transform_for_retiming** command. The default value for this option is *multiclass*.

-check_design

Indicates that additional information about the design is to be displayed before and after retiming. This information includes the number of cells in different categories (for example, hierarchy cells with **dont_touch** attributes or non-movable sequential cells) and more detailed information about the selection of the preferred flip-flop respectively latch. You use this information to help in troubleshooting if retiming does not show the expected results.

-verbose

For use only with the **-check_design** option. Indicates that the explicit names

of the cells are to be displayed along with the number of cells in each category for most categories of the **-check_design** option. The explicit naming of cells can help to locate a problem; however, the lists of output names might be long.

-print_critical_loop

Indicates that the critical loop of the design, as seen during retiming, is to be displayed. The critical loop is defined as the sequence of directly-connected combinational and sequential cells whose total combinational delay divided by the number of registers in the loop has a higher value than any other loop in the design. The critical loop limits the minimum clock period that can be achieved by retiming. Use this option to help in troubleshooting problem areas of the design if the intended clock period cannot be achieved with the given number of sequential cells in the design. If you are pipelining a data path, you might need to add pipeline stages in the HDL code.

-clock *clock_name*

Specifies the name of the clock whose sequential cells are to be retimed. The clock must not be a virtual clock, i.e. it must have a clock port associated with it. The name of the clock is either the name specified in the **create_clock** command or the name of the clock port, if the **create_clock** command had no name specified. The registers of the clock are all sequential cells which are triggered by this clock. The connection from the clock port to the sequential cell can be through clock gating cells, buffer cells and inverter cells. If the **-clock** option is specified and edge-triggered registers are retimed, registers of other clocks are not retimed. If level-sensitive latches are retimed and the **-clock** is specified, the latches driven by this clock as well as those driven by other clocks needed to complete a two-phase clock system are retimed. If edge-triggered registers are retimed, only registers triggered by one specific edge of the clock are retimed. By default the registers triggered by the rising edge are retimed. A different edge can be specified using the **-edge** option.

-edge rise | fall

Specifies whether the registers triggered by the rising or the falling edge of the clock are to be retimed. This option can only be used together with the **-clock** option. When level-sensitive latches are retimed this option does not matter.

-latch

Specifies that level-sensitive latches are to be retimed instead of edge-triggered sequential cells (flip-flops). If this option is used the edge-triggered sequential cells in the design will not be moved. In order to have be able to retime latches they must be driven by a symmetrical two-phase clocks system. Latches that are used to prevent glitches in gated clocks will not be moved, even if the **-latch** option is used. These latches are in the fanin of clock-gating cells.

-justification_effort low | medium | high

Specifies the justification effort level that is to be used during backward justification of registers. You can specify one of low, medium or high as a value for this option. Specifying low would make sure that justification terminates very quickly, but the QoR may be bad. Specifying medium could give good QoR with larger runtime. Specifying high could give the best QoR without any regard for runtime. The default value for this option is medium.

`-only_attributed_designs`

Specifies that instead of the top-level design only instances of those designs in the hierarchy below the current designs that have the "optimize_registers" attribute set are retimed. The "optimize_registers" attribute can be set by using the **set_optimize_registers** command. The instances with attributes will be retimed using the constraints derived from their environment.

DESCRIPTION

The **optimize_registers** command operates by default on the current design. If the "`-only_attributed_designs`" is used it will work only inside those hierarchical cell of the current design that are instances of designs having the `optimize_registers` attribute. The command operates in two phases. During the first phase, it performs retiming by moving the sequential cells in the design to meet a target clock period and minimize the number of sequential cells while maintaining that clock period. The second phase consists of an incremental compile, which adjusts the design to the changed fanout structure. By default, **optimize_registers** uses the clock period of the clocks being retimed, if available. Otherwise, **optimize_registers** returns without moving registers. The final incremental compile targets the clock period defined by **create_clock**.

If the design has multiple clocks which are not virtual clocks and the **-clock** option has not been used, sequential cells for all clocks will be retimed during the first phase of retiming. When retiming edge-triggered registers the retiming will be performed one clock at a time. When retiming level-sensitive latches the retiming will be performed for sets of clocks which together with their latches form a symmetric two phase clock system. Clocks with a larger clock period will be retimed before clocks with a smaller clock period. If two clocks have the same clock period the clock with the larger number of movable sequential cells will be retimed first. When retiming edge-triggered registers for each single clock the registers triggered by the rising edge will be retimed before those triggered by the falling edge. Please be advised that this default order may not yield the best possible results. Also retiming all clocks in phase one means that there will be no incremental optimization of the combinational logic when retiming different clocks. Therefore it is recommended that you determine the best order for retiming clocks yourself and apply it using multiple runs of **optimize_registers** using the `-clock` option.

The **optimize_registers** command has the following requirements:

- 1) A gated clock must be derived from one of the design's clock ports or an other gated clock through a unate clock gating cell (usually an logic AND or logic OR gate). The clock gating control logic may contain latches. The clock network between the base clock port and gated clocks may contain buffer and inverter cells.
- 2) Flip-flops and master-slave elements or cannot be present in the same design. Master and slave clock waveforms cannot overlap.
- 3) All clock pins of all flip-flops in the design must be connected to their base clock or a gated clock derived from the base clock in the following way: The connection from the clock origin (i.e. the clock port or the clock gating cell) to the clock pins can contain buffer and inverter cells with one input and one output pin. All clock pins must be connected either by an even number (including zero) or odd number of inverters to the clock origin. Buffer and inverter cells on the clock

network may be removed during retiming. Therefore any existing clock tree has to be resynthesized.

4) All sequential cells must be single bit, or it must be possible to decompose them into single-bit registers.

5) Only certain FPGA technologies are supported. See the DCFPGA documentation for more details.

6) Designs cannot contain a combinational loop.

7) The timing constraints for the design should be set in the following way: The external clock ports of the design must have a clock constraint created by the `create_clock` command. These will be called the base clocks. All primary inputs of the design should have a non-negative input delay relative to one or more of the base clocks. All primary outputs of the design should have a non-negative output delay relative to one of the base clocks. Negative input and output delays will be tolerated but the quality of the final retiming result may be worse than expected. Point-to-point timing exceptions as created by the `set_false_path`, `set_multicycle_path`, `set_max_delay` and `set_min_delay` commands will be respected but their presence may reduce the quality of results. In the presence of such point-to-point exceptions timing constraint violations may be worsened by retiming. Therefore it is strongly recommended to not apply retiming to designs with these exceptions. Case analysis constraints will also be ignored when moving the sequential cells. The incremental compilation after moving the sequential cells takes all types of constraints into account.

8) Designs cannot contain unmapped synthetic library components.

The movement of sequential cells and the handling of hierarchical subdesigns can be controlled as follows (in addition to the `async_trans` and `sync_trans` options):

1) If a sequential cell has the **`dont_touch`** attribute set, **`optimize_registers`** does not move the sequential cell itself, nor does it move any other sequential cell across that cell.

2) If instances contain the **`dont_touch`** attribute, they are not ungrouped. The **`optimize_registers`** command does not ungroup and does not place any registers inside an instance that has the **`dont_touch`** attribute set. If a hierarchical cell does not contain sequential cells and has the **`dont_touch`** attribute set, sequential cells can move across the cell. If the cell does contain sequential cells and has the **`dont_touch`** attribute set, sequential cells cannot move across the cell .

3) The **`optimize_registers`** command can move sequential cells into a level of hierarchy. In this case, a clock pin is inserted into the interface of the instance if there was no clock pin previously. The new clock pin is named after the clock pin of the enclosing hierarchical instance.

The **`optimize_registers`** command supports handling of black box cells (that is, cells for which no timing is specified; for example, placeholders for RAMs). The **`optimize_registers`** command models a black box cell as if the cell were external to the current design, without actually changing the interface of the design itself.

The **`optimize_registers`** command includes a delay modeling capability. For the sake of predictability, the algorithm selects from the target library a flip-flop or latch

that has a small setup time, good drive at the outputs, and low input loading as compared with other flip-flops or latches in the library. This flip-flop respectively latch is designated as the preferred flip-flop or preferred latch. The preferred flop/latch helps to provide tighter bounds on the performance variation after the **optimize_registers** sequence. To select a particular preferred flip-flop, use the **set_register_type** command. To exclude certain flip-flops, use the **set_dont_use** command.

The retiming process produces some delay report information. For details, see the *Design Compiler Reference Manual: Register Retiming*. After retiming, implementation selection is no longer performed on synthetic library components in the design by subsequent executions of the **compile** command.

optimize_registers command is also supported in multicorn-multimode (MCMM) technology which is available with Design Compiler Graphical.

EXAMPLES

The following example shows **optimize_registers** being applied to a netlist.

```
prompt> read pre_retiming.db
```

```
/* * Delete dont_touch'es used for * hierarchical compile strategy. */
```

```
prompt> optimize_registers
```

SEE ALSO

```
current_design(2)
pipeline_design(2)
set_dont_touch(2)
set_dont_use(2)
set_transform_for_retiming(2)
set_register_type(2)
set_optimize_registers(2)
optimize_reg_retime_clock_gating_latches(3)
optimize_reg_max_time_borrow(3)
```

pipeline_design

Pipelines combinational designs by adding registers at the outputs and by retiming the circuit.

SYNTAX

```
integer pipeline_design
[-stages number_of_stages]
[-stall_ports port_list]
[-stall_polarity high | low]
[-sync_reset reset_port
 | -async_reset reset_port]
[-reset_polarity high | low]
[-clock_port_name clock_port]
[-check_design [-verbose]]
[-print_critical_loop]
[-minimum_period_only]
[-register_outputs]
[-exact_map]
[-no_compile]
```

Data Types

<i>number_of_stages</i>	integer
<i>port_list</i>	list
<i>reset_port</i>	string
<i>clock_port</i>	string

ARGUMENTS

- stages** *number_of_stages*
Specifies the number of pipeline stages the design is to have after execution of the command. The number of stages is one more than the number of registers encountered on any path from any data input port to any data output port of the design. The minimum possible value and the default value is 2.
- stall_ports** *port_list*
Specifies one or more 1-bit wide input ports of the design to be used as stall ports. The ports must exist in the design before the **pipeline_design** command is executed. For information about how the connection is made depending on the number of stall ports, see Item 7 in the DESCRIPTION section in this man page. If no stall ports are specified, loading of the registers is always enabled. The list of port names must be enclosed in curly braces.
- stall_polarity** high | low
Specifies the polarity of the stall ports as active high (the default) or active low. When the input value at a stall port is active, the corresponding registers keep their state value at the rising edge of the clock.
- sync_reset** *reset_port*
Specifies an existing 1-bit-wide *reset_port* to be used as a synchronous reset for the design. By default, there is no reset. If this option is specified,

registers are connected to the *reset_port* after retiming so that they are reset synchronously by an active reset signal. This option and **-async_reset** are mutually exclusive.

-async_reset *reset_port*

Specifies an existing 1-bit-wide *reset_port* to be used as an asynchronous reset for the design. By default, there is no reset. If this option is specified, registers are connected to the *reset_port* after retiming so that they are reset asynchronously by an active reset signal. This option and **-sync_reset** are mutually exclusive.

-reset_polarity *high* | *low*

Specifies the polarity of the active reset as active high or active low (the default). This option is used with the **-async_reset** or **-sync_reset** options.

-clock_port_name *clock_port*

Specifies the name of the clock port; the default is **clock**. The clock port is a 1-bit wide input port connected to the clock pins of all registers in the design, and must already exist before execution of the **pipeline_design** command.

-check_design

Indicates that additional information about the design is to be displayed before and after retiming. This information includes the number of cells in different categories, such as *dont_touched* hierarchy cells or non-movable sequential cells, and more detailed information about the selection of the preferred flip-flop. Use this information to help with troubleshooting, if retiming does not show the expected results.

-verbose

Indicates that the explicit names of the cells are to be displayed along with the number of cells in each category for most categories of the **-check_design** option. The explicit naming of cells can help to locate a problem; however, the lists of output names might be long. This option is used only with the **-check_design** option.

-print_critical_loop

Displays the critical loop of the design, as seen during retiming. The critical loop is defined as the sequence of directly-connected combinational and sequential cells whose total combinational delay divided by the available number of registers has a higher value than any other loop through the design. The critical loop limits the minimum clock period that can be achieved by retiming. Use this option when troubleshooting problematic areas of the design if the intended clock period cannot be achieved with the given number of pipeline stages.

-minimum_period_only

Indicates that only the minimum period step of the retiming algorithm (minimum clock period retiming) is to be executed, and not the minimum area (register count optimization) step. By default, both the minimum period and minimum area optimization steps are executed. This option is useful if you want to have a fast turnaround when trying to optimize the design's timing. The runtime is reduced, but your area results will not be optimal. Once you are satisfied with the timing, you can attempt to reduce area by running the retiming command without this option.

-register_outputs

Registers the primary outputs of the pipelined design. The total number of registers encountered on a path from a data input port to a data output port is still *number_of_stages - 1*; that is, no extra registers are added. Because each primary output is always connected directly to a register when this option is used, the retiming algorithm is constrained. The delay of combinatorial logic between pipeline stages can be longer than the case when this option is not used. This means that the minimum clock period that can be achieved for a given number of pipelining stages using the **-register_outputs** option is usually longer than that which can be achieved without the option.

-exact_map

Specifies that during incremental compile, exact mapping of sequential cells is carried out. A single flip-flop cell, as opposed to a flip-flop with additional logic implementing enable, reset and so on, is used. This option can be used together with the **-register_outputs** option to prevent combinatorial logic such as inverters from being placed between a register of the last stage and the primary output driven by this register. The two options usually ensure that a last stage register is directly connected to an output port. There may be an area overhead when this option is used.

-no_compile

Indicates that the default incremental logic synthesis step, normally performed after computation of the optimal register placement, is to be omitted. Using this option, no design rule fixing will be performed. Generic sequential cells may remain in the design, if the **-no_compile** option is selected. Using this option, you can choose a logic compilation script that is adapted to your design instead of relying on the default used internally by **optimize_registers**. It is important to perform this logic synthesis to obtain the best possible timing results.

DESCRIPTION

This command pipelines combinational designs by adding registers at the outputs and by retiming the circuit. It works on the current design. Register initialization and loading are controlled using reset and stall ports of the design, which are connected to the registers to perform this function. During retiming, placement of the registers in the design is determined to achieve a target clock period and to minimize the number of registers required. This command is applied to a mapped gate-level netlist, which must not contain sequential elements or feedback loops.

The **pipeline_design** command is executed as described in the following steps:

- 1) All non-constant and connected output ports of the design are identified, and as many registers as specified by the number of stages reduced by one are inserted at each of these output ports.
- 2) A preferred flip-flop is selected from the technology library for the estimation of setup time and clock-to-Q-delay. After retiming, preference is given to load-enable flip-flops. You can control the selection of the preferred flip-flop by using the **set_register_type** command with the **-exact** and **-flip_flop** options.
- 3) The target clock period is determined by the clock port created using the **create_clock** command. A correction with the setup time and clock-to-Q delay of the preferred flip-flop is performed.
- 4) If the desired clock period value can be achieved, a minimum area retiming is

performed for this clock period. Otherwise, a minimum area retiming for the minimum possible clock period is performed.

5) The critical loop is displayed if requested.

6) The registers are connected to the clock port and to a reset port, if it is specified.

7) The registers are connected to the stall ports. Three situations are possible:

a) There is only one stall port. All registers are controlled by this stall port.

The registers are loaded if the stall pin is inactive. The registers keep their state if the stall pin is active.

b) There is more than one stall port. In this case, the number of stall ports must be one less than the number of stages. The flip-flops are indexed by their distance from the input ports. Those flip-flops that can be reached from the inputs directly get the index one. Those that can be reached by just crossing one other register get the index two, and so on. Flip-flops with the index *i* are controlled by the *i*-th stall port given in the command line.

c) There is no stall port. The loading of the internal registers is always enabled.

8) Sequential mapping is performed to select the best possible flip-flop cells for the stall and reset conditions.

9) If **-no_compile** is specified, no postprocessing of the netlist is performed at all. Generic or unmapped cells may remain in the design. If **-no_compile** is not specified, an incremental compile with medium mapping effort is performed on the retimed design.

The **pipeline_design** command has the following requirements:

- The design on which the command is executed must be mapped and purely combinational without feedback loops.

- The inserted registers are edge-triggered flip-flops with one clock signal. Latches and master-slave flip-flops are not supported.

- All clock, reset, and stall ports specified in the command line must be 1-bit wide input ports that already exist in the design.

- The active reset always clears (sets to zero) all registers in the pipelined design. Note that the initial state that is achieved by this is in general different from the initial state of a design that would be produced by the following steps: Creating an RTL description of the design with all registers located at the primary inputs or primary outputs and being cleared by the active reset, followed by retiming the design with **optimize_registers**.

Thus, outputs of the pipelined design should be used only as many clock cycles after the reset became inactive as given by the number of stages minus one. During this time, the stall signals must be inactive. That is, assuming that the pipeline has *N* stages, you have to wait *N* clock cycles after active reset until the following match: output values of the netlist created by **pipeline_design** and those of the netlist created by instantiating registers in the RTL and using **optimize_registers**.

- Only certain FPGA technologies are supported.

- Timing delay for the stall and reset distribution networks is not taken into account during retiming.

- The technology library must contain a rising-edge-triggered D flip-flop with no reset, no load enable, and no test input.

- The only timing constraints that are supported are non-negative input and output delays on the primary input and output ports of the design relative to the design's clock. Input and output delays relative to virtual clocks, case analysis constraints, and point-to-point timing exceptions as set by **set_false_path**, **set_multicycle_path**, **set_max_delay** and **set_min_delay** are tolerated but do not influence the optimization of the register locations and should be avoided.
- Designs cannot contain tri-state cells or unmapped synthetic library components.

The placement of registers and the handling of hierarchical subdesigns can be controlled as follows. Instances are not ungrouped if they contain the **dont_touch** attribute. The **pipeline_design** command does not ungroup and does not place registers inside an instance that has the **dont_touch** attribute. If necessary, registers can move across blocks containing the **dont_touch** attribute.

The **pipeline_design** command supports handling of black box cells (that is, cells for which no timing is specified; for example, placeholders for RAMs). However, the results might not be as intended. The **pipeline_design** command models a black box cell as if the cell were external to the current design, without changing the interface of the design itself. Therefore, the use of black boxes is not recommended.

The retiming process produces some delay report information. For details, see the *Design Compiler Reference Manual: Register Retiming*.

After retiming, implementation selection is no longer performed on synthetic library components in the design by subsequent executions of the **compile** command.

EXAMPLES

In the following example, assume that a multiplier design must be pipelined with three stages and two distinct stall ports with active polarity low. Asynchronous reset with polarity low is specified. The clock period should be five nanoseconds.

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;

entity multiplier is
    port(a, b      : in SIGNED(4 downto 0);
         clock     : in STD_LOGIC;
         stall_1   : in STD_LOGIC;
         stall_2   : in STD_LOGIC;
         reset     : in STD_LOGIC;
         z         : out SIGNED(9 downto 0));
end multiplier;

architecture arc of multiplier is
begin
    z <= a * b;
end arc;
```

The following command sequence achieves the requirements:

```
prompt> analyze -f vhd1 multiplier.vhd1
prompt> elaborate multiplier
prompt> compile
prompt> create_clock clk -period 5
prompt> pipeline_design -stall_ports {stall_1, stall_2} \
-stages 3 -stall_polarity low -async_reset reset
```

The following command sequence causes all registers to be controlled by only stall port stall_1:

```
prompt> analyze -f vhd1 multiplier.vhd1
prompt> elaborate multiplier
prompt> compile
prompt> create_clock clk -period 5
prompt> pipeline_design -stall_ports stall_1 \
-stages 3 -stall_polarity low -async_reset reset
```

SEE ALSO

```
current_design(2)
optimize_registers(2)
set_dont_touch(2)
set_dont_use(2)
set_register_type(2)
```


preview_dft

Previews, but does not implement, the test points, scan chains, and on-chip clocking control logic to be added to the current design.

SYNTAX

```
status preview_dft
[-show object_types]
[-test_points all]
[-test_mode mode_name]
[-test_wrappers all]
[-bsd cells | data_registers | instructions | tap | all]
[-script]
[-verbose]
```

Data Types

<i>object_types</i>	list
<i>mode_name</i>	string

ARGUMENTS

`-show object_types`

Reports information for the objects specified as values, in addition to the summary report the command produces by default. You must specify one or more of the following values for the *object_types* argument:

- The **bidirectionals** argument displays bidirectional port conditioning in scan shift. Reports the specified and resolved conditioning for each bidirectional port.

- The **cells** argument displays scan cells. Scan segments are distinguished by using keys that default to **s**. Scan link locations are marked by keys placed on the cells that precede them. Wire keys default to **w**; scan-out lockup keys default to **l**. You can change key defaults by using the **test_scan_segment_key**, **test_scan_link_so_lockup_key**, and **test_scan_link_wire_key** variables.

- The **scan** argument displays hierarchical **set_scan_element** command attribute values. Although **set_scan_element** commands can be applied to hierarchical cells, leaf cells, library cells, and designs, the **preview_dft** command reports scan attribute values only for hierarchical cells and leaf cells. These reflect the impact of every **set_scan_element** command you apply to the design. Scan cells with true scan attributes are distinguished by using keys that default to **t**. You can change the default by using the **test_scan_true_key** environment variable.

- The **scan_clocks** argument displays scan clock domains. For multiplexed flip-flop scan styles, it displays the first cell or segment in the chain. An ordered triplet shows the scan clock name, trigger time, and whether the clock edge is rising or falling.

The **preview_dft** command does not distinguish capture time and launch time because these are the same for edge-triggered elements. Additional entries

are generated when either the scan clock or trigger time changes. It displays exit and entry points. Lines containing three periods (.) show that there are intervening scan cells. Only entry points have ordered triplets, which are aligned for readability. If internal clocks have been created in the design by using the **set_scan_configuration** command with the **-internal_clocks** option or the **create_test_clock** command with the **-internal_clocks** option, the **preview_dft** command displays internal clock pin names instead of the external clock names. The triplet is replaced by "(No clock)" if the cell or segment does not have scan clock information.

For other scan styles, using the **preview_dft** command with the **-show_scan_clocks** option prints the scan clocks for every scan chain.

- The **scan_signals** argument displays scan signals, including hookup pins and sense values between them. If scan ports are not shared with mission mode ports, the **preview_dft** command generates interim port names by using the strings specified following environment variables, as appropriate, with %s characters deleted. The following names can change when the ports are created to avoid naming conflicts:

```
test_clock_port_naming_style
test_scan_clock_a_port_naming_style
test_scan_clock_b_port_naming_style
test_scan_clock_port_naming_style
```

- The **segments** argument displays scan segments. It also displays segment scan inputs, scan outputs, scan enables, and scan clocks.
- The **tristates** argument displays the disabling in scan shift for all tristate nets in a design. It reports the specified and resolved disabling for each tristate net.
- The **voltages** argument displays the voltage value for a scan cell or scan segment. It displays the operating voltages of the cell or segment in a scan chain. The voltage is displayed only for the first cell or segment and any other cell or segment at voltage boundaries.
- The **power_domains** argument displays the power domain ID for a scan cell or scan segment. It displays the power domain ID of the cell or segment in a scan chain. The power domain ID is displayed only for the first cell or segment, and any other cell or segment at power domain boundaries.
- The **scan_summary** argument displays scan chain information in short format.
- The **all** argument displays all information about a scan configuration. The **-show** and **-script** options are mutually exclusive; do not use them together. If you attempt to use them together, the command generates a warning message, and the **-show** option is ignored.

-test_points all

Reports all test points information, in addition to the summary report the **preview_dft** command produces by default. The information displayed includes names assigned to the test points, locations of violations being fixed, names of test mode tags, logic states that enable the test mode, and names of data sources or sinks for the test points.

The **-test_points** and **-script** options are mutually exclusive; do not use them

together. If you attempt to use them together, the command generates a warning message, and the **-test_points** option is ignored. By default, this option is off.

-test_mode *mode_name*

Previews DFT logic and the scan architecture for a single given mode when operating in multimode. This reduces the length of the **preview_dft** report. By default, **preview_dft** shows scan architecture in all the modes.

-test_wrappers *all*

Reports all information about core wrappers, in addition to the summary report the command produces by default. This option displays the number of interface ports, ports wrapped, type of wrapper cells inserted, and ports excluded from wrapping. You cannot use this option with the **-script** option. By default, this option is off.

-bsd *cells | data_registers | instructions | tap | all*

Specifies that **preview_dft** reports detailed information about one or more sections of the boundary-scan architecture. The valid values are as follows:
cells shows the boundary-scan register cells.
data_registers shows the name and length of each boundary-scan test data register.
instructions shows the boundary-scan instructions and their optimization codes.
tap shows the TAP ports.
all shows all details of the boundary-scan architecture.

-script

Produces a shell script, written to standard output, that specifies the required scan configuration. You cannot use this option with any of the following options: **-show**, **-test_points** and **-test_wrappers**. If you attempt to use **-script** with these options, the command generates a warning message, and ignores the **-show**, **-test_points**, and **-test_wrappers** options. By default, this option is off.

-verbose

Generates multiple warning messages for a set of similar cells or pins with the same design rule violation. By default, a warning message is generated for only the first cell or pin found for a rule violation.

DESCRIPTION

This command previews, but does not implement, the test points, scan chains, and clock controllers to be added to the current design. The preview is based on the current set of scan, test point, and clock controller specifications. The command generates a report without actually performing any synthesis tasks. The report shows the effects of the current specifications, which you can modify, and then rerun the command as many times as needed until you are satisfied with the previewed results.

This command performs the same scan chain design functions as the **preview_scan** command and generates information on the test points specified for insertion in the design.

Process

The **preview_dft** command first generates information on the scan architecture that will be implemented. In the case of a DFT MAX insertion, **preview_dft** provides information about the compressor being created, and for basic scan, the specific chain information for the design.

Next, the command generates and displays a scan chain design that satisfies scan specifications on the current design. This design is exactly the scan chain design that is presented to the **insert_dft** command for synthesis, so you can preview your test point and scan chain designs without synthesizing the design; you can change your specifications to explore the design space, as necessary.

Executing the command without options generates a summary report that includes the number of test points, test_modes, test point enables, sources and sinks, number of scan chains, test methodology, and scan style. Applicable clock domain constraints, scan enable (or scan enable inverted), and hookup pins are included if the scan style is **multiplexed_flip_flop**. For each scan chain, the command displays the chain name, scan-in port, scan-out port, and length.

If scan ports are not shared with mission mode ports and new ports are created, the command generates interim port names. This is performed by appending an internal integer chain index to the strings specified in the following environment variables, as appropriate, with %s characters deleted. These names can change when the ports are created to avoid cell naming conflicts.

```
test_scan_in_port_naming_style
test_scan_out_port_naming_style
test_scan_enable_port_naming_style
test_scan_enable_inverted_port_naming_style
```

EXAMPLES

The following example displays scan clocks.

```
prompt> preview_dft -show scan_clocks
```

The following example displays scan cells.

```
prompt> preview_dft -show cells
```

The following example shows the **preview_dft** command with the **-show** option set to **all**, reporting scan chain design information. The cell named *ff1* is not added to the scan chain because it is running a **set_scan_element** command set to **false**.

```
prompt> preview_dft -show all
```

```
    Loading design 'WC66'
    Checking test design rules
No test points.
```

```
preview_dft
632
```

Architecting Scan Chains

Warning: Cannot add 'ff1' to chain 'chain0'.
The element is not being scanned (TEST-376).

Preview scan report

Design : hsc3

Version: 1999.10

Date : Wed Oct 22 17:14:24 2007

Number of chains: 1

Test methodology: full_scan

Scan style: multiplexed_flip_flop

Clock domain: mix_clocks

- (l) shows cell scan-out drives a lockup latch
- (s) shows cell is a scan segment
- (t) shows cell has a true scan attribute
- (w) shows cell scan-out drives a wire

Scan chain 'chain0' (test_si --> test_so) contains 10 cells:

a/ff1	(clk1, 45.0, rising)
a/ff2 (l)	
b/ff1	(clk2, 45.0, rising)
b/ff2 (w)	
b/s/ff1	
b/s/ff2	
c/ff1 (w)	(clk1, 45.0, rising)
c/ff2 (l)	
ff2 (t)	(clk2, 45.0, rising)
ff3	

No scan signals

No segments

1 cell has scan true:

ff2

1 cell has scan false:

ff1

1

The following example shows the results when using the **preview_dft** command with the **-script** option run on the same design.

```
prompt> preview_dft -script
```

```
/**/
```

```
/*****
```

```
Created by preview_dft() on Wed Jan 24 16:35:00 2006
```

```
*****/
```

```
set_scan_path chain0 -dedicated_scan_out false {\n  a/ff1 \n  a/ff2 \n  b/ff1 \n  b/ff2 \n  b/s/ff1 \n  b/s/ff2 \n  c/ff1 \n  c/ff2 \n  ff2 \n  ff3 \n}
```

```
1
```

The following example shows the **preview_dft** command reporting test point and scan chain design information. The report also shows on-chip clocking controllers that will be added.

```
prompt> preview_dft -test_point all
```

```
Information: Using test design rule information from previous dft_drc run.
```

```
Running Autofix
```

```
Architecting Test Points
```

```
*****
```

```
PLL clock information preview
```

```
*****
```

```
*****
```

```
Cell name: my_pll_cntrl
```

```
Design name: snps_clk_mux
```

```
*****
```

```
Number of bits per clock: 2
```

```
Clock chain length:      2
```

```
Clock chain count:      1
```

```
Fast clock pins:
```

```
my_pll/pll_clk
```

```
Slow clock ports/pins:
```

```
CLK
```

```
Checking compatibility of PLL clock controller 'snps_clk_mux'
```

```
Architecting Scan Chains
```

```
preview_dft
```

```
634
```

```

*****
Preview_dft report
For      : 'Insert_dft' command
Design   : TOP
Version:  A-2007.12
Date     : Thu Dec 28 08:10:07 2007
*****

```

```

Number of chains: 3
Scan methodology: full_scan
Scan style: multiplexed_flip_flop
Clock domain: no_mix

```

Scan chain '1' (test_si1 --> test_so1) contains 1 cell

Scan chain '2' (test_si2 --> Z1) contains 1 cell

Scan chain '3' (test_si3 --> test_so3) contains 2 cells

```

***** Test Point Plan Report *****
Total number of test points   : 2
Number of Autofix test points: 2
Number of Wrapper test points: 0
Number of test modes          : 1
Number of test point enables  : 0
Number of data sources        : 2
Number of data sinks          : 0
*****

```

TEST POINTS

CLIENT	NAME	TYPE	LOCATIONS	TEST MODES	TEST POINT ENABLE	DATA SOURCE/ SINK
Autofix CP	autofix_tp handler	F-01 handler_3	ff1/ CLK			
Autofix CP	autofix_tp_1 handler	F-01 handler_3	ff2/ handler_2			

TEST MODES

TAG	NEW/ EXISTING	TYPE	NAME	CLOCK
handler	New	Port	test_mode	none

TEST POINT ENABLES

TAG	NEW/ EXISTING	TYPE	NAME	CLOCK
handler_3	unknown	Logic 1	unknown	none

DATA SOURCES

TAG	NEW/ EXISTING	TYPE	NAME	CLOCK
CLK	Existing	Port	CLK	none
handler_2	New	Port	data_source	none

The following example displays a scan summary.

```
prompt> preview_dft -show scan_summary
```

```
*****
```

```
Current mode: Internal_scan
```

```
*****
```

```
Number of chains: 1
```

```
Scan methodology: full_scan
```

```
Scan style: multiplexed_flip_flop
```

```
Clock domain: no_mix
```

Chain	Scan Ports (si --> so)	# of Cells	Inst/Chain	Clock (port, time, edge)
S 1	test_sil --> test_sol	1	u1	(clk, 45.0, rising)

SEE ALSO

```
current_design(2)
insert_dft(2)
set_dft_configuration(2)
set_dft_signal(2)
set_dont_touch(2)
set_scan_configuration(2)
set_scan_element(2)
set_scan_link(2)
set_scan_path(2)
write(2)
target_library(3)
test_default_scan_style(3)
test_scan_link_so_lockup_key(3)
test_scan_link_wire_key(3)
test_scan_segment_key(3)
```

preview_dft

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.

```
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)`

print_variable_group

Lists the variables defined in a specified variable group, along with their current values.

SYNTAX

```
status print_variable_group  
group
```

Data Types

group string

ARGUMENTS

group Specifies the name of the group of variables for which you want to see the definitions and current values.

DESCRIPTION

The **print_variable_group** command lists the variables and related values defined in *group*. The current values for *group* are as follows:

all	All variables
acs	Variables that affect Automated Chip Synthesis
bsd	Variables that affect the insert_dft , optimize_bsd , check_bsd , and write_bsd1 commands
compile	compile command variables
dpcm	Variables that affect DPCM
hdl	Variables that affect reading, writing, and optimizing HDL
insert_dft	insert_dft command variables
io	read_file , read_lib and write command variables
links_to_layout	links_to_layout variables
multibit	Variables that affect compile multibit mapping
power	Variables that affect Power Compiler commands
preview_scan	Variables that affect the preview_scan command
suffix	Variables that define file suffixes
synlib	cache_ls command variables
system	Global system variables
test	DFT Compiler variables
testmanager	Variables that affect fault list handling
timing	Variables that affect timing
vhdlio	Variables that affect writing VHDL files and libraries
write_test	write_test command variables

The available variable groups are configurable. Change variable groups or create new ones with the **group_variable** command.

EXAMPLES

The following example displays the variables in the `acs` variables group, along with their current values:

```
prompt> print_variable_group timing
case_analysis_log_file      = ""
case_analysis_sequential_propagate = "false"
case_analysis_with_logic_constants = "false"
create_clock_no_input_delay = "false"
disable_auto_time_borrow   = "false"
disable_case_analysis      = "false"
disable_conditional_mode_analysis = "false"
disable_library_transition_degradation = "false"
1
```

SEE ALSO

```
dc_shell(1)
group_variable(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)
```

propagate_annotated_delay_up

This command is obsolete and has been replaced by the **propagate_ilm** command. For more information, see the man page for the **propagate_ilm** command.

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_clock_gating_style(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_placement

Propagates placement information for all leaf cells in the specified list of hierarchical cells from their respective reference designs to the current design.

SYNTAX

```
int propagate_placement  
[-verbose]  
cell_list
```

ARGUMENTS

-verbose
Prints out additional information messages regarding cell location adjustment.

DESCRIPTION

This command propagates the location and orientation for all leaf cells in the specified list of hierarchical cells from their respective reference designs to the current design.

The location and orientation of each leaf cell in the hierarchical cell is adjusted according to the top-level floorplan. The hierarchical cell instance can be oriented with any of these orientations: N, S, FN, and FS. The orientation of the reference ILM block is assumed to be N.

For example, if the lower-left co-ordinates of the die area for the reference design of the hierarchical cell is (x1, y1), and the hierarchical cell is placed at (x2, y2) with orientation N in the top-level floorplan, then after this command is executed, then a leaf cell with original location (x, y) will be placed at (x2 + x - x1, y2 + y - y1).

Note that if the die area is not specified for the reference design of the hierarchical cell, then (x1, y1) is the lower-left co-ordinates of its core placement area.

The current design before executing this command must be the top design. Execute this command after linking the current design. After this command is executed, you can verify the location of cells inside the specified list of hierarchical cells using the command **report_cell -physical**.

To propagate placement of leaf cells inside Interface Logic Model (ILM), use the `propagate_ilm` command.

EXAMPLES

The following example propagates placement information for hierarchical cells foo1, foo2 to the top design. The location of the leaf cells inside the hierarchical cells is then reported using the command **report_cell -physical**.

```
prompt> read_db top.db
prompt> read_db foo1_sub_design.db foo2_sub_design.db
prompt> current_design top
prompt> link
prompt> propagate_placement -verbose {foo1 foo2}
prompt> report_cell -physical foo1/* foo2/*
```

SEE ALSO

```
report_cell(2)
set_placement_area(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)
```


push_down_model

Used to create a new hierarchy around the current design for use with wrapping models flow in SoCTest.

SYNTAX

```
int push_down_model  
new_design_name
```

Data Types

```
new_design_name      string
```

ARGUMENTS

```
new_design_name  
    Specifies the name of the new design to be created.
```

DESCRIPTION

The **push_down_model** command creates a new level of hierarchy around the current design. It creates a new design with the specified design name, instantiates the current design as an instance in the new design, creates ports in the new design for all corresponding pins of the instantiated cell, and connects all pins of the instantiated cell to the corresponding ports of the new design. Current design is set to the new design created by this command.

EXAMPLES

The following example creates a new design by name "sub1_pushed" around the current design "sub1".

```
prompt> current_design sub1  
{"sub1"}  
prompt> push_down_model sub1_pushed  
Current design is 'sub1_pushed'.  
prompt>
```

SEE ALSO

```
insert_dft(2)  
preview_dft(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_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

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_bsd1

Reads the boundary-scan description language (BSD1) file for a boundary-scan design.

SYNTAX

```
status read_bsd1  
file_name
```

Data Types

```
file_name      string
```

ARGUMENTS

```
file_name  
    Specifies the name of the input BSD1 file.
```

DESCRIPTION

Reads the BSD1 file for a boundary-scan design and generates an internal data structure for use with BSD pattern generation by `create_bsd_patterns` command.

Before reading a BSD1 file, you should read in a netlist for the TOP level design and set the current design to the TOP level design. This netlist can be a black box description of the TOP level design; the module can contain only the IOs of the design.

EXAMPLES

The following example reads the BSD1 input file `design1.bsd1` for the boundary-scan design, `design1`, and generates patterns into various formats for the Boundary Scan logic of the design.

```
prompt> read_file -f verilog design1.v  
prompt> current_design design1  
prompt> link  
prompt> read_bsd1 design1.bsd1  
prompt> create_bsd_patterns  
prompt> write_test -f stil_testbench -o design_stil_tb  
prompt> write_test -f verilog -o design_verilog_tb  
prompt> write_test -f wgl_serial -o design_wgl_tb
```

SEE ALSO

```
create_bsd_patterns(2)  
current_design(2)  
read_file(2)  
write_test(2)
```

read_db

Reads in one or more design or library files in Synopsys database (.db) format.

SYNTAX

```
string read_db  
file_names
```

Data Types

```
file_names      list
```

ARGUMENTS

```
file_names  
    Specifies names of one or more files to be read.
```

DESCRIPTION

Reads design information from Synopsys database (db) files into **dc_shell-t**.

This command is derived from **read_file -format db**. For more information, see the man page for the **read_file** command.

SEE ALSO

```
read_file(2)
```

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_file

Reads designs or libraries into memory, or reads libraries into the shell.

SYNTAX

```
list read_file  
file_list  
[-define macro_names]  
[-format format_name]  
[-ilm]  
[-library library_name]  
[-rtl]  
[-single_file single_file_name]  
[-work library_name]  
[-scenarios scenario_list]  
[-active_scenarios active_scenario_list]  
[-names_file file_list]
```

Data Types

<i>file_list</i>	list
<i>macro_names</i>	list
<i>format_name</i>	string
<i>library_name</i>	string
<i>single_file_name</i>	string
<i>scenario_list</i>	list
<i>active_scenario_list</i>	list

SYNTAX for lc_shell

```
list read_file  
  
file_list
```

Data Types for lc_shell

<i>file_list</i>	list

ARGUMENTS

file_list
Specifies a list of files to read. When specifying more than one *input_file*, enclose the list of names in braces ({}).

-define *macro_names*
Specifies a list of top-level macros for verilog and systemverilog only.

-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
verilog	IEEE Standard Verilog
sverilog	IEEE Standard SystemVerilog
vhdl	IEEE Standard VHDL
equation	Synopsys equation format
pla	Berkeley (Espresso) PLA format
st	Synopsys State Table format

-ilm

Reads from the Milkyway ILM (interface logic model) view.

-library *library_name*

Remaps the work library to *library_name* For VHDL only. By default, the **analyze** command stores all output in the work library. To store design elements in libraries other than library specified by using the **-work** option, use the **-library** option.

The **read_file** command has three object types: **design** (to change design names and references to that design), **reference** or **component** (to change reference names), and **pin** (to change reference pin names). If you use the **reference** or **component** arguments, the design names field is ignored and the reference name is changed in the incoming EDIF files (input files).

-rtl

Specifies that an RTL design is being read. This option can be used only in conjunction with **-f verilog** or **-f vhdl**. When used, the **read_file** command invokes the HDL Compiler directly without attempting to determine if the specified files are gate-level netlists. For details, see the **hdlin_auto_netlist_reader** variable man page.

-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.

-work *library_name*

Remaps the work library in the same way as the **-library** option. It is an alias for **-library** for VHDL only.

-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 reads a file containing multiple designs, so each design is assigned its own internal design filename. The asterisk (*) indicates the current design. For more information, see the **list_designs** man page.

```
prompt> read_file -format verilog foo.v
Loading file '/usr/bill/dc/foo.v'
A, B, C
```

```
prompt> list_designs
```

	Design	File	Path
	-----	----	----
*	A	A.ddc	/usr/bill/dc
	B	B.ddc	/usr/bill/dc
	C	C.ddc	/usr/bill/dc

The following example shows that when you use the **-single_file** option, all designs are assigned the same filename:

```
prompt> read_file -format verilog -single_file mylib.ddc foo.v
Loading file '/usr/bill/dc/foo.v'
A, B, C
```

```
prompt> list_designs
```

	Design	File	Path
	-----	----	----

*	A	mylib.ddc	/usr/bill/dc
	B	mylib.ddc	/usr/bill/dc
	C	mylib.ddc	/usr/bill/dc

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

```
analyze(2)
all_scenarios(2)
all_active_scenarios(2)
change_names(2)
create_scenario(2)
elaborate(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_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_parasitics

Reads net parasitics information from an SPEF, DSPF, or RSPF file, and uses it to annotate the current design.

SYNTAX

```
int read_parasitics
[-syntax_only]
[-elmore | -arnoldi]
[-increment] [-pin_cap_included]
[-net_cap_only] [-complete_with zero | wlm]
[-path path_name]
[-strip_path path_name]
[-quiet | -verbose]
[-dont_write_to_db]
[file_list]
```

Data Types

<i>path_name</i>	string
<i>file_list</i>	list

ARGUMENTS

- syntax_only**
Specifies not to perform back-annotation. You can use this option to check whether your parasitics file is valid.
- elmore**
Specifies to create an RC tree and back-annotate delay-estimates. The delays are computed out of the parasitics information based on the Elmore delay model. If neither '-elmore' nor '-arnoldi' is specified, the RC tree is created without estimating and back annotating delays.
- arnoldi**
Specified to create an RC tree and back-annotate delay-estimates. The delays are computed out of the parasitics information via the Arnoldi delay calculator. If neither '-elmore' nor '-arnoldi' is specified, the RC tree is
- increment**
Specifies not to discard previously annotated parasitics on the nets listed in the parasitics file. This option is used for annotating hierarchical parasitics files. By default, the RC annotation specified in the parasitics file overwrites the previous parasitics annotations of the nets listed in the parasitics file.
- pin_cap_included**
Specifies to the command that the RC network includes the pin capacitances. By default, the RC network does not include them.
- net_cap_only**
Specifies that only capacitances are going to be back-annotated. Pin-to-pin

delay back-annotation is skipped. By default, both the capacitances and the pin-to-pin delays are back-annotated.

`-complete_with zero | wlm`

Specifies to automatically complete the RC tree after parsing. This option should be used if a parasitics file contains incomplete parasitics information; for example, if only a segment of a wire is annotated. During the RC completion phase, missing wire segments are automatically added. If `zero` is specified, all newly-added segments and nodes are considered to have 0 resistance and 0 capacitance. If `wlm` is specified, the capacitance and resistance for a new segment is determined by the current wire load model.

`-path path_name`

Specifies the path from the current design to the specified subdesign for which the parasitics file has been created.

`-strip_path path_name`

Specifies that all objects in the parasitics file have a prefix path that needs to be stripped off. This is usually a result of generating a parasitics file for a subdesign, and using this subdesign as the current design.

`-quiet`

Specifies not to perform **report_annotated_delay** when the parasitics file has been read. By default, after reading the parasitics file, the **report_annotated_delay** command is executed. This command reports the back-annotated delays of all nets.

`-verbose`

Prints annotation report after parsing.

`-dont_write_to_db`

Suppresses writing back to the Synopsys database (db) to save runtime.

`file_list`

Specifies a list of files from which to read parasitics information. Supported formats are Standard Parasitic Exchange Format (SPEF), Detailed Standard Parasitics Format (DSPF), and Reduced Standard Parasitics Format (RSPF).

DESCRIPTION

The **read_parasitics** command reads parasitics information for nets of the current design from a disk file. You can specify parasitics in either a reduced form or a detailed form.

Reduced parasitics consist of an RC pie model specified for each driver pin of a net. An RC pie model contains two capacitances and one resistance. The first capacitance connects to the net driver pin. The resistance connects to the net driver pin and to a subnode to which the second capacitance connects. Both capacitances connect to ground. The parasitics file using the reduced form also specifies the pin-to-pin net delays.

Detailed parasitics consist of a network of resistances and capacitances for each net specified in the parasitics file. The capacitances must be with respect to

ground. Coupling capacitances are not supported. Resistances between the nets and ground are ignored. Delays are computed via the Elmore delay model (use the **-elmore** option) or the Arnoldi delay model (use the **-arnoldi** option), respectively.

Net and instance pin names in the design must match instance names in the parasitics file. For example, if you create the parasitics file from a design using VHDL naming conventions, the design name must use them, too.

To remove loads and delays annotated by using the **read_parasitics** command, use the **reset_design** or **remove_annotated_delay** command.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following command reads the parasitics file named *adder.spef* from disk and uses it to annotate the computed Elmore delays to pins and loads to nets. The parasitics file contains a description of the RC network for nets of a design.

```
prompt> read_parasitics adder.spef -elmore
```

The following example removes annotated delays and loads from the current design.

```
prompt> remove_annotated_delay
```

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  
[-format db | ddc]  
design_name  
-source pass  
-type pre | post
```

ARGUMENTS

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

read_pin_map

Reads in a port-to-pin mapping file, which defines the design port-to-package pin mapping for a boundary-scan design.

SYNTAX

```
status read_pin_map  
path_name
```

Data Types

path_name string

ARGUMENTS

path_name
The full name of the file containing the port-to-pin mapping for the design.

DESCRIPTION

The **write_bsd1** command requires the design port-to-package pin mapping information in order to generate a boundary-scan description language (BSD1) file for a boundary-scan design. The **read_pin_map** command allows you to read in a port-to-pin mapping file for a boundary-scan design. All the signal ports in the design must be listed in the port-to-pin mapping file. If a port listed in the port-to-pin mapping file is not a port of the design, it is assumed to be a linkage port.

To remove all port-to-pin mapping for the a design, use **remove_pin_map**. To list all the port-to-pin mappings read in for the design, use **report_packages**.

EXAMPLES

The following is an example of a port to pin mapping file for a design:

```
PACKAGE = test_pkg;  
  
PORT = OUT5, PIN = (P41, P40, P39, P38, P37, P36);  
PORT = OUT2, PIN = (P26, P27, P28, P29, P30, P31);  
PORT = IN1, PIN = P1;  
PORT = IN2, PIN = P2;  
PORT = IN3, PIN = P3;  
PORT = EN, PIN = P4;  
PORT = RESETN, PIN = P5;  
PORT = IN4, PIN = P6;  
PORT = CLK, PIN = P7;  
PORT = CLOCKDR, PIN = P8;  
PORT = configENABLE, PIN = P9;  
PORT = jtag_tdi, PIN = P10;  
PORT = jtag_tdia, PIN = P11;  
PORT = jtag_tms, PIN = P12;
```

```

PORT = jtag_tck, PIN = P13;
PORT = jtag_trst, PIN = P14;
PORT = configCAPTUREx, PIN = P15;
PORT = configENABLEx, PIN = P16;
PORT = OUT3, PIN = P17;
PORT = OUT6, PIN = P18;
PORT = OUT6enable, PIN = P19;
PORT = configCAPTURE, PIN = P20;
PORT = jtag_tdo, PIN = P21;
PORT = OUT1(3), PIN = P22;
PORT = OUT1(2), PIN = P23;
PORT = OUT1(1), PIN = P24;
PORT = OUT1(0), PIN = P25;
PORT = OUT4(3:0), PIN = (P32, P33, P34, P35);
PORT = CONFIGURATION(3:0), PIN = (P42, P43, P44, P45);
PORT = VDD, PIN = (P46, P48, P50);
PORT = GND, PIN = (P47, P49, P51);

```

The following example reads in a design port-to-package pin mapping file for a boundary-scan design.

```
prompt> read_pin_map bscan.map
```

SEE ALSO

```

get_attribute(2)
remove_pin_map(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)
saif_map(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)
```

```
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_sverilog

Reads in one or more design or library files in SystemVerilog format.

SYNTAX

```
string read_sverilog  
file_names  
[-netlist]  
[-rtl]
```

Data Types

file_names list

ARGUMENTS

file_names
 Specifies the names of one or more files to be read.

DESCRIPTION

The **read_sverilog** command reads design information from SystemVerilog files into dc_shell.

This command is derived from **read_file -format sverilog**. For more information, see the man page for the **read_file** command.

SEE ALSO

read_file(2)

read_test_model

Reads a test model file.

SYNTAX

```
list read_test_model  
[-format ddc | ctl]  
[-design design_name]  
model_files
```

Data Types

<i>design_name</i>	string
<i>model_files</i>	list

ARGUMENTS

-format ddc | ctl
Specifies the format of the test model. A test model can be stored in a design .ddc file (default) or as ASCII text in a Core Test Language (CTL) file. By default, the model is assumed to be in .ddc format.

-design *design_name*
Specifies the name of the design to attach the test model to. This is meaningful only for CTL test models. By default, the model is attached to the current design.

model_files
Specifies the names of the test model files to read. This argument is required. When you specify the format as **ctl**, you can specify only one file.

DESCRIPTION

The **read_test_model** command reads test model files.

The test model files represent the test behavior of designs. The attributes that are represented include all DFT signals and their purpose, the test timing, and the sequence of events needed to shift test vectors in and out of the design. For more information about attributes that may be represented, see the *DFT Compiler User Guide: Scan* and the *DFT MAX User Guide: Adaptive Scan*.

When the format is specified as **ddc**, you can specify multiple model files to read. Each model of each .ddc file is read in. Since .ddc models include the designation of the designs they model, the **-design** option is ignored.

The **read_test_model** command reads in only the test model and the interface definition of a design, and ignores any other information. Since the gate implementation of a design is not loaded in the database, significant memory savings can be achieved. For a .ddc file containing only design interface definitions and test models, the **read_file** and **read_test_model** commands are synonymous.

When the format is specified as **ctl**, a single test model is read from a CTL (ASCII) file and is associated with the current design, or the design specified through the **-design** option. The CTL format is used for test models generated outside of a Synopsys environment.

A successful read of a model overwrites any test model information in memory.

Test models are stored on disk using the **write_test_model** or the **write** command. Transfer test models from one session to another either by storing full designs, or by storing only test models. Unless there is a requirement to hide a design's implementation, it is best to store the full designs, since it reduces the risk of confusion about the contents of the .ddc files.

EXAMPLES

The following command sequence stores full designs to disk. It facilitates external data management by having a single .ddc file for a design.

```
prompt> write -format ddc MyDesign -out MyDesign.ddc

prompt> remove_design MyDesign

prompt> read_test_model MyDesign.ddc
```

The following command sequence stores only the test model to a disk file. It requires the separate management of the implementation of the design.

```
prompt> write_test_model MyDesign -out MyDesign.ddc

prompt> remove_design MyDesign

prompt> read_file MyDesign.ddc
```

The last **read_file** command above is equivalent to a **read_test_model** command, because the *MyDesign.ddc* file contains only the design interface and the test model.

SEE ALSO

```
list_test_models(2)
remove_test_model(2)
report_test_model(2)
write_test_model(2)
current_design(3)
```

read_test_protocol

Reads a test protocol file into memory.

SYNTAX

```
int read_test_protocol
[-verbose]
[-test_mode test_mode_name]
[-overwrite]
[-section section_name]
input_file_name
```

Data Types

<i>test_mode_name</i>	string
<i>section_name</i>	string
<i>input_file_name</i>	string

ARGUMENTS

- verbose
Specifies that verbose output is displayed when set to **true**.
- test_mode *test_mode_name*
Specifies a test mode name to read and store a test protocol. If not specified, the current test mode is used by default.
- overwrite
Specifies to overwrite timing, procedures, or macros if they already exist in memory when reading in a new test protocol. For example, if there is already a protocol in the memory with timing specified, and you read in a new protocol with different timing, this option specifies to overwrite the existing timing with the new timing.
- section *section_name*
Specifies a section name of the protocol. With this option, the **read_test_protocol** command reads only the specified section and ignores others. The only valid parameter is **test_setup**, which describes the initialization sequence.
- input_file_name*
Specifies the name of the test protocol file in STIL format to read. The default file name is *design_name.spf*.

DESCRIPTION

Reading a test protocol file defines the scan test protocol for the current design. The scan test protocol provides a means to formally define the process by which a design is tested, such as the sequence of scan-in vectors, parallel vectors, and scan-out vectors performed for each test pattern. The protocol defined for a design is used to drive scan test design rule checking.

The format of the test protocol file is STIL. See the DFT Compiler/DFT MAX and TetraMAX user guides for details of the protocol file format.

The **read_test_protocol** command enables you to define an arbitrary test protocol for a design by reading in a text file describing the protocol. Use **write_test_protocol** to write out an existing protocol for a design in the same text format.

This command removes any protocol that may be present in memory through the previous use of the **read_test_protocol**, **read_init_protocol** or **create_test_protocol** command.

If the test protocol file name is not specified for this command, the default is *design_name.spf*.

With the STIL protocol format, you cannot distinguish between a clock signal and an asynchronous signal. You must define asynchronous signals with the **set_dft_signal** command before reading your custom STIL protocol.

The following example defines the asynchronous signal *rstn* for design *UPC1* and reads a test protocol from the file *UPC1.spf*.

```
prompt> set_dft_signal -type Reset -active_state 0 -port rstn
prompt> read_test_protocol UPC1.spf
```

EXAMPLES

The following example reads a test protocol from the *newUPC1.spf* file:

```
prompt> read_test_protocol newUPC1.spf
Reading test protocol file 'newUPC1.spf'...
```

The following example reads a test protocol for the *UPC1* design from the *UPC1.spf* default file:

```
prompt> read_test_protocol
Reading test protocol file 'UPC1.spf'...
```

The following example reads a test_setup section of the protocol. For example, assume that the *init_sequence.spf* file has the following test_setup vector sequence:

```
MacroDefs {"test_setup" {W "_default_WFT";
    V {"A" = 0; "B" = 0; "CP" = 0;
}
    V {"A" = 1; "B" = 1; "CP" = P;
}
    V {"A" = 0; "B" = 0; "CP" = 1; "AA" = 1; "BB" = 1;
}
}
}
```

The following command reads in the initialization sequence and ignores the rest of the protocol file:

```
prompt> read_test_protocol -section test_setup init_sequence.spf
```

SEE ALSO

`create_test_protocol(2)`
`dft_drc(2)`
`remove_test_protocol(2)`
`set_dft_signal(2)`
`write_test_protocol(2)`
`current_design(3)`

read_verilog

Reads in one or more design or library files in Verilog format.

SYNTAX

```
status read_verilog  
[-netlist | -rtl]  
verilog_files
```

Data Types

verilog_files list

ARGUMENTS

-netlist

Specifies that the design being read is a structural or gate-level design. This option invokes a netlist reader optimized for gate-level designs. If any of the specified files contain RTL-level constructs, an error is issued. Use of the **-netlist** option might enable the tool to read larger gate-level designs than would otherwise be the case.

This option is mutually exclusive with the **-rtl** option.

-rtl

Specifies that the design being read is an RTL design. This option invokes the Presto RTL Verilog reader directly without attempting to determine if the specified files are gate-level netlists. For details, see the **hdlin_auto_netlist_reader** variable man page.

This option is mutually exclusive with the **-netlist** option.

verilog_files

Specifies the name of one or more Verilog files to be read.

DESCRIPTION

The **read_verilog** command reads design information from Verilog files into memory.

If the automatic mode is on, the netlist reader (read with the **-netlist** option) is automatically invoked. If the input contains RTL constructs, the system automatically invokes the RTL reader (read with **-rtl** option).

This command is derived from **read_file -format verilog**. For more information, see the man page for the **read_file** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

SEE ALSO

read_verilog
706

read_vhdl

Reads in one or more design or library files in VHDL format.

SYNTAX

```
string read_vhdl  
[-netlist]  
file_names
```

Data Types

file_names list

ARGUMENTS

-netlist

It indicates that the design being read is a structural or gate-level design. This option can only be used in conjunction with **-f verilog** or **-f vhdl**. This option invokes a netlist reader optimized for gate-level designs. If any of the given files contain RTL-level constructs, an error is issued. Use of the **-netlist** option may enable `dc_shell` to read larger gate-level designs than would otherwise be the case.

This option is only available if the **hdlin_enable_presto_for_vhdl** is switched on. This variable is switched off by default.

file_names

Specifies the names of one or more files to be read.

DESCRIPTION

This command reads design information from VHDL files into the appropriate shell. It is derived from the **read_file** command as used for VHDL. For more information, see the **read_file** command man page.

SEE ALSO

`read_file(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)`

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.

redirect

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)
```

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.

remove_annotated_check

-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

remove_annotated_delay

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)

```
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)`
`find(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_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
[-bus]
[-quiet]
object_list
attribute_name
```

Data Types

<i>object_list</i>	list
<i>attribute_name</i>	string

ARGUMENTS

-bus
Indicates to remove the attribute from the bus as a whole, and not from each individual bus member.

-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.

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 returns the list of objects from which the attribute is 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

This command removes the **output_not_used** attribute on a port.

```
prompt> remove_attribute [get_ports OUT1] output_not_used
```


{OUT1}

In the following example, a warning message is issued because the cell "U9" cannot be found.

```
prompt> remove_attribute [get_cells {U1 U9}] dont_touch
Warning: Can't find object 'U9' in design 'example'
{U1}
```

Using the **-quiet** option with the previous example inhibits the warning messages.

```
prompt> remove_attribute -quiet [get_cells {U1 U9}] dont_touch
{U1}
```

In this example, the **dont_use** attribute of a library cell is removed.

```
prompt> remove_attribute [get_lib_cells tech_lib/GATE] dont_use
{tech_lib/GATE}
```

In the following example, the specified attribute cannot be found.

```
prompt> remove_attribute [get_cells *] arrival
```

SEE ALSO

collections(2)
define_user_attribute(2)
get_attribute(2)
list_attributes(2)
reset_design(2)
set_attribute(2)

remove_boundary_cell

Removes boundary cell configuration for the specified ports or core cells.

SYNTAX

```
status remove_boundary_cell  
-class core_wrapper | shadow_wrapper | bsd  
-ports port_list | -core core_cell_list  
[-function function_type]
```

Data Types

<i>port_list</i>	list
<i>core_cell_list</i>	string
<i>function_type</i>	string

ARGUMENTS

-class *core_wrapper* | *shadow_wrapper* | *bsd*
Specifies the name of the class for which the boundary cell configuration is to be removed.
Valid values for this option are **core_wrapper**, **shadow_wrapper**, and **bsd**.
This is a required argument.

-ports *port_list*
Specifies the list of ports for which the boundary cell configuration is to be removed.
There is no default value for this option. You must use either the **-ports** option or the **-core** option.

-core *core_cell_list*
Specifies the list of core cells for which the boundary cell configuration is to be removed.
When the *core_list* is not empty, the *port_list* specifies the ports of these core cells.
There is no default value for this option.

-function *function_type*
Specifies the function of the specified ports for which the boundary cell configuration is to be removed.
Valid values for this option are as follows:
input
output
control
bidir
observe
input_inverted
output_inverted
bidir_inverted
There is no default value for this option.

DESCRIPTION

The **remove_boundary_cell** command removes boundary cell configuration on the specified core cells or ports for the specified class. The core cells or ports must exist in the current design.

EXAMPLES

The following command sets the boundary cell configuration on *port1*:

```
prompt> set_boundary_cell -class core_wrapper -type WC_D1 \  
        -safe_state none -port_list port1  
1
```

The following command sets the boundary cell information on *port2*:

```
prompt> set_boundary_cell -class core_wrapper -type WC_S1_S \  
        -safe_state 0 -port_list port2  
1
```

The following command removes the boundary cell configuration for the core wrapper class on the *port1* and *port2* ports:

```
prompt> remove_boundary_cell -class core_wrapper \  
        -port_list {port1, port2}  
1
```

SEE ALSO

insert_dft(2)
preview_dft(2)
set_boundary_cell(2)
set_bsd_configuration(2)
set_wrapper_configuration(2)

remove_boundary_cell_io

Removes the boundary cell IO specifications from the current boundary-scan design.

SYNTAX

```
status remove_boundary_cell_io  
[-all]  
[-cell cell_list]
```

Data Types

cell_list string

ARGUMENTS

-all
Removes all boundary cell IO specifications from the design.

-cell *cell_list*
Removes all user-defined boundary IO cells specified in the current design.

DESCRIPTION

The **remove_boundary_cell_io** command removes the boundary cell IO specifications from the current design. If you invoke **remove_boundary_cell_io** with no options, the command is ignored. You receive an informational message describing this action.

EXAMPLES

The following example removes the boundary cell IO specification from the design:

```
prompt> set_boundary_cell_io -type boundary \  
          -cell what_testcase_bs_core_inst/bsr_gpio_3_inst \  
          -access {out_pi what_testcase_bs_core_inst/U21/Z \  
                  out_po what_testcase_bs_core_inst/bsr_gpio_3_inst/U11/Z}  
  
prompt> remove_boundary_cell_io -cell {what_testcase_bs_core_inst/  
bsr_gpio_3_inst}
```

The following example removes all boundary cell IO specifications from the design:

```
prompt> set_boundary_cell_io -type boundary \  
          -cell what_testcase_bs_core_inst/bsr_gpio_3_inst \  
          -access {out_pi what_testcase_bs_core_inst/what_testcase_core_inst/U21/  
Z \  
                  out_po what_testcase_bs_core_inst/bsr_gpio_3_inst/U11/Z}  
  
prompt> set_boundary_cell_io -type boundary \
```

```
-cell what_testcase bs_core_inst/bsr_gpio_0_inst \  
-access {out_pi what_testcase bs_core_inst/what_testcase_core_inst/U19/  
Z \  
    out_po what_testcase bs_core_inst/bsr_gpio_0_inst/U22/Z}  
  
prompt> remove_boundary_cell_io -all
```

SEE ALSO

```
report_boundary_cell_io(2)  
set_boundary_cell_io(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)  
report_bounds(2)  
set_cell_location(2)  
update_bounds(2)
```

remove_bsd_compliance

Removes the Compliance Ports specifications from the current boundary-scan design.

SYNTAX

```
status remove_bsd_compliance  
[-all]  
[-name pattern_name]
```

Data Types

pattern_name string

ARGUMENTS

-all
Removes all compliance port specifications from the design.

-name *pattern_name*
Removes all user-defined pattern names specified if present in the current design.

DESCRIPTION

The **remove_bsd_compliance** command removes the compliance ports specifications from the current design. If you invoke **remove_bsd_compliance** with no options, the command is ignored. You receive an informational message describing this action.

EXAMPLES

The following example sets and then removes the compliance port specification from the design:

```
prompt> set_bsd_compliance -name pat1 -pattern {my_bidi 0 my_eni 1}  
  
prompt> remove_bsd_compliance -name pat1
```

The following example sets and then removes all compliance ports specifications from the design:

```
prompt> set_bsd_compliance -name pat1 -pattern {my_bidi 0 my_eni 1}  
  
prompt> set_bsd_compliance -name pat2 -pattern {my_ini 0}  
  
prompt> remove_bsd_compliance -all
```


SEE ALSO

`report_bsd_compliance(2)`
`set_bsd_compliance(2)`

remove_bsd_instruction

Removes boundary-scan instructions from the instruction list to be used by **insert_dft** for the current design.

SYNTAX

```
int remove_bsd_instruction  
instruction_list
```

Data Types

```
instruction_list      list
```

ARGUMENTS

```
instruction_list  
    A comma-separated list containing one or more boundary scan instructions to  
    remove from the set of boundary scan instructions for the current design.
```

DESCRIPTION

The **remove_bsd_instruction** command specifies one or more boundary scan instructions to be removed from the set of boundary scan instructions to be implemented by **insert_dft** for the current design. At present, you can specify only the BYPASS, EXTEST, SAMPLE, IDCODE, CLAMP, and HIGHZ instructions using this command.

EXAMPLE

The following example removes IDCODE and CLAMP from the list of boundary scan instructions to be used by the **insert_dft** command.

```
prompt> remove_bsd_instruction {IDCODE, CLAMP}
```

SEE ALSO

```
insert_dft(2)  
set_bsd_instruction(2)
```

remove_bsd_linkage_port

Removes the Linkage Ports specifications from the current boundary-scan design.

SYNTAX

```
status remove_bsd_linkage_port  
[-all]  
[-port_list port_list]
```

Data Types

port_list string

ARGUMENTS

-all
Removes all linkage port specifications from the design.

-port_list *port_list*
Removes all user-defined linkage ports specified in the current design.

DESCRIPTION

The **remove_bsd_linkage_port** command removes the linkage ports specifications from the current design. If you invoke **remove_bsd_linkage_port** with no options, the command is ignored and you receive an informational message.

EXAMPLES

The following example removes the linkage port specification from the design.

```
prompt> set_bsd_linkage_port -port_list {my_bidi my_ini out1}  
  
prompt> remove_bsd_linkage_port -port_list {my_bidi out1}
```

The following example removes all linkage ports specifications from the design.

```
prompt> set_bsd_linkage_port -port_list {my_bidi my_ini out1}  
  
prompt> remove_bsd_linkage_port -all
```

SEE ALSO

report_bsd_linkage_port(2)
set_bsd_linkage_port(2)

remove_bsd_power_up_reset

Removes the power up reset port specifications from the current boundary-scan design.

SYNTAX

```
status remove_bsd_power_up_reset
```

ARGUMENTS

The **remove_bsd_power_up_reset** command has no arguments.

DESCRIPTION

The **remove_bsd_power_up_reset** command removes the power up reset port specifications from the current design.

EXAMPLES

The following example removes the power up reset port specifications from the design:

```
prompt> set_bsd_power_up_reset -reset_pin_name in0 -active high \  
         -delay 500 -cell_name U15
```

```
prompt> remove_bsd_power_up_reset
```

SEE ALSO

```
report_bsd_power_up_reset(2)  
set_bsd_power_up_reset(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_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_bus(2)`
`create_net(2)`
`remove_net(2)`

remove_cache

Selectively removes elements from the synthetic library cache directories.

SYNTAX

```
int remove_cache
[-design_lib list] [-module list]
[-implementation list]
[-parameters parameter_list]
[-tech_lib list]
[-wire_load list]
[-operating_conditions list]
[-directory dir_list]
[-smaller size | -larger size]
[-netlist_only | -model_only]
[-accessed_since days
 | -accessed_beyond days]
```

Data Types

<i>list</i>	<i>dir_list</i>
<i>parameter_list</i>	<i>list</i>
<i>dir_list</i>	<i>list</i>
<i>size</i>	<i>int</i>
<i>days</i>	<i>float</i>

ARGUMENTS

- design_lib *list*
Restricts the deletion to the design libraries in the list. Strings in the list can contain the "*" wildcard character (eg: { DW_01 DW_* *SNPS* }).
- module *list*
Restricts the deletion to the modules in the list. Strings in the list can contain the "*" wildcard character.
- implementation *list*
Restricts the deletion to the implementations in the list. Strings in the list can contain the "*" wildcard character.
- parameters *parameter_list*
Restricts the deletion to the parameters in the list. See the Description section for more information on *parameter_list*.
- tech_lib *list*
Restricts the deletion to the technology libraries in the list. Strings in the list can contain the "*" wildcard character.
- wire_load *list*
Restricts the deletion to the wire loads in the list. Strings in the list can contain the "*" wildcard character.

-operating_conditions *list*
 Restricts the deletion to the operating conditions in the list. Strings in the list can contain the "*" wildcard character.

-directory *dir_list*
 Uses the given list of directories, instead of the cache-read and cache-write variables, as the cache directories from which to delete.

-smaller *size*
 Restricts the deletion to cache elements that are strictly smaller than *size* bytes.

-larger *size*
 Restricts the deletion to cache elements that are strictly larger than *size* bytes.

-netlist_only
 Deletes netlists only.

-model_only
 Deletes models only.

-accessed_since *days*
 Restricts the deletion to cache elements whose last access time has been from now to *days* days ago.

-accessed_beyond *days*
 Restricts the deletion to cache elements whose last access time is older than *days* days.

DESCRIPTION

Selectively removes cache elements from synthetic library cache directories. The command also removes any empty directories it encounters in the cache directories. Without command arguments, all cache elements in the cache directories are removed. If there are command arguments that restrict the deletion, the cache element is deleted only if it matches each command argument specified. An element matches a command argument if it matches one of the items in the command argument's list. Without the **-directory** command argument, the cache directories are the ones listed in the **cache_read** and **cache_write** variables.

For a description of parameter matching and an explanation of netlists and models in the cache, refer to the **report_cache** man page. Except for the various output formats in **report_cache**, any command line option for **report_cache** can also be used for **remove_cache**.

For more information, see the DesignWare documentation set.

EXAMPLES

None.

SEE ALSO

report_cache(2)
create_cache(2)
report_synlib(2)
set_wire_load_model(2)
set_wire_load_mode(2)
set_wire_load_selection_group(2)
set_wire_load_min_block_size(2)
cache_read(3)
cache_write(3)

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
```

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.

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)
```


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)  
compile(2)  
remove_attribute(2)  
report_constraint(2)  
set_max_capacitance(2)  
compile_fix_cell_degradation(3)
```

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_gating

Directs the **compile -incremental**, and **compile_ultra -incremental** commands to remove clock gating from objects clock-gated by Power Compiler.

SYNTAX

```
integer remove_clock_gating  
[-gated_registers gated_register_list]  
[-min_bitwidth minsize_value]  
[-gating_cells clock_gating_cells_list]  
[-all]  
[-no_hier]  
[-verbose]  
[-undo]
```

Data Types

<i>gated_register_list</i>	list
<i>minsize_value</i>	integer
<i>clock_gating_cells_list</i>	list

ARGUMENTS

-gated_registers *gated_register_list*
Specifies that all registers in the *gated_register_list* are ungated.

-min_bitwidth *minsize_value*
Specifies that all registers in banks with a size smaller than the *minsize_value* will be ungated.

-gating_cells *clock_gating_cells_list*
Specifies that all objects gated by clock-gating cells in the *clock_gating_cells_list* are ungated and removes clock-gating cells from the design.

-all
Removes clock gating from all levels of hierarchy, unless the **-no_hier** option is used, in which case clock gating is only removed from the top level of the current design.

-no_hier
Removes clock gating from the top level of the current design only when used with the **-all** or **-min_bitwidth** options.

-verbose
Prints additional information as the command executes.

-undo
Removes directives specified by any previously executed **remove_clock_gating** commands. Based on the specified options, this option deletes directives specified by previously executed **remove_clock_gating** commands. Use the **-undo** option before the **compile -incr** command modifies the netlist to remove clock

gating.

DESCRIPTION

This command directs the **compile -incremental**, and **compile_ultra -incremental** commands to remove clock gating from objects (registers and clock-gating cells) clock gated by Power Compiler. It provides a mechanism to selectively remove clock gating from various parts of the design. Power Compiler performs clock gating at the register transfer level (RTL) during execution of the **compile_ultra -gate_clock** command.

Further down the compile flow, you might have to selectively remove some clock-gating elements from the design. The **remove_clock_gating** command provides a mechanism to do so without having to start at RTL again.

This command provides directives to the **compile** and **compile_ultra** commands used with the **-incremental** option that perform circuit modifications to remove clock gating from the design. See the man pages for the **compile** and **compile_ultra** commands for more information on the **-incremental** options.

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 ungated are remapped to new sequential cells. This might result in new pin names for the registers. A register can be clock-gated by multiple clock gates in a multi-stage clock-gating design. If the register is ungated, all clock gating on the clock path for this register is removed. However, if a single clock gate is specified for removal, 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 transformation if the new and old pin names do not match. The **remove_clock_gating** command issues a warning if there are pin-based timing exceptions on the register to be ungated.

Although it is not required, it is good practice to run the **report_timing** command again to confirm that the pin-based exceptions are being properly applied after the registers are ungated during execution of the **compile** command with the **-incremental** option.

EXAMPLES

The following example provides a directive to the **compile -incremental** command to remove clock gating from all the registers gated by the cell named **clk_gate_pipeline_reg_A** and removes the cell:

```
prompt> remove_clock_gating -gating_cell \  
        clk_gate_pipeline_reg_A
```

The following example provides a directive to the **compile -incremental** command to remove 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 cell, it is also removed.

```
prompt> remove_clock_gating \  
        -gated_registers {pipeline_reg_A[0] pipeline_reg_A[2]}
```

The following example undoes the effect of the directive in the first example above, to remove the clock-gating cell named *clk_gate_pipeline_reg_A* and ungate all registers gated by this cell. Run this command before running the **compile -incremental** command.

```
prompt> remove_clock_gating -undo \  
        -gating_cell clk_gate_pipeline_reg_A
```

The following example provides a directive to the **compile -incremental** command to remove clock gating from all gated registers and to remove all clock-gating cells from the current level of the design hierarchy:

```
prompt> remove_clock_gating -all -no_hier
```

The following example provides a directive to the **compile -incremental** command to remove clock gating from all gated registers and remove all clock-gating cells from all designs at or below the current level of the design hierarchy:

```
prompt> remove_clock_gating -all
```

SEE ALSO

```
compile(2)  
compile_ultra(2)  
set_clock_gating_style(2)  
set_false_path(2)  
set_max_delay(2)  
set_min_delay(2)  
set_multicycle_path(2)
```

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.

remove_clock_groups

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_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_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_constraint

Removes all constraint attributes, clocks, and path delay information from the current design. Note that this command will be obsolete in the next release. It will be replaced by the **remove_sdc** command. Please adjust your scripts accordingly.

SYNTAX

```
int remove_constraint
-all
[-all_sdc]
```

ARGUMENTS

-all
Removes all target values from the current design.

-all_sdc
Removes all sdc constraints. By default, this option is off.

DESCRIPTION

This command removes **max_area**, **max_power**, **max_fanout**, and **max_transition** attributes from the current design. It also removes clock objects and all path delay information created by the **set_max_delay**, **set_min_delay**, **set_false_path**, **set_max_time_borrow**, and **set_multicycle_path** commands.

You can undo individual commands by using the **remove_attribute** or **reset_path** command. To remove clocks, you can use the **remove_clock** command.

The **reset_design** command removes all constraints and all attributes.

Multicorner-Multimode Support

This command uses information from active scenarios only.

EXAMPLES

The following example removes all constraint information from the current design.

```
prompt> remove_constraint -all
```

SEE ALSO

```
current_design(2)
remove_attribute(2)
remove_clock(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_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

```
get_cts_scenario(2)
set_cts_scenario(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)`
`rename_design(2)`

remove_dft_connect

Removes existing DFT connectivity specifications.

SYNTAX

```
status remove_dft_connect  
label_name | -all  
[-all]
```

Data Types

label_name string

ARGUMENTS

label_name
Removes only the connectivity specification with the associated *label_name*.
This argument is mutually exclusive with **-all**. You must specify one, but not both.

DESCRIPTION

The **remove_dft_connect** command is used to remove DFT connectivity associations specified by the **set_dft_connect** command.

To remove a specific connectivity specification, specify the *label_name* of the association as the argument.

To remove all connectivity specifications, specify **-all**.

You must specify either *label_name* or **-all**.

EXAMPLES

The following example first sets 3 connectivity specifications, then removes the specification associated with label CON2, and finally removes all specified connectivity by using **-all** option:

```
prompt> set_dft_connect CON1 -source SE1 -type clock_gating_control \  
-target [list U1 U2]  
  
prompt> set_dft_connect CON2 -source SE2 -type clock_gating_control \  
-target [list U3 U4] -exclude U3/u  
  
prompt> set_dft_connect CON3 -source SE3 -type scan_enable \  
-target [list ff1 ff2]  
  
prompt> remove_dft_connect CON2
```

```
prompt> remove_dft_connect -all
```

SEE ALSO

```
report_dft_connect(2)  
set_dft_connect(2)  
set_dft_signal(2)
```

remove_dft_design

Removes the specified design so that it cannot be used for DFT insertion.

SYNTAX

```
integer remove_dft_design  
-design_name design_name  
  | -type design_type  
  | -all
```

Data Types

```
design_name      string  
design_type      string
```

ARGUMENTS

```
-design_name design_name  
            Identifies the design to be removed.  
  
-type design_type  
            Specifies the type of the DFT design to be removed. Valid types are DECODE,  
            CONTROL_FORCE, Z_CONTROL_FORCE, OBSERV_DGEN, SHIFT_REG, BC1, BC2, BC3, BC4,  
            BC5, BC7, BC8, BC9, BC10, WC_D1, WC_D1_S, WC_S1, WC_S1_S, INSTRREG, TAP,  
            TAP_UC, LBIST_CONTROLLER, LBIST_CODEEC, LBIST_XCONTROLLER, LBIST_XCODEC,  
            CLK_MUX, CLK_CHAIN, MBIST_CONTROLLER, MBIST_WRAPPER, and PAD.  
  
-all  
            Removes all DFT designs.
```

DESCRIPTION

The **remove_dft_design** command removes a design so that it is not used for DFT insertion.

You must use one of the three options: **-design_name**, **-type**, or **-all**. These options are mutually exclusive; use only one option for each invocation of the command.

EXAMPLES

The following example removes *my_des* so that it is not used for DFT insertion:

```
prompt> define_dft_design -design_name my_des \  
        -type WC_D1 -interface \  
        {shift_clk capture_clk h capture_en capture_en h \  
        shift_en shift_dr h cti si h cto so h cfi \  
        data_in h cfo data_out h  
prompt> ....  
prompt> remove_dft_design -design_name my_des
```

The following example removes all DFT designs of type BC4:

```
prompt> define_dft_design -design my_des1 -type BC4 \  
        -interface {capture_clk cap_clk h capture_en cen h \  
        shift_dr shift h si ti h so to h data_in di h data_out do h  
prompt> define_dft_design -design my_des2 -type BC4 \  
        -interface {capture_clk cap_clk h capture_en cen h \  
        shift_dr shift h si ti h so to h data_in di h data_out do h  
prompt> remove_dft_design -type BC4
```

SEE ALSO

```
define_dft_design(2)  
insert_dft(2)  
preview_dft(2)  
report_dft_design(2)
```

remove_dft_equivalent_signals

Removes all the equivalent dft signals specified with `set_dft_equivalent_signals` for the specified primary signal commands. Supported only in XG mode.

SYNTAX

```
int remove_dft_equivalent_signals  
primary_signal
```

ARGUMENTS

`primary_signal`
Specifies the primary signal used in **set_dft_equivalent_signals** to specify valid equivalences for the set of signals.

DESCRIPTION

This command removes the equivalent set of signals specified using **set_dft_equivalent_signals**. Use this command to remove a given equivalence specified thru **set_dft_equivalent_signals**.

EXAMPLES

The following example illustrates removing equivalent scan signals for `clk1`.

```
prompt> remove_dft_equivalent_signals clk1
```

SEE ALSO

```
preview_dft(2)  
insert_dft(2)  
set_dft_signal(2)  
set_dft_equivalent_signals(2)  
report_dft_equivalent_signals(2)
```

remove_dft_location

Removes the DFT Hierarchy location specification for the current design.

SYNTAX

```
integer remove_dft_location  
[-  
type DFTMAX | TCM | LOCKUP_LATCH | PIPELINE_SI_LOGIC | PIPELINE_SE_LOGIC | WRAPPER]
```

DESCRIPTION

The **remove_dft_location1** command can be used to remove DFT hierarchy location specification for the current design.

```
-type DFTMAX | TCM | LOCKUP_LATCH | PIPELINE_SI_LOGIC | PIPELINE_SE_LOGIC | WRAPPER  
    Specifies the types of DFT logic to be removed from their specified DFT  
    locations.  
    More than one type of DFT logic can be specified with this option.  
    There is no default value for this option. When this option is not specified,  
    default dft hierarchy location will be removed.
```

EXAMPLES

The following example removes DFT hierarchy location for design top.

```
prompt> current_design top  
prompt> set_dft_location core_inst/dft_inst  
prompt> remove_dft_location  
Removed dft location 'core_inst/dft_inst' for design 'top'  
1  
prompt>
```

SEE ALSO

```
insert_dft(2)  
set_dft_location(2)  
report_dft_location(2)
```

remove_dft_partition

Permanently removes a list of design DFT partitions associated with a design.

SYNTAX

```
status remove_dft_partition  
[list_of_partition_labels]
```

Data Types

list_of_partition_labels list

ARGUMENTS

list_of_partition_labels
Specifies the partitions to be removed from the specification.

DESCRIPTION

This command permanently removes a list of design DFT partitions associated with a design.

EXAMPLES

The following example removes the partitions defined as part1 and part2:

```
prompt> define_dft_partition part1 -instances {U1 U2}  
prompt> define_dft_partition part2 -instances {U3 U4}  
prompt> remove_dft_partition {part1 part2}
```

SEE ALSO

```
current_dft_partition(2)  
define_dft_partition(2)  
insert_dft(2)  
report_dft_partition(2)  
preview_dft(2)  
set_dft_signal(2)  
set_scan_compression_configuration(2)  
set_scan_configuration(2)  
set_scan_path(2)
```


remove_dft_signal

Removes from specified ports the attributes that identify those ports as DFT signals in the current design.

SYNTAX

```
int remove_dft_signal  
-view existing_dft | spec  
-port port_list  
-test_mode mode_list  
[-hookup_pin pin_name]
```

Data Types

<i>port_list</i>	list
<i>mode_list</i>	list
<i>pin_name</i>	string

ARGUMENTS

-view existing_dft | spec
Indicates the view to which the command applies. Valid views are **existing_dft** and **spec**.
Setting the view to **existing_dft** indicates that the command refers to the existing usage of a port. Use **existing_dft** to remove a specification made with the **set_dft_signal -view existing_dft** command.
Setting the view to **spec**, the default, indicates that the command refers to ports that the tool must use during DFT insertion. Use **spec** to remove a specification made with the **set_dft_signal -view spec** command.
This argument is required.

-port *port_list*
Specifies a list of ports to which the command applies.

-test_mode *mode_list*
Specifies the modes to which the command applies. The default mode is **Internal_scan**. Specifying **all** as the *mode_list* indicates that the specification applies to all modes. Any mode other than the default mode or **all** must be created before running the command with this option.

-hookup_pin *pin_name*
Specifies the hookup pin to which the command applies.

DESCRIPTION

The **remove_dft_signal** command removes from specified ports the attributes that identify those ports as DFT signal ports in the current design. These attributes were placed on the ports with the **set_dft_signal** command.

EXAMPLES

The following example removes the DFT signal attributes from the *TM* port:

```
prompt> remove_dft_signal -port TM
```

SEE ALSO

```
insert_dft(2)  
preview_dft(2)  
report_dft_signal(2)  
set_dft_signal(2)
```

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_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_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_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\(2\)](#)

```
report_clock(2)
```

remove_host_options

Remove any `-max_cores` specification set by the **set_host_options** command.

SYNTAX

status **remove_host_options**

ARGUMENTS

The **remove_host_options** command has no arguments.

DESCRIPTION

The **remove_host_options** command is used to unset the parameters set by the **set_host_options** command. The only such parameter is the `-max_cores` option. Running **remove_host_options** has the effect of disabling parallel execution.

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*.

```
remove_ideal_latency  
796
```



```
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.

```
remove_ideal_transition  
802
```

```
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_ignored_layers

Removes ignored routing layers in congestion analysis and RC estimation. This command is supported only in topographical mode.

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.

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_ignored_layers
804

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)`

`remove_input_delay`

```
set_output_delay(2)
```

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_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_multibit

Removes the multibit components from the current design. Removes or detaches cells from the multibit components in a design.

SYNTAX

```
status remove_multibit  
object_list
```

Data Types

```
object_list      list
```

ARGUMENTS

```
object_list  
    Specifies a list of multibit components, cells, and cell instances to remove  
    or detach. The objects must be from the current design. This argument is  
    required.
```

DESCRIPTION

This command deletes multibit components from the design and detaches cells or cell instances from multibit components.

Specify multibit components in the *object_list* to remove the component from the design. Specify cells or cell instances in the list to detach the cell from the multibit component in which the cell is contained.

This command does not delete cells from the design. It only manipulates multibit components in the design.

This command requires that the design be linked. If an attempt to link the design fails, the command is terminated.

EXAMPLES

The following example shows the multibit component status on the current design before any changes are made:

```
prompt> report_multibit  
  
*****  
Report : multibit  
Design : subtest  
Version: 2007.02  
Date   : Fri Aug 29 10:01:19 2007  
*****
```


Attributes:

b - black box (unknown)
h - hierarchical
n - noncombinational
r - removable
u - contains unmapped logic

Multibit Component : y_reg

Cell	Reference	Library	Area	Width	Attributes
y_reg[3]	**SEQGEN**		0.00	1	n, u
y_reg[2]	**SEQGEN**		0.00	1	n, u
y_reg[1]	**SEQGEN**		0.00	1	n, u
y_reg[0]	**SEQGEN**		0.00	1	n, u
Total 4 cells			0.00	4	

Multibit Component : z_reg

Cell	Reference	Library	Area	Width	Attributes
z_reg[3]	**SEQGEN**		0.00	1	n, u
z_reg[2]	**SEQGEN**		0.00	1	n, u
z_reg[1]	**SEQGEN**		0.00	1	n, u
z_reg[0]	**SEQGEN**		0.00	1	n, u
Total 4 cells			0.00	4	

Total 2 Multibit Components

This example first removes all multibit components and then shows the updated report:

```
prompt> remove_multibit ""
1
```

```
prompt> report_multibit
```

```
*****
Report : multibit
Design : subtest
Version: 2007.02
Date   : Fri Aug 29 10:01:19 2007
*****
```

Attributes:

b - black box (unknown)
h - hierarchical
n - noncombinational
r - removable
u - contains unmapped logic

Total 0 Multibit Components

This example first removes the specified cell from the multibit component. The component remains with its width reduced by 1. The example then shows the updated report:

```
prompt> remove_multibit z_reg[2]
```

```
prompt> report_multibit
```

```
*****
Report : multibit
Design : subtest
Version: 2007.02
Date   : Fri Aug 29 10:01:19 2007
*****
```

Attributes:

```
  b - black box (unknown)
  h - hierarchical
  n - noncombinational
  r - removable
  u - contains unmapped logic
```

Multibit Component : y_reg

Cell	Reference	Library	Area	Width	Attributes
y_reg[3]	**SEQGEN**		0.00	1	n, u
y_reg[2]	**SEQGEN**		0.00	1	n, u
y_reg[1]	**SEQGEN**		0.00	1	n, u
y_reg[0]	**SEQGEN**		0.00	1	n, u
Total 4 cells			0.00	4	

Multibit Component : z_reg

Cell	Reference	Library	Area	Width	Attributes
z_reg[3]	**SEQGEN**		0.00	1	n, u
z_reg[1]	**SEQGEN**		0.00	1	n, u
z_reg[0]	**SEQGEN**		0.00	1	n, u
Total 3 cells			0.00	3	

Total 2 Multibit Components

This example runs the command on a cell instance. The instance named U1/U2 is a unique instantiation of the design named BOT.

```
prompt> remove_multibit U1/U2/some_reg
1
```

This example runs the command on a multibit component from a subdesign instance. The instance named U1/U2 is a unique instantiation of the design named BOT.

```
prompt> remove_multibit U1/U2/mult_comp  
1
```

SEE ALSO

```
create_multibit(2)  
get_multibits(2)  
report_compile_options(2)  
report_multibit(2)  
set_multibit_options(2)
```

remove_net

Removes nets from the *current design*.

SYNTAX

```
int remove_net  
net_list | -all  
[-only_physical]
```

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.

-only_physical

Causes only physical nets to be removed. in the current design. The physical nets meaning those net do not connect to logical netlist.

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.

Use the **remove_bus** command to remove bused nets. Scalar (single bit) nets that are components of a bused net cannot be removed. You must remove bused nets first.

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 the physical nets:

```
prompt> remove_net -only_physical {physnet1 physnet2}
```

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_bus(2)
```

remove_operand_isolation

Removes the isolation logic inserted by Power Compiler during the operand isolation step.

SYNTAX

```
status remove_operand_isolation  
[-from from_list]  
[-to to_list]
```

Data Types

<i>from_list</i>	list
<i>to_list</i>	list

ARGUMENTS

-from *from_list*
Specifies the pins, ports, or nets from which a timing path starts.

-to *to_list*
Specifies the pins, ports, or nets on which a timing path ends.

DESCRIPTION

Use of the **remove_operand_isolation** command is not recommended. This feature will be obsolete in a future release. The **remove_operand_isolation** command removes the isolation logic on the specified timing path. A timing path is identified by the combination of *from_list* and *to_list*. You must specify either *from_list* or *to_list* or both. The command removes all isolation logic on the specified timing paths, independent of timing violations on these paths. The isolation logic on other timing paths is retained, even if there are timing violations on some of the paths.

To remove isolation logic on all timing paths that have violations, use the following commands:

```
prompt> set_operand_isolation_slack 0  
prompt> compile -incremental
```

EXAMPLES

The following example removes isolation logic on timing paths that end at the D pin of a register:

```
prompt> remove_operand_isolation -to [get_pins A_reg/D]
```

SEE ALSO

`set_operand_isolation_slack(2)`

`remove_operand_isolation`

```
set_operand_isolation_style(2)
do_operand_isolation(3)
```

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_pass_directories

Removes the data directories associated with the specified passes.

SYNTAX

```
int remove_pass_directories  
pass_list
```

ARGUMENTS

pass_list
A list of directory names that specify the passes whose data directories are removed.

DESCRIPTION

Removes the data directories associated with the specified passes. For use only in `dc_shell-t` (Tcl mode of `dc_shell`). This command deletes the specified directories under the Automated Chip Synthesis working directory (specified by the **`acs_work_dir`** variable) for each specified pass.

SEE ALSO

```
create_pass_directories(2)  
acs_work_dir(3)
```

remove_pin_map

Removes a design port-to-package pin mapping for a boundary-scan design.

SYNTAX

```
int remove_pin_map  
package_name
```

Data Types

package_name string

ARGUMENTS

package_name
The name of the package in a port-to-pin mapping file read in for the design.
This package name must match one of the package names specified in a port-to-pin mapping file previously read using the **read_pin_map** command.

DESCRIPTION

The **remove_pin_map** command allows you to delete a port-to-pin mapping for a boundary-scan design read in through the **read_pin_map** command.

Use **report_packages** to list the package names of the port-to-pin mappings files that you read in for the current design.

EXAMPLES

The following example deletes a design port to package pin mapping, *test_pkg*, for a boundary-scan design.

```
prompt> remove_pin_map test_pkg
```

SEE ALSO

```
get_attribute(2)  
read_pin_map(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
```

```

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_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.

Bussed ports cannot be removed with **remove_port**; use **remove_bus** to remove bused ports. Also, scalar (single bit) ports that are components of a bused port cannot be removed. In this case, you must first remove the bused port.

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_bus(2)
```


remove_power_domain

Removes the specified power domains.

SYNTAX

```
status remove_power_domain  
domain_name | -all
```

Data Types

domain_name string

ARGUMENTS

domain_name
Specifies the name of the power domain to be deleted. The name should be a simple (non-hierarchical) name.
In UPF mode, this is a required option. In non-UPF mode, 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.

-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 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] \
        -object_list [get_cells mid1]
1
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_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 **propagated_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_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.

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_rtl_load

Removes previously-set capacitance and resistance RTL load values from pins, ports, and nets.

SYNTAX

```
int remove_rtl_load  
-all | pin_net_list
```

Data Types

pin_net_list list

ARGUMENTS

-all

Specifies that all RTL loads in the current design are to be removed. You must specify either **-all** or *pin_net_list*; if you specify both, **-all** takes precedence.

pin_net_list

Specifies a list of ports, pins, and nets in the current design whose RTL loads are to be removed. You must specify either **-all** or *pin_net_list*; if you specify both, **-all** takes precedence.

DESCRIPTION

The **remove_rtl_load** command removes RTL load information, previously annotated by **set_rtl_load** or **calculate_rtl_load**, from the specified objects or from all objects in the current design. If a net is specified, the RTL loads are removed from all pins of the net.

The **reset_design** command removes all attributes from the current design, including all capacitance and resistance RTL load values.

EXAMPLES

The following example removes RTL load values from each of the specified pins and ports.

```
prompt> remove_rtl_load {my_port my_ram/ADDR[*]}
```

The following example removes RTL load values from the pins of the net net1.

```
prompt> remove_rtl_load my_hier/net1
```

The following example removes all RTL loads from the current design.

```
prompt> remove_rtl_load -all
```

SEE ALSO

`current_design(2)`
`reset_design(2)`
`set_rtl_load(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_group

Removes an existing scan group specification previously specified using the **set_scan_group** command.

SYNTAX

```
status remove_scan_group  
scan_group_name
```

Data Types

scan_group_name string

ARGUMENTS

scan_group_name
 Specifies the name of the scan group. The scan group must be a scan group previously defined using the **set_scan_group** command.

DESCRIPTION

This command removes an existing scan group specification previously specified using the **set_scan_group** command.

The **remove_scan_group** command cannot remove scan groups under these two circumstances:

- The scan group you want to remove has not been previously defined using the **set_scan_group** command.
- The scan group you want to remove was included in a **set_scan_path** specification.

In such cases, **remove_scan_group** exits with a warning.

EXAMPLES

The following is an example of the **remove_scan_group** command:

```
prompt> remove_scan_group my_shift_reg
```

SEE ALSO

```
insert_dft(2)  
preview_dft(2)  
report_scan_group(2)  
set_scan_group(2)
```

remove_scan_link

Removes a scan link specification for the current design.

SYNTAX

```
int remove_scan_link  
scan_link_name  
{Wire | Lockup}  
[-test_mode test_mode]
```

Data Types

<i>scan_link_name</i>	string
<i>test_mode</i>	string

ARGUMENTS

scan_link_name
Specifies the name of the scan link to be removed from the specification.

{Wire | Lockup}
Specifies the scan link type. **Wire** specifies a wire scan link type. **Lockup** specifies a lock-up latch scan link type.

-test_mode *test_mode*
Specifies the test mode of the specification. The default value is *current_mode*.

DESCRIPTION

The **remove_scan_link** command removes a scan link specified through the **set_scan_link** command. Scan links connect scan cells, scan segments, and scan ports within scan chains. DFT Compiler supports scan links that are implemented as wires and scan-out, lock-up latches.

The **scan_link_name** option must uniquely identify the scan link as a design object.

To see the scan links specified, use the **report_scan_link** command.

To review the scan link specifications, use the **preview_dft** command with the **-show** option set to **cells**.

To create scan links and add them to the current design, use the **insert_dft** command.

EXAMPLES

The following example declares a scan-out, lock-up latch named *a_lckup* and then removes it with **remove_scan_link**:

```
prompt> current_design WC66
```

```
prompt> set_scan_link a_lckup Lockup  
prompt> remove_scan_link a_lckup Lockup
```

SEE ALSO

```
current_design(2)  
report_scan_link(2)  
set_dont_touch(2)  
set_scan_configuration(2)  
set_scan_element(2)  
set_scan_link(2)  
set_scan_path(2)  
write(2)  
test_default_scan_style(3)
```


remove_scan_path

Remove the scan path specification for the current design in **set_scan_path**.

SYNTAX

```
int remove_scan_path  
[-chain scan_chain_name]  
[-view existing_dft | spec]  
[-test_mode test_mode]
```

Data Types

scan_chain_name string

ARGUMENTS

-chain *scan_chain_name*

Specifies to use the scan path name provided, which must be a valid, existing scan path name. If you do not specify a chain name, the command will error out.

-view existing_dft | spec

Indicates the view to which the specification applies. When you specify **-view existing_dft**, the specification removal refers to the existing usage of a scan chain. This is used when working with DFT inserted designs. For example, to inform that tool that chain A is being removed, use the following command:
prompt> **remove_scan_path -chain A -view existing_dft**

When you specify **-view spec**, the default value for **-view**, the specification refers to scan chains that the tool must use during DFT insertion. For example, to instruct the tool to remove scan chain A, use the following command:

prompt> **remove_scan_path -chain A -view spec**

-test_mode *test_mode*

Specifies the mode to which the specification applies. The default mode is *Internal_scan*. Specifying **all** to make the specification apply to all modes. Any mode other than the default mode or **all** must be created prior to use.

EXAMPLES

The following example sets a scan path using **set_scan_path** and then removes the scan path using **remove_scan_path** the spec view:

prompt> **current_design core**

prompt> **set_scan_path 1 -view spec**

prompt> **remove_scan_path -view spec -chain 1**

SEE ALSO

`insert_dft(2)`
`report_scan_path(2)`
`set_scan_path(2)`

remove_scan_register_type

Removes existing scan register types, previously set by **set_scan_register_type**, from specified cells or from the current design.

SYNTAX

```
integer remove_scan_register_type  
[cell_or_design_list]
```

Data Types

cell_or_design_list list

ARGUMENTS

cell_or_design_list
Lists the names of cells or designs from which to remove the **-type** scan register type specification. The default is the current design. Cells or designs on this list must already have had the **-type** specification placed on them by the **set_scan_register_type** command.

DESCRIPTION

The **remove_scan_register_type** command removes existing scan register types set on specified cells or designs by the **set_scan_register_type** command. The register types are removed without changing the design.

If you invoke **remove_scan_register_type** without any options, the specification on the current design is removed.

To determine the current settings resulting from the previous use of the **set_scan_register_type** command, execute **report_scan_register_type**.

EXAMPLES

The following example sets and then removes a **-type** specification on the cell U1:

```
prompt> set_scan_register_type -type {FD1S} U1  
prompt> remove_scan_register_type U1
```

SEE ALSO

```
current_design(2)  
set_scan_register_type(2)
```

remove_scan_replacement

Removes the table entries specified thru **set_scan_replacement**, a table of one-to-one mappings of flops to their equivalent scan flops from the target library that are to be used by **insert_scan** or **compile -scan** when scan replacing cell instances.

SYNTAX

```
int remove_scan_replacement  
[non_scan_seq_cell_list]
```

Data Types

non_scan_seq_cell_list list

ARGUMENTS

non_scan_seq_cell_list
The specified cells must be valid cells in the library and must have entries in the scan replacement table (entries are placed in the scan replacement table by the **set_scan_replacement** command).

DESCRIPTION

The **remove_scan_replacement** command removes the scan replacement entries for the cells in *non_scan_seq_cell_list* from the scan replacement table.

The scan replacement table entries are added by the **set_scan_replacement** command. Both **compile -scan** and **insert_scan** use this mapping table to do scan replacement. You can see the scan replacement table entries with the **report_scan_replacement** command.

EXAMPLES

In the following example, all the entries for FD1 are removed.

```
prompt> remove_scan_replacement FD1
```

SEE ALSO

```
current_design(2)  
set_scan_replacement(2)  
remove_scan_register_type(2)  
report_scan_replacement(2)  
reset_design(2)  
target_library(3)
```

remove_scan_suppress_toggling

Removes the existing user specifications that were provided through the **set_scan_suppress_toggling** command in terms of a list of scan flip-flops to be gated by the **insert_dft** command.

SYNTAX

```
status remove_scan_suppress_toggling
```

ARGUMENTS

The **remove_scan_suppress_toggling** command has no arguments.

DESCRIPTION

The **remove_scan_suppress_toggling** command removes all of the existing specifications provided by through the **set_scan_suppress_toggling** command.

EXAMPLES

The following example shows how to issue the **remove_scan_suppress_toggling** command:

```
prompt> remove_scan_suppress_toggling
```

SEE ALSO

```
insert_dft(2)  
report_scan_suppress_toggling(2)  
set_scan_suppress_toggling(2)
```

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_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

```
compile(2)  
set_target_library_subset(2)  
report_target_library_subset(2)  
check_target_library_subset(2)  
set_operating_conditions(2)  
target_library(3)
```

`remove_target_library_subset`
852

remove_test_mode

Removes the mode declared by the **define_test_mode** command.

SYNTAX

```
int remove_test_mode  
test_mode_label
```

ARGUMENTS

```
test_mode_label  
    Specifies the mode to remove.
```

DESCRIPTION

Removes the mode declared by the **define_test_mode** command. *This command only affects the mode and model information. It does not change the implementation of the design. In particular, for existing modes, it does not remove DFT logic. It simply removes the tool's knowledge about this logic.*

EXAMPLES

The following example removes the spec test mode named my_test_model.

```
prompt> remove_test_mode my_test_model
```

SEE ALSO

```
define_test_mode(2)  
current_test_mode(2)
```

remove_test_model

Permanently removes the test model associated with a design.

SYNTAX

```
int remove_test_model  
[-design design_name]
```

ARGUMENTS

-design *design_name*
Specifies the design from which to remove the test model. Substitute the name of the design you want for *design_name*. The default is *current_design*.

DESCRIPTION

Permanently removes the test model associated with a design. If no design name is specified, the test model for the current design is removed.

SEE ALSO

```
read_test_model(2)  
list_test_models(2)
```

remove_test_point_element

Removes the test point element specification for a particular test point type and a list of pin objects for the current design.

SYNTAX

```
int remove_test_point_element  
list_of_design_pin_objects  
[-  
type control_0 | control_z0 | control_1 | control_z1 | control_01 | control_z01 | f  
orce_0 | force_z0 | force_1 | force_z1 | force_01 | force_z01 | observe]
```

Data Types

list_of_design_pin_objects list

ARGUMENT

list_of_design_pin_objects
Specifies the list of design pin objects to which the
remove_test_point_element command applies.

-type control_0 | control_z0 | control_1 | control_z1 | control_01 | control_z01 |
force_0 | force_z0 | force_1 | force_z1 | force_01 | force_z01 | observe
Indicates which test point type the remove_test_point_element command applies
to.

DESCRIPTION

Use the **remove_test_point_element** command to remove a user-defined test point specification from the design for a particular test point type and a list of design pin objects.

EXAMPLES

The following example shows how to remove the user-defined test point specification for the type observe and for the pin U0/Q and U1/Q

```
prompt> remove_test_point_element [list U0/Q U1/Q] -type observe
```

SEE ALSO

```
set_test_point_element(2)  
report_test_point_element(2)  
preview_dft(2)  
insert_dft(2)
```

remove_test_protocol

Removes a test protocol from memory for the current design.

SYNTAX

```
int remove_test_protocol  
[-design design_name]  
[-test_mode test_mode_name]
```

Data Types

<i>design_name</i>	string
<i>test_mode_name</i>	string

ARGUMENTS

-design *design_name*
Removes the protocol for the specified design name. When this option is not specified, the command removes the protocol from the current design.

-test_mode *test_mode_name*
Removes the test protocol from the specified test mode. When this option is not specified, the current test mode is used.

DESCRIPTION

The **remove_test_protocol** command removes a test protocol. When **-test_mode** is used, the command removes the protocol for the specified mode. Without this option, the command removes the protocol for the current mode. When **-design** is used, the command removes the protocol from the specified design. Without this option, the command removes the protocol from the current design.

EXAMPLES

The following example removes the test protocol from the current design named *foo*:

```
prompt> remove_test_protocol  
Removing test protocol from current design 'foo'...
```

SEE ALSO

`create_test_protocol(2)`
`write_test_protocol(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_upf

Removes the UPF constraints from the design. This command is only supported in dc_shell.

SYNTAX

```
status remove_upf
```

ARGUMENTS

The **remove_upf** command has no arguments.

DESCRIPTION

The **remove_upf** command is used to clean up UPF data before a compile command is executed for the design.

EXAMPLES

The following example shows a typical use of the **remove_upf** command:

Loading the design RTL and libraries

```
prompt> load_upf upf_file_1
```

```
prompt> remove_upf
```

```
prompt> load_upf upf_file_2
```

```
prompt> compile_ultra
```

SEE ALSO

`compile_ultra(2)`

`load_upf(2)`

remove_user_attribute

Removes a user-specified attribute from a design or library object.

SYNTAX

```
list remove_user_attribute  
object_list  
attribute_name  
[-bus]  
[-quiet]
```

Data Types

<i>object_list</i>	list
<i>attribute_name</i>	string

ARGUMENTS

object_list
Specifies a list of design or library objects from which to remove the attribute.

attribute_name
Specifies the name of the attribute to remove.

-bus
Removes the attribute from the bus as a whole, and not from each individual bus member.

-quiet
Turns off the warning message that is issued if the attribute or objects are not found.

DESCRIPTION

This command removes the user-specified attribute from the specified objects. For a complete list of attributes, see the **attributes** man page.

The command returns the list of objects from which the attribute is removed. If an empty string is returned, no object was removed.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command removes the **output_not_used** attribute on the OUT1 port:

```
remove_user_attribute  
860
```



```
prompt> remove_user_attribute [get_ports OUT1] output_not_used  
{OUT1}
```

SEE ALSO

```
set_attribute(2)  
set_user_attribute(2)
```

remove_verification_priority

Removes the **verification_priority** attribute from the specified objects.

SYNTAX

```
status remove_verification_priority  
[-all]  
object_list
```

Data Types

object_list list

ARGUMENTS

-all
Removes the **verification_priority** attribute from all of the designs and instances in the designs.

object_list
Specifies a list of objects from which the attribute is to be removed.

DESCRIPTION

This command removes the **verification_priority** attribute from the specified objects.

SEE ALSO

```
get_attribute(2)  
list_attributes(2)  
reset_design(2)  
set_verification_priority(2)
```

remove_voltage_area

Removes voltage areas from the current design.

SYNTAX

```
int remove_voltage_area  
-all | patterns  
[-name list]
```

Data Types

patterns *list*

ARGUMENTS

-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)  
report_voltage_area(2)
```

remove_wire_load_min_block_size

Removes the **wire_load_min_block_size** attribute from the current design.

SYNTAX

```
status remove_wire_load_min_block_size
```

ARGUMENTS

The **remove_wire_load_min_block_size** command has no arguments.

DESCRIPTION

This command removes the **wire_load_min_block_size** attribute from the current design. To set the wire load minimum block size, use the **set_wire_load_min_block_size** command, with the wire load mode enclosed in single quotes.

Use the **remove_wire_load_model** command to remove **wire_load_min_block_size** attributes.

EXAMPLES

The following example removes the **wire_load_min_block_size** attribute from the design TOP:

```
prompt> current_design TOP
```

```
prompt> remove_wire_load_min_block_size
```

SEE ALSO

```
current_design(2)  
remove_wire_load_model(2)  
remove_wire_load_selection_group(2)  
reset_design(2)  
set_load(2)  
set_local_link_library(2)  
set_wire_load_min_block_size(2)  
auto_wire_load_selection(3)  
link_library(3)
```

remove_wire_load_model

Removes wire load model attributes from designs, ports, hierarchical cells, or the specified cluster of the current design.

SYNTAX

```
int remove_wire_load_model  
[-min] [-max]  
[object_list]
```

Data Types

object_list list

SYNTAX

```
int remove_wire_load_model  
[-min] [-max]  
[object_list]
```

ARGUMENTS

-min

Removes the wire load model or selection group for minimum delay analysis only. If **-max** is specified, then both min and max delay wire load models are removed.

-max

Removes the wire load model or selection group for maximum delay analysis only. Any model set for minimum delay analysis is not affected.

object_list

Specifies a list of designs, ports, and cells that are to have their wire load model attributes removed. If you use this option, the -cluster option is ignored. If neither -cluster nor **object_list** is specified, the wire load model of current design is removed.

DESCRIPTION

Removes wire load model attributes from designs, ports, and hierarchical cells of current design (all specified in the *object_list*), or from the specified cluster of current design. If neither cluster nor *object_list* is specified, the wire load model of the current design is removed. If *object_list* is specified, the -cluster option is ignored.

You can also use the **reset_design** command to remove wire load model attributes; however, using the **remove_wire_load_model** command is recommended.

EXAMPLES

The following example removes wire load model attributes from the design TOP.

```
prompt> current_design TOP  
prompt> remove_wire_load_model
```

The following example removes the minimum wire load model from the design list DESIGN_LIST.

```
prompt> remove_wire_load_model -min DESIGN_LIST
```

SEE ALSO

```
characterize(2)  
current_design(2)  
report_lib(2)  
reset_design(2)  
set_load(2)  
set_local_link_library(2)  
set_wire_load_model(2)  
set_wire_load_mode(2)  
set_wire_load_min_block_size(2)  
set_wire_load_selection_group(2)  
auto_wire_load_selection(3)  
link_library(3)
```

remove_wire_load_selection_group

Removes wire load model selection group from designs and cells, or from a specified cluster of the current design.

SYNTAX

```
int remove_wire_load_selection_group  
[-min] [-max]  
[object_list]
```

Data Types

object_list list

SYNTAX

```
int remove_wire_load_selection_group  
[-min] [-max]  
[object_list]
```

ARGUMENTS

-min

Removes the selection group from designs, cells, or a cluster for minimum delay analysis only. Specifying neither or both **-max** and **-min** options will remove both max and min wire load selection groups.

-max

Removes the selection group from designs, cells or a cluster for maximum delay analysis only. Specifying neither or both **-max** and **-min** options will remove both max and min wire load selection groups.

object_list

Removes the min or max selection group from the specified designs and cells in list. If you use this option, the -cluster option is ignored. If neither "-cluster" nor *object_list* is specified, the selection group of current design is removed.

DESCRIPTION

This command removes the **wire_load_model_selection_group** and **wire_load_model_selection_group_lib** attributes from designs and cells, or from a specified cluster. If none of designs, cells, or cluster is specified, this command removes the attributes of current design.

The removal of selection group can be for min delay analysis, max delay analysis, or both of them. This command removes wire load selection group for min delay and max delay analysis if neither -min nor -max is specified.

To verify the removal of selection group associated with a specific cluster, use the

report_clusters command. For a more detailed report on wire load selection groups associated with designs, cells, or a cluster, use the **report_wire_load** command.

EXAMPLES

The following example removes the selection group for the current design.

```
prompt> current_design TOP  
prompt> remove_wire_load_selection_group 2layermetal
```

The following example removes the min selection group for cluster 'env2'.

```
prompt> current_design TOP  
prompt> remove_wire_load_selection_group -min -cluster env2
```

The following example removes the selection group from a sub-design LOW and a hierarchical cell U1/U2 in the top design.

```
prompt> current_design TOP  
prompt> remove_wire_load_selection_group {LOW U1/U2}
```

SEE ALSO

```
characterize(2)  
current_design(2)  
set_load(2)  
set_wire_load_selection_group(2)  
report_lib(2)  
reset_design(2)  
set_local_link_library(2)  
remove_wire_load_min_block_size(2)  
remove_wire_load_model(2)  
link_library(3)
```


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_design

Renames design in memory, or moves a list of designs to a file.

SYNTAX

```
status rename_design  
design_list  
[target_name]  
[-prefix prefix_name]  
[-postfix postfix_name]  
[-update_links]
```

Data Types

<i>design_list</i>	list
<i>target_name</i>	string
<i>prefix_name</i>	string
<i>postfix_name</i>	string

ARGUMENTS

design_list
Specifies a list of designs to be renamed.

target_name
Specifies the new name of the design. If *design_list* has only one design, *target_name* will be the new design name; if *design_list* has multiple designs, *target_name* has to be a file name and all the designs in *design_list* are moved to the specified file with their base design names. One of the arguments *target_name*, *-prefix* and *-postfix* are required but they are mutually exclusive.

DESCRIPTION

In the first form, renames a design in memory. The simplest use of this command is to assign a new name to a singledesign.

Within a file containing the design, every design name must be unique. An error occurs if the name of the new design you use in the command already exists in the file.

In the second form, **rename_design** moves the specified list of designs to a file that can accept multiple designs.

To save these files to disk, use the **write** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The example renames the design "TEST" to "BILL":

```
prompt> rename_design TEST BILL
Renaming design 'TEST' to 'BILL'
1
```

In this example, the **get_designs** command is used with the **rename_design** command to move all the designs currently read into memory to the same file "all.ddc".

```
prompt> list_designs -show_file
```

Design	File	Path
-----	-----	-----
T2	T2.ddc	/usr/project
AD	AD.ddc	/test
HA	HA.ddc	/test

```
prompt> rename_design [get_designs *] /usr/project/all.ddc
Renaming design 'T2' to '/usr/project/all.ddc:T2'
Renaming design 'AD' to '/usr/project/all.ddc:AD'
Renaming design 'HA' to '/usr/project/all.ddc:HA'
1
```

```
prompt> list_designs -show_file
```

Design	File	Path
-----	-----	-----
T2	all.ddc	/usr/project
AD	all.ddc	/usr/project
HA	all.ddc	/usr/project

1

A file must be specified as the second argument of the **rename_design** command, as the error example that follows indicates.

```
prompt> rename_design {A B C} all.ddc:TEST3
Error: Cannot copy multiple designs to a single design
0
```

The **-prefix**, **-postfix**, and **-update_links** options provide convenient and efficient way to rename designs and change cell links to the new reference designs. For instance, the following script prepends the string "NEW_" to the name of design D, and changes links for its instance cells:

```
prompt> get_cells -hier -filter "ref_name == D"
{b_in_a/c_in_b/d1_in_c b_in_a/c_in_b/d2_in_c}
prompt> rename_design D -prefix NEW_ -update_links
Information: Renaming design /test_dir/D.ddc:D to /test_dir/
D.ddc:NEW_D. (UIMG-45)
prompt> get_cells -hier -filter "ref_name == D"      # no such cells!
```

```
prompt> get_cells -hier -filter "ref_name == NEW_D"  
{b_in_a/c_in_b/d1_in_c b_in_a/c_in_b/d2_in_c}  
1
```

SEE ALSO

```
copy_design(2)  
remove_design(2)  
change_link(2)  
current_design(3)
```

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_clock_gates

Replaces manually inserted clock gates with Power Compiler clock gates.

SYNTAX

```
int replace_clock_gates  
[-global]  
[-no_hier]
```

ARGUMENTS

-global
Performs hierarchical clock gate replacement in all subdesigns as one global step.

-no_hier
Limits clock gate replacement to the top level of the current design.

DESCRIPTION

This command performs clock gating replacement of manually inserted clock gates with Power Compiler clock gates. Before using this command, you can optionally use the **set_clock_gating_style** command to specify the structure of the clock-gating circuitry to be inserted. If no style is explicitly set, a default clock gating style is applied -- see the man page of the **set_clock_gating_style** command for more details about the default clock gating style.

Manually inserted clock gates can introduce glitches on the clock net, which can cause incorrect operation after synthesis. By replacing the manually inserted clock gates with Power Compiler clock gates, correct operation of the synthesized circuit is ensured. Furthermore, the clock gates are properly recognized by Power Compiler. As such, the gate-level commands such as **rewire_clock_gating** and **remove_clock_gating** are available to optimize the clock-gated netlist. During physical synthesis the clock gates are grouped together with their driven modules and registers.

Replacement of manually inserted clock gates with Power Compiler clock gates is especially useful when you want to clock gate an entire module. In many cases such module-level clock gates cannot be inserted automatically. The **replace_clock_gates** command is recommended for creating module-level clock gating.

Before clock gating replacement can be performed, the clock ports must be identified with the **create_clock** command. All combinational cells that are not buffers or inverters will be considered for clock gating, unless they are excluded with the **set_module_clock_gates** command. Clock gate replacement can only be applied to combinational cells that perform simple AND/NOR or OR/NAND functionality on the clock net. Also, the clock must drive registers or modules that are triggered by the same clock edge. The clock gating style should be compatible with the functionality of the combinational cell and the edge type of the gated registers.

For example, with a latch-based clock gating style, only cells with AND/NOR functionality driving positive-edge registers, and cells with OR/NAND functionality

driving negative-edge registers will be replaced. If a clock net drives both positive- and negative-edge registers, or if the clock net is fed to an output port, the manually inserted clock gates for that clock net will not be replaced. Black-box modules and modules with clock ports that drive special sequential cells (such as memory cells) can be module level clock gated by specifying the edge type with the **set_replace_clock_gates** command. Furthermore, if a latch-free clock gating style is selected, the enable inputs to the manually inserted clock gate should be driven by registers that are clocked by a clock signal that is compatible with the clock input to the clock gate.

When the **-no_hier** option is omitted, clock gate replacement is performed on all subdesigns. Every subdesign is processed independently. Otherwise, only the top level of the current design is processed.

To control the names of modules, cells and gated clock nets created by this command, you could set different naming style variables listed at the end.

EXAMPLES

The commands in the following example show the typical flow for using the **replace_clock_gates** command. After reading or elaborating the design and defining the clock ports, you set the clock-gating style to specify how to perform clock gate replacement. The **replace_clock_gates** command replaces the manually inserted clock gates.

```
prompt> read_verilog design.v
prompt> current_design top
prompt> link
prompt> set_clock_gating_style -sequential latch
prompt> create_clock clk
prompt> replace_clock_gates
```

SEE ALSO

```
compile(2)
create_clock(2)
set_clock_gating_style(2)
power_cg_module_naming_style(3)
power_cg_cell_naming_style(3)
power_cg_gated_clock_net_naming_style(3)
```

replace_synthetic

Implements all synthetic library parts of a design using generic logic.

SYNTAX

```
int replace_synthetic  
[-ungroup]
```

ARGUMENTS

-ungroup
Ungroups all implemented synthetic library parts into their containing designs. If **-ungroup** is not specified, synthetic library parts are implemented as a level of hierarchy.

DESCRIPTION

Use of command **replace_synthetic** is not recommended. This command will be obsolete in a future release. Processes all synthetic library parts in a hierarchical design. The processing is a subset of the synthetic library part processing performed by **compile**. A simple area-based resource sharing step is called on the design, followed by implementing all parts with minimum area implementations. To disable resource sharing, set **hdlin_resource_allocation = none**. Alternatively, you can call **set_resource_allocation none** for the design.

Several high-level optimizations during **compile** (including timing-driven resource sharing) depend on synthetic library parts. Using **replace_synthetic** before **compile** disables these optimizations.

Use **replace_synthetic** to interface to tools that do not recognize unimplemented, synthetic library parts. Use **replace_synthetic** before using FSM **extract**, or before writing a pre-**compile** description for simulation.

replace_synthetic does not process references that are marked **dont_touch**.

EXAMPLES

The following example implements synthetic parts in the current design.

```
prompt> replace_synthetic
```

SEE ALSO

```
compile(2)  
hlo_resource_allocation(3)  
synthetic_library(3)
```


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 -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_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

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

Displays attributes and their values associated with a cell, net, pin, port, instance, or design.

SYNTAX

```
string report_attribute  
[-net]  
[-cell]  
[-design]  
[-reference]  
[-hierarchy]  
[-instance]  
[-pin]  
[-port]  
[-nosplit]  
[object_list]
```

Data Types

object_list list

ARGUMENTS

- net
Displays attributes of nets in the current instance or current design. The output is sorted alphanumerically by net name.
- cell
Displays attributes of cells in the current instance or current design. The output is sorted alphanumerically by cell name.
- design
Displays attributes of the current instance or current design.
- reference
Displays attributes of references in the current design.
- hierarchy
Reports attributes for objects in the hierarchy tree.
- instance
Display attributes of instances in the current instance or current design.
- pin
Displays attributes of pins in the current instance or current design.
- port
Displays attributes of ports in the current instance or current design.
- 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.

`object_list`
Specifies a list of objects to report.

DESCRIPTION

Displays attributes and their values associated with an object. An object can be a cell, net, pin, port, instance, or a design. The `object_list` option takes precedence over all other options. For example, **report_attribute** "U1" -hierarchy -cell is the same as **report_attribute** "U1". Instance-specific attributes are reported only at the top level of the hierarchy. If an object has no attributes, no attributes are reported. Only attributes specified directly on the design object are reported. Inherited attributes are not reported.

If you set the `current_instance`, 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 displays the default attribute report for the current design:

```
prompt> report_attribute
```

```
*****
Report : Attributes
Design : test
Version: 3.0
Date   : Fri Aug 30 16:40:49 1991
*****
```

Design	Object	Type	Attribute Name	Value
test	U1	cell	boundary_optimization	true
test	U1	cell	test_dont_fault	1
test	U1	cell	dont_touch	true
test	test	design	structure	true
test	test	design	flatten	true
test	test	design	flatten_effort	1
test	cnt1	net	net_capacitance	1.000000
test	d	net	net_capacitance	0.000000
test	in	net	net_capacitance	1.000000
test	U1/A	pin	probe_point	true
test	U1/A	pin	dont_touch_network	true
test	U1/Z	pin	dont_touch_network	true
test	cnt1	port	load	7.000000

test	d	port	load	9.000000
test	in	port	max_fanout	2.000000
test	in	port	max_transition	4.000000

The following example reports attributes for cells in the current design:

prompt> **report_attribute -cell**

```
*****
Report : Attributes
Design : test
Version: 3.0
Date   : Fri Aug 30 16:40:51 1991
*****
```

Design	Object	Type	Attribute Name	Value
test	U1	cell	boundary_optimization	true
test	U1	cell	test_dont_fault	1
test	U1	cell	dont_touch	true

The following example reports attributes for the design named test:

prompt> **report_attribute -design**

```
*****
Report : Attributes
Design : test
Version: 3.0
Date   : Fri Aug 30 16:40:51 1991
*****
```

Design	Object	Type	Attribute Name	Value
test	test	design	structure	true
test	test	design	flatten	true
test	test	design	flatten_effort	1

The following example reports attributes for nets in the current design:

prompt> **report_attribute -net**

```
*****
Report : Attributes
Design : test
Version: 3.0
Date   : Thu Sep  5 17:13:27 1991
*****
```

Design	Object	Type	Attribute Name	Value
test	cnt1	net	net_capacitance	1.000000
test	d	net	net_capacitance	0.000000
test	in	net	net_capacitance	1.000000

The following example reports attributes for ports in the current design:

```
prompt> report_attribute -port
```

```
*****
Report : Attributes
Design : test
Version: 3.0
Date   : Thu Sep  5 17:13:27 1991
*****
```

Design	Object	Type	Attribute Name	Value
test	cntl	port	load	7.000000
test	d	port	load	9.000000
test	in	port	max_fanout	2.000000

The following example reports attributes for pins in the current design:

```
prompt> report_attribute -pin
```

```
*****
Report : Attributes
Design : test
Version: 3.0
Date   : Thu Sep  5 17:13:28 1991
*****
```

Design	Object	Type	Attribute Name	Value
test	U1/A	pin	probe_point	true
test	U1/A	pin	dont_touch_network	true
test	U1/Z	pin	dont_touch_network	true

The following example reports attributes for instances in the current design:

```
prompt> report_attribute -instance
```

```
*****
Report : Attributes
Design : UPC5
Version: 3.0
Date   : Thu Sep  5 17:12:32 1991
*****
```

Design	Object	Type	Attribute Name	Value
UPC5	H1/U9/B	pin	test_dont_fault	2
UPC5	H1/U50	cell	test_dont_fault	1

The following example reports attributes for subdesigns:

```
prompt> report_attribute -hierarchy
```

```
*****
```

```

Report : Attributes
Design : TOP
Version: 3.0
Date   : Mon Sep  9 14:37:45 1991
*****

```

Design	Object	Type	Attribute Name	Value
TOP	TOP	design	default_latch_type_exact	LD3
TOP	TOP	design	flatten	true
TOP	MINUS1	net	dont_touch	true
ADDER	U1	cell	boundary_optimization	true
ADDER	ADDER	design	default_latch_type_exact	LD3
ADDER	ADDER	design	default_flip_flop_type	Ia1.0(Q)
ADDER	CARRY	net	dont_touch	true
ADDER	U1	cell	boundary_optimization	true
ADDER	ADDER	design	default_latch_type_exact	LD3
ADDER	ADDER	design	default_flip_flop_type	Ia1.0(Q)
ADDER	CARRY	net	dont_touch	true
ADDER	U3/A	pin	dont_touch_network	true
FULL_SUBTRACTOR	U2	cell	boundary_optimization	true
FULL_SUBTRACTOR	FULL_SUBTRACTOR	design	structure	true

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_auto_ungroup

Displays information about the cell hierarchies that have been ungrouped using the **compile** command with either the **-auto_ungroup area** or the **-auto_ungroup delay** option.

SYNTAX

```
int report_auto_ungroup
[-nosplit]
[-full]
```

ARGUMENTS

-nosplit

Prevents line-splitting and facilitates writing software to extract information from the report output. Most of the auto ungroup information appears 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 in the report. By default, if there is a submodule in multiple locations in a hierarchy, the submodule's components are listed only once, with an ellipsis (...) indicating the contents of a previously displayed module.

DESCRIPTION

This command displays the cell instances that are ungrouped using the **compile** command with either the **-auto_ungroup area** or the **-auto_ungroup delay** option. The auto ungroup information is stored on the current design. To produce a report of the hierarchies that are ungrouped during the compile process, set the current design to be the same as the compiled design.

EXAMPLES

The following is an example of an auto ungroup report of the current design:

```
prompt> report_auto_ungroup

*****
Report : Auto Ungrouping
Design : hier
Version: 2001.08
Date   : Mon Oct  9 16:44:38 2000
*****

=====
| Instance | | |
| Name     | Cell Name | Num. Insts |
|-----|-----|-----|
=====
```

	H1		sub1		6	
	H2		sub2		6	
=====						

1

SEE ALSO

`compile(2)`

report_autofix_configuration

Displays options set by the set_autofix_configuration command.

SYNTAX

```
integer report_autofix_configuration  
[-type all | clock | set | reset | xpropagation | internal_bus | external_bus]
```

ARGUMENT

-type all | clock | set | reset | xpropagation | internal_bus | external_bus
Indicates which type the **report_autofix_configuration** command applies to.
"-type all" displays all configurations. See the example below.

DESCRIPTION

The **report_autofix_configuration** command displays options set by the **set_autofix_configuration** command on the current design. Use **type -all** to display all configurations.

EXAMPLES

The following is an example of the **report_autofix_configuration** command:

```
prompt> report_autofix_configuration -type all  
  
*****  
Report : Autofix configuration  
Design : my_current_design  
Version: A-2007.12  
Date   : Fri Dec 28 05:38:07 2007  
*****  
  
=====
```

TEST MODE:	all_dft
VIEW	: Specification

```
=====
```

Fix type:	
FixBidirectional:	
Fix method:	Input
Fix type:	Set
Fix method:	Mux
Fix latches:	Disable
Fix type:	Reset
Fix method:	Mux
Fix latches:	Disable
Fix type:	Clock
Control signal:	clock_autofix_mode (TestMode)

TestData signal:	clock_autofix_clock_s (TestData ScanMaste
rClock)	
Fix method:	Mux
Fix latches:	Disable
Fix clocks used as data:	Disable

SEE ALSO

[reset_autofix_configuration\(2\)](#)
[set_autofix_configuration\(2\)](#)

report_autofix_element

Displays options set by the set_autofix_element command.

SYNTAX

```
status report_autofix_element  
[-type all | clock | set | reset | xpropagation | internal_bus | external_bus]  
[list_of_design_objects]
```

ARGUMENTS

-type all | clock | set | reset | xpropagation | internal_bus | external_bus
Indicates which type the **report_autofix_element** command applies to.

list_of_design_objects
Lists the design objects for which to report the autofix elements.

DESCRIPTION

The **report_autofix_element** command displays options set by the **set_autofix_element** command on the current design. The report is generated for all designs below the current design. If a *list_of_design_objects* is specified, the report is generated only for the design(s) specified in the list. Use **-type all** to display all elements.

EXAMPLES

The following is an example of the **report_autofix_element** command:

```
prompt> report_autofix_element  
  
*****  
Report : Autofix element  
Design : my_current_design  
Version: A-2007.12  
Date   : Fri May 28 11:00:29 2007  
*****  
  
=====
```

TEST MODE:	all_dft
VIEW	: Specification

```
=====
```

Designs/Instances:	u1 u3
Fix type:	Set
Control signal:	TMC (TestMode)
TestData signal:	SET (Reset)
Fix method:	Gate
Fix latches:	Enable

Designs/Instances:	a0 a1
Fix type:	Internal_bus
Control signal:	TMC (TestMode)

Fix method:

Disable_all

1

SEE ALSO

`reset_autofix_element(2)`

`set_autofix_element(2)`

`report_autofix_element`

report_boundary_cell

Reports boundary cell configuration specified for the current design.

SYNTAX

```
status report_boundary_cell
```

ARGUMENTS

The **report_boundary_cell** command has no arguments.

DESCRIPTION

This command reports boundary cell configuration specified for various ports or core cells of the current design.

The following format is used to report the boundary cell configuration:

Boundary Cell Structures	Status
-----	-----
Class:	<class_name>
Function:	<function_type>
Ports:	<port_name>
Cells:	<core_list>
Type:	<cell_type>
Design Name:	<design_name>
Register IO Implementation:	<impl_name>
Use Dedicated Wrapper Clock:	<boolean>
Safe State:	<state>
...	

EXAMPLES

The following example reports the boundary cell configuration for the core wrapper class on port1 and port2:

```
prompt> set_boundary_cell -class core_wrapper -type WC_D1 -safe_state none \  
        -port_list port1  
1  
  
prompt> set_boundary_cell -class core_wrapper -design my_des1 -safe_state 0 \  
        -function input -port_list port2 -register in_place -use_dedicated TRUE  
1  
  
prompt> report_boundary_cell  
  
*****  
Report : Boundary Cell  
Design : M1  
Version: W-2004.12
```

Date : Thu Sep 16 09:46:38 2004

Boundary Cell Structures	Status
-----	-----
Class:	Core Wrapper
Function:	
Ports:	port1
Cells:	
Type:	WC_D1
Design Name:	
Register IO Implementation:	
Use Dedicated Wrapper Clock:	
Safe State:	none
Class:	Core Wrapper
Function:	Input
Ports:	port2
Cells:	
Type:	
Design Name:	my_des1
Register IO Implementation:	In Place
Use Dedicated Wrapper Clock:	True
Safe State:	0

1

SEE ALSO

insert_dft(2)
preview_dft(2)
set_boundary_cell(2)
set_bsd_configuration(2)
set_wrapper_configuration(2)

report_boundary_cell_io

Displays options set by the set_boundary_cell_io command.

SYNTAX

status **report_boundary_cell_io**

ARGUMENTS

The **report_boundary_cell_io** command has no arguments.

DESCRIPTION

The **report_boundary_cell_io** command displays options set by the **set_boundary_cell_io** command on the current design.

The command reports the boundary cell IO pins set by **set_boundary_cell_io** command.

EXAMPLES

The following are examples of the **report_boundary_cell_io** command:

```
prompt> report_boundary_cell_io
```

```
*****  
Report : Boundary Cell IO  
Design : what_testcase  
Version: Z-2007.03  
Date   : Fri Dec 21 08:44:55 2006  
*****
```

```
Cell Name:  what_testcase_bs_core_inst/bsr_gpio_3_inst/data_bsr_inst_U1_U1_U3  
Type:  Boundary  
OutPI:  what_testcase_bs_core_inst/what_testcase_ts_core_inst/  
what_testcase_core_inst/U21/Z  
OutPO:  what_testcase_bs_core_inst/bsr_gpio_3_inst/U11/Z
```

```
Cell Name:  what_testcase_bs_core_inst/bsr_gpio_2_inst/data_bsr_inst_U1_U1_U3  
Type:  Boundary  
InPI:  what_testcase_bs_core_inst/what_testcase_ts_core_inst/  
what_testcase_core_inst/U17/A  
InPO:  what_testcase_bs_core_inst/bsr_gpio_2_inst/U12/Z
```

```
Cell Name:  what_testcase_bs_core_inst/bsr_gpio_1_inst/data_bsr_inst_U1_U1_U3  
Type:  Boundary  
InPI:  what_testcase_bs_core_inst/what_testcase_ts_core_inst/  
what_testcase_core_inst/U16/A  
InPO:  what_testcase_bs_core_inst/bsr_gpio_1_inst/U15/Z  
OutPI:  what_testcase_bs_core_inst/what_testcase_ts_core_inst/  
what_testcase_core_inst/U24/Z
```

OutP0: what_testcase_bs_core_inst/bsr_gpio_1_inst/U22/Z

SEE ALSO

`remove_boundary_cell_io(2)`
`set_boundary_cell_io(2)`

report_bounds

Reports bounds in the design.

SYNTAX

```
int report_bounds  
-all | bound | -name name_list  
[-verbose]
```

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_bounds(2)
update_bounds(2)
set_cell_location(2)
remove_bounds(2)
```

report_bsd_compliance

Displays options set by the set_bsd_compliance command.

SYNTAX

integer **report_bsd_compliance**

ARGUMENTS

This command has no arguments.

DESCRIPTION

The **report_bsd_compliance** command displays the options set by the **set_bsd_compliance** command on the current design.

The command reports the compliance pins and the bit pattern set for them by **set_bsd_compliance**.

EXAMPLES

The example shows the compliance report generated by **report_bsd_compliance**:

```
prompt> report_bsd_compliance

*****
Report : BSD Compliance
Design : TOP
Version: Z-2007.03-H01
Date   : Fri Sep 22 08:44:55 2006
*****

  pattern-name    port-name    port-value
  -----
  p1
                my_bidi        0
                my_eni         1
```

SEE ALSO

current_test_mode(2)
report_dft_configuration(2)
report_scan_configuration(2)
report_scan_path(2)
set_bsd_compliance(2)
set_dft_configuration(2)
set_scan_configuration(2)
set_scan_path(2)

report_bsd_instruction

Displays options set by the `set_bsd_instruction` command.

SYNTAX

```
status report_bsd_instruction
[-view spec | existing_dft]
[-instruction list_of_instruction_names]
```

Data Types

list_of_instruction_names list

ARGUMENTS

`-view spec | existing_dft`
Specifies the view on which to report. Valid options are **spec** and **existing_dft**. If not specified, the tool reports both the **spec** and **existing_dft** views.

`-instruction list_of_instruction_names`
Specifies the name of an instruction or a list of instructions on which to report. If no instructions are specified, the tool reports all defined instructions.

DESCRIPTION

This command displays the options set by the **set_bsd_instruction** command on the current design.

It reports on register, opcode, capture value, input and output clock conditions, signature, internal scan pin, instruction enable pins, private, and timing of the instructions.

EXAMPLES

The following command reports on both the **spec** and the **existing_dft** view:

```
prompt> report_bsd_instruction

*****
Report : BSD Instruction
Design : TOP
Version: Z-2007.03-H01
Date   : Fri Sep 22 08:45:24 2006
*****

=====
TEST MODE: all_dft
```



```

VIEW      : Specification
=====
EXTEST
    Register      BOUNDARY
    Opcodes       0100
    Input_clock_condition  PI
    Output_condition  NONE

EXTEST_TRAIN
    Register      BOUNDARY
    Opcodes       1101
    Input_clock_condition  PI
    Output_condition  NONE
    Clock_cycles  tck      100
                   clk1    200
                   clk2    550

```

```

=====
TEST MODE: all_dft
VIEW      : Existing DFT
=====

```

```

IDCODE
    Register      DEVICE_ID
    Opcodes       0110
    Capture_value  000100000000000000001000000000011
    Input_clock_condition  PI
    Output_condition  NONE

EXTEST_PULSE
    Register      BOUNDARY
    Opcodes       1110
    Input_clock_condition  PI
    Output_condition  NONE
    Time          11.59

INITIALIZE
    Register      BOUNDARY
    Opcodes       1011
    Input_clock_condition  TCK
    Output_condition  HIGHZ
    Signature      SIGDATA
    Internal_scan   U11/in
    Inst_enable     U4/TN
                   U15/in0
                   U15/in1

    Private

```

The following command reports on the specified instructions in the **existing_dft** view:

```

prompt> report_bsd_instruction -view existing_dft \
      -instruction [list EXTEST_PULSE]

```

```
*****
Report : BSD Instruction
Design : TOP
Version: Z-2007.03-H01
Date   : Tue Sep 12 11:33:44 2006
*****
```

```
=====
TEST MODE: all_dft
VIEW      : Existing DFT
=====
EXTEST_PULSE
    Register          BOUNDARY
    Opcodes           1110
    Input_clock_condition  PI
    Output_condition    NONE
    Time               11.59
```

SEE ALSO

```
current_test_mode(2)
report_dft_configuration(2)
report_scan_configuration(2)
report_scan_path(2)
set_bsd_instruction(2)
set_dft_configuration(2)
set_scan_configuration(2)
set_scan_path(2)
```

report_bsd_linkage_port

Displays options set by the `set_bsd_linkage_port` command.

SYNTAX

integer **report_bsd_linkage_port**

ARGUMENTS

This command has no arguments.

DESCRIPTION

This command displays options set by the **set_bsd_linkage_port** command on the current design.

EXAMPLES

The example shows the report generated by **report_bsd_linkage_port**:

```
prompt> report_bsd_linkage_port

*****
Report : BSD Linkage Port
Design : TOP
Version: Z-2007.03-H01
Date   : Fri Sep 22 08:44:55 2006
*****

    my_bidi
    my_ini
    my_eni
    out1
```

SEE ALSO

```
current_test_mode(2)
report_dft_configuration(2)
report_scan_configuration(2)
report_scan_path(2)
set_bsd_linkage_port(2)
set_dft_configuration(2)
set_scan_configuration(2)
set_scan_path(2)
```

report_bsd_power_up_reset

Displays options set by the `set_bsd_power_up_reset` command.

SYNTAX

integer **report_bsd_power_up_reset**

DESCRIPTION

This command displays options set by the **set_bsd_power_up_reset** command on the current design. It also reports the power up reset pins set by **set_bsd_power_up_reset**.

EXAMPLES

The example shows the report generated by the **report_bsd_power_up_reset** command:

```
prompt> report_bsd_power_up_reset

*****
Report : BSD Power Up Reset
Design : TOP
Version: Z-2007.03-H01
Date   : Fri Sep 22 08:44:55 2006
*****

  pin-name      polarity      delay
  -----
  U4/TN         low          13
```

SEE ALSO

```
current_test_mode(2)
report_dft_configuration(2)
report_scan_configuration(2)
report_scan_path(2)
set_bsd_power_up_reset(2)
set_dft_configuration(2)
set_scan_configuration(2)
set_scan_path(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)  
balance_buffer(2)  
clean_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)
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]
```

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.

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
```

```
*****
```

```
Report : bus
```

```
Design : SUBTRACTOR
```

```
Version: v2.0
```

```
Date   : Fri Mar 16 14:22:01 1991
```

```
*****
```

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

[create_bus\(2\)](#)
[remove_bus\(2\)](#)

report_cache

Reports on the contents of synthetic library caches.

SYNTAX

```
int report_cache
[-design_lib list]
[-module list]
[-implementation list]
[-parameters parameter_list]
[-tech_lib list]
[-wire_load list]
[-operating_conditions list]
[-directory dir_list]
[-smaller size | -larger size]
[-accessed_since days
 | -accessed_beyond days]
[-netlist_only | -model_only]
[-sort_largest | -sort_oldest | -sort_cache_key]
[-statistics]
```

Data Types

<i>list</i>	<i>dir_list</i>
<i>parameter_list</i>	<i>list</i>
<i>dir_list</i>	<i>list</i>
<i>size</i>	<i>int</i>
<i>days</i>	<i>float</i>

ARGUMENTS

-design_lib *list*
Restricts the report to the design libraries in the list. Strings in the list can contain the wildcard character "*" (for example, { DW_01 DW_* *SNPS* }).

-module *list*
Restricts the report to the modules in the list. Strings in the list can contain the wildcard character "*".

-implementation *list*
Restricts the report to the implementations in the list. Strings in the list can contain the wildcard character "*".

-parameters *parameter_list*
Restricts the report to the parameters in the list.

-tech_lib *list*
Restricts the report to the technology libraries in the list. Strings in the list can contain the wildcard character "*".

-wire_load *list*
Restricts the report to the wire loads in the list. Strings in the list can

contain the wildcard character "*".

-operating_conditions *list*

Restricts the report to the operating conditions in the list. Strings in the list can contain the wildcard character "*".

-directory *dir_list*

Uses the specified list of directories, instead of those listed by the **cache_read** and **cache_write** variables, as the cache directories to report on.

-smaller *size*

Restricts the report to cache elements that are smaller than the specified number of bytes.

-larger *size*

Restricts the report to cache elements that are larger than the specified number of bytes.

-accessed_since *days*

Restricts the report to cache elements whose last access time is between now and the specified number of days ago.

-accessed_beyond *days*

Restricts the report to cache elements whose last access time is more than the specified number of days ago.

-netlist_only

Reports only on netlists.

-model_only

Reports only on models.

-sort_largest

Sorts the report by cache element size.

-sort_oldest

Sorts the report by cache element access time.

-sort_cache_key

Sorts the report by cache key. The cache key consists of the following fields: root UNIX directory, design library, module, implementation, parameters, technology library, wire load, and operating conditions. The root directory and parameters are not used in the sorting.

-statistics

Reports timing and area information of the cached models.

DESCRIPTION

The **report_cache** command reports on the cache elements in the directories listed by the **cache_read** and **cache_write** variables. If the directory *dir* is listed as a cache directory, the directory *dir/synopsys_cache_v*.** is the top of the cache directory structure.

There are two types of elements stored in the cache: netlists and models. Netlists are characterized by not being mapped to technologies and having no wire load or operating conditions. Models, however, always have wire load and operating condition settings. They contain timing models and areas of the corresponding netlists. For models, wire load and operating conditions can have the value *default*. For wire load, *default* means wire load selection was used if it is defined in the technology library or no wire load was used. For operating conditions, *default* means that *nom_temperature*, *nom_process*, and *nom_voltage* was used.

Without command arguments, all cache elements in the cache directories are listed. If there are command arguments that restrict the report, the cache element must pass each command argument specified. An element is accepted by a command argument if it matches one of the items in the command argument's list. If the **-statistics** option is specified, only cached models will be listed.

For example, the command

```
report_cache -module {DW01_add DW02_mult} -implementation {wall rpl}
```

lists the ripple adder and the Wallace Tree multiplier.

The **-parameters** option has additional features. The *parameter_list* is a space-separated list of quoted parameter specifications (for example, {"N=8,M=6" "N=3"}). Each quoted parameter specification is a comma separated list of parameter settings. Parameter settings are a parameter name followed by an equal sign followed by a parameter value. Parameter settings can also support ranges for the parameter value such as "N = [8;17)" where "[" means the endpoint is inclusive and "(" means the endpoint is exclusive. Thus, if you execute **report_cache -parameters {"N = [8;17)"}**, cache elements whose parameter N is greater than or equal to 8 and less than 17 are matched. The braces "{" and "}" are important. '-parameters "n=8, m=6"' get parsed as '-parameters {"n=8" "m=6"}' instead of '-parameters {"n=8, m=6"}', so you cannot match a cache element with 2 parameters, n and m, which are the braces in this case. The special parameter specification "no_parameters" also matches elements with no parameters. For example, {"n=8, no_parameters"} matches elements with no parameters or matches elements that have the parameter "n=8".

Matching parameter names has additional complications because cache elements have a variable number of parameters and the parameter names vary. For example, an adder has one parameter, n, whereas a multiplier has two parameters, n and m. Therefore, parameter specifications are matched using two rules. First, if the names and values of a parameter specification match the cache element parameters exactly, the element is accepted. Second, if all the cache element parameters are covered in a parameter specification but there are extra parameters in the parameter specification, the cache element is also

matched. Otherwise, the cache element is not accepted. Thus, in the previous example, you can match both the adder and multiplier by using a parameter specification that includes both the parameters n and m. Remember that leaving off the **-parameters** option will cause all parameter specifications to be accepted.

EXAMPLES

The default output format lists cache elements clustered by directory, technology library, wire load, and operating conditions. Following is an example of the default output.

```
prompt> report_cache -larger 5000
```

```
*****
```

```
Report : cache
```

```
Version: v3.1-development
```

```
Date   : Wed Mar 31 14:17:03 1993
```

```
*****
```

DESIGN	MODULE	IMPLEMENT-	PARAMETERS	ACCESS	SIZE
LIBRARY		ATION		days	bytes

```
Cache Directory Root: '/remote/rd30/df/tcache/'
```

```
Technology Library: 'and_or.db'
```

```
Wire Load: no_wire_load
```

```
Operating Conditions: no_operating_conditions
```

DW01	DW01_ADD_ABC	str		0.0	5203
DW01	DW01_absval	cla	width=3	0.0	10064
DW01	DW01_add	cla	width=5	0.0	17376
DW01	DW01_inc	cla	width=3	0.0	9136
DW01	DW01_inc	cla	width=6	0.0	10912
DW02	DW02_mult	csa	A_width=3	0.0	29440
			B_width=3		

```
Cache Directory Root: '/remote/rd30/df/tcache/'
```

```
Technology Library: 'and_or.db'
```

```
Wire Load: '-default-'
```

```
Operating Conditions: '-default-'
```

DW01	DW01_ADD_ABC	str		0.0	6162
DW01	DW01_GP_SUM	str		0.0	5302
DW01	DW01_MUX	str		0.0	5134
DW01	DW01_XOR2	str		0.0	5020
DW01	DW01_absval	cla	width=3	0.0	9768
DW01	DW01_add	cla	width=5	0.0	20982
DW01	DW01_inc	cla	width=3	0.0	10469
DW01	DW01_inc	cla	width=6	0.0	13654
DW02	DW02_mult	csa	A_width=3	0.0	31452
			B_width=3		

Queries can be substantially more complicated, as in the following example.

```
prompt> report_cache -dir ~ -design_lib DW02 \  
        -mod *mult -accessed_since 21 \  
        -param {"n=[3;100), m=[9;100]" "n=(6; 100), m= (6; 100)"}
```

For more information, see the *DesignWare Reference Manual* and the *Library Compiler Reference Manual Volume 1* and *Volume 2*.

SEE ALSO

```
create_cache(2)  
remove_cache(2)  
report_synlib(2)  
set_wire_load_mode(2)  
set_wire_load_model(2)  
set_wire_load_min_block_size(2)  
set_wire_load_selection_group(2)  
cache_read(3)  
cache_write(3)
```

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_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_compile_options(2)
report_design(2)
report_hierarchy(2)
report_net(2)
report_reference(2)
report_default_significant_digits(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

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

[apply_clock_gate_latency\(2\)](#)
[compile\(2\)](#)
[compile_ultra\(2\)](#)
[remove_clock_gating\(2\)](#)
[reset_clock_gate_latency\(2\)](#)
[rewire_clock_gating\(2\)](#)
[set_clock_gate_latency\(2\)](#)
[set_clock_gating_style\(2\)](#)

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:

- **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

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

report_compile_options

Displays information about the compile options for the design of the current instance, if set; or for the current design otherwise.

SYNTAX

```
int report_compile_options  
[-nosplit]
```

ARGUMENTS

-nosplit

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. This option prevents line-splitting and facilitates writing software to extract information from the report output.

DESCRIPTION

Displays the compile directives of the current instance or current design and all of the designs in the hierarchy. The compile directives determine how a design is optimized.

If you set **current_instance**, the report is generated for the design of that instance; otherwise the report is generated for the current design.

EXAMPLES

The following is an example of a compile-options report for the design "AN2_DESIGN".

```
prompt> report_compile_options
```

```
*****  
Report : compile_options  
Design : AN2_DESIGN  
Version: 1998.02  
Date   : Fri Dec 29 14:24:08 1997  
*****
```

Design	Compile Option	Value
AN2_DESIGN	flatten	false
	structure	true
	structure_boolean	false
	structure_timing	true

In the following example, compile options are set for the design "encoder", and a report is generated.

```
prompt> set_structure false
```

```
prompt> set_flatten true
```

```
prompt> report_compile_options
```

```
*****
Report : compile_options
Design : encoder
Version: 1998.02
Date   : Fri Dec 29 14:24:08 1997
*****
```

Design	Compile Option	Value
encoder	flatten	true
	flatten_effort	low
	flatten_minimize	single
	flatten_phase	false
	structure	false
	structure_boolean	false
	structure_timing	false

In the following example, cost priority and multiple port nets options are set for the design "encoder", and a report generated:

```
prompt> set_cost_priority -delay
```

```
prompt> set_fix_multiple_port_nets -feedthroughs -outputs
```

```
prompt> report_compile_options
```

```
*****
Report : compile_options
Design : encoder
Version: 1998.02
Date   : Fri Dec 29 14:24:08 1997
*****
```

Design	Compile Option	Value
encoder	flatten	true
	flatten_effort	low
	flatten_minimize	single
	flatten_phase	false
	structure	false
	cost_priority	max_delay
	fix_multiple_port_nets	feedthroughs
		outputs

SEE ALSO

compile(2)

report_compile_options

```
set_flatten(2)
set_structure(2)
set_cost_priority(2)
set_fix_multiple_port_nets(2)
```

report_congestion

Reports the congestion statistics.

SYNTAX

status **report_congestion**

ARGUMENTS

The **report_congestion** command has no arguments.

DESCRIPTION

Congestion optimization and reporting is available with Design Compiler Graphical.

This command computes and displays congestion statistics for the current design.

If you call this command two or more consecutive times without any changes to the design, global routing is skipped for the second time and the report is based on the global routing done in the first 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.

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.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example runs the **report_congestion** command.

```
prompt> report_congestion
```

```
*****  
Report : congestion  
Design : test_design  
Version: A-2007.12  
Date   : Mon Oct 29 10:43:51 2007  
*****
```

```
Both Dirs: Overflow = 5 Max = 1 (5 GRCs) GRCs = 5 (11.90%)
```

```
report_congestion  
968
```

H routing: Overflow = 5 Max = 1 (5 GRCs) GRCs = 5 (11.90%)
V routing: Overflow = 0 Max = 0 (0 GRCs) GRCs = 0 (0.00%)

SEE ALSO

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

`report_constraint`

- 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.

Creation and usage of scenarios is available with Design Compiler Graphical.

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)

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)
compile_fix_cell_degradation(3)

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_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

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. This command is supported only in topographical mode.

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:

- 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

The option is disabled in Design Compiler.

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_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

```

analyze(2)
define_design_lib(2)
elaborate(2)
read_file(2)

```


report_dft_configuration

Displays the options specified by the **set_dft_configuration** command.

SYNTAX

integer **report_dft_configuration**

ARGUMENTS

This command has no arguments.

DESCRIPTION

This command displays options specified by the **set_dft_configuration** command on the current design.

The command shows which DFT structures are enabled or disabled.

EXAMPLES

The following is an example of the report generated by the **report_dft_configuration** command:

```
prompt> report_dft_configuration

*****
Report : DFT configuration
Design : top
Version: B-2008.09
Date   : Thu Apr  3 16:58:58 2008
*****

DFT Structures                               Status
-----
Scan:                                         Enable
Fix Sets:                                    Disable
Fix Resets:                                  Disable
Fix Clocks:                                  Disable
Fix Busses:                                  Enable
Fix Bdirectional Ports:                     Enable
Fix X Propagation:                           Disable
Control Points:                             Disable
Observe Points:                             Disable
Logic BIST:                                  Disable
Wrapper:                                     Disable
Boundary scan:                               Disable
Scan Compression:                           Disable
Pipeline Scan Data:                         Disable
Clock Controller:                           Disable
ConnectClockGating:                         Enable
```

Mode Decoding Style:

Binary

SEE ALSO

```
report_dft_configuration(2)
report_dft_signal(2)
report_scan_configuration(2)
report_scan_path(2)
set_dft_configuration(2)
set_dft_signal(2)
set_scan_configuration(2)
set_scan_path(2)
```

report_dft_connect

Reports the existing DFT connectivity specifications.

SYNTAX

```
status report_dft_connect
```

ARGUMENTS

The **report_dft_connect** command has no arguments.

DESCRIPTION

The **report_dft_connect** command reports the current DFT connectivity associations. The command reports associations specified by the **set_dft_connect** command.

EXAMPLES

```
prompt> report_dft_connect
```

SEE ALSO

```
remove_dft_connect(2)  
set_dft_connect(2)  
set_dft_signal(2)
```

report_dft_design

Reports all user-specified DFT designs.

SYNTAX

```
int report_dft_design  
[-all]  
[-type design_type_name]  
[-design_name design_name]
```

DESCRIPTION

The **report_dft_design** command reports all user-specified DFT designs in the following format.

Design Name	Design Type
<name1>	<type1>

EXAMPLE

The following example reports all user-specified DFT designs.

```
prompt> define_dft_design -design_name my_des \ -type WC_D1 -interface  
{shift_clk capture_clk h \ capture_en capture_en h shift_en shift_dr h \ cti  
si h cto so h cfi data_in h cfo data_out h} 1 prompt> define_dft_design -  
design my_des1 -type BC4 \ -interface {capture_clk cap_clk h capture_en cen  
h \ shift_dr shift h si ti h so to h data_in di h data_out do h} 1 prompt>  
define_dft_design -design my_des2 -type BC4 \ -interface {capture_clk cap_clk  
h capture_en cen h \ shift_dr shift h si ti h so to h data_in di h data_out  
do h} 1 prompt> report_dft_design  
Design Name      Design Type  
-----  
my_des           WC_D1  
my_des1          BC4  
my_des2          BC4  
-----
```

SEE ALSO

```
insert_dft(2)  
preview_dft(2)  
define_dft_design(2)  
remove_dft_design(2)
```

report_dft_drc_rules

Reports the severity of the DRC violations for the cells in the design.

SYNTAX

```
integer report_dft_drc_rules  
[-all]  
[-violation drc_error_ID]  
[-cell cell_list]
```

Data Types

<i>drc_error_ID</i>	string
<i>cell_list</i>	string

ARGUMENTS

-all
Reports all of the cells for which the violation severity has been changed.

-violation *drc_error_ID*
Reports the cells for which the severity of the specified violation(s) has been changed.

-cell *cell_list*
Specifies the cell(s) for which the violation severity is to be reported.

DESCRIPTION

The **report_dft_drc_rules** command is used to check the violation severity status for the cells in the design.

Use the **set_dft_drc_rules** command to change the violation severity. To reset the severity of the violation to the default value, use the **reset_dft_drc_rules** command.

EXAMPLES

The following example changes the severity of DRC violation TEST-504 for all cells to *warning*, TEST-505 for cells reg3 and reg4 to *warning*, and TEST-505 for cell reg6 to *ignore*. The corresponding **report_dft_drc_rules** command outputs are shown below:

```
prompt> set_dft_drc_rules -warning {TEST-504}  
prompt> set_dft_drc_rules -warning {TEST-505} -cell {reg3 reg4}  
prompt> set_dft_drc_rules -ignore {TEST-505} -cell {reg6}
```

```
prompt> report_dft_drc_rules
```

Severity of the following violations have been changed by the user.

TEST-504 : Constant 0 violation
TEST-505 : Constant 1 violation

The following command reports all of the cells for which the violation severity is changed:

```
prompt> report_dft_drc_rules -all
```

Violation Name	Default Severity	Specified Severity	Range/ Cell list
TEST-504	error	warning	all cells
TEST-505	error	warning	reg3
	error	warning	reg4
	error	ignore	reg6

The following command reports the cells for which the specified violation severity is changed:

```
prompt> report_dft_drc_rules -violation {TEST-504 TEST-505}
```

Violation Name	Default Severity	Specified Severity	Range/ Cell list
TEST-504	error	warning	all cells
TEST-505	error	warning	reg3
	error	warning	reg4
	error	ignore	reg6

The following command reports the severity of violations for the specified cells:

```
prompt> report_dft_drc_rules -cell {reg2 reg4}
```

Cell Name	Violation Name	Default Severity	Specified Severity
reg2	TEST-504	error	warning
reg4	TEST-504	error	warning
	TEST-505	error	warning

SEE ALSO

dft_drc(2)
insert_dft(2)
reset_dft_drc_rules(2)
set_dft_drc_rules(2)

report_dft_equivalent_signals

Reports all of the equivalent DFT signals specified with `set_dft_equivalent_signals` command.

SYNTAX

integer **report_dft_equivalent_signals**

ARGUMENTS

This command has no arguments.

DESCRIPTION

This command reports the equivalent set of signals specified using the **set_dft_equivalent_signals** command.

EXAMPLES

The following example illustrates reporting all equivalent scan signals:

```
prompt> report_dft_equivalent_signals
```

SEE ALSO

```
insert_dft(2)  
preview_dft(2)  
remove_dft_equivalent_signals(2)  
set_dft_equivalent_signals(2)  
set_dft_signal(2)
```

report_dft_insertion_configuration

Displays options set by set_dft_insertion_configuration command.

SYNTAX

```
int report_dft_insertion_configuration
```

ARGUMENTS

This command has no arguments.

DESCRIPTION

This command displays options set by set_dft_insertion_configuration command on the current design.

EXAMPLES

The following are examples of the report_dft_insertion_configuration command.

```
prompt> report_dft_insertion_configuration

*****
Report : DFT insertion configuration
Design : A
Version: 2003.12-DFT-POWER-BETA1
Date   : Wed Aug 20 17:37:20 2003
*****

Options                                Status
-----                                -
Map_effort                             Medium
Preserve_design_name                   False
Route_scan_enable                      False
Route_scan_clock                       False
Route_scan_serial                      False
Synthesis_optimization                 All
Unscan                                 False
1
```

SEE ALSO

```
set_dft_signal(2)
set_scan_configuration(2)
set_scan_path(2)
report_dft_configuration(2)
report_dft_signal(2)
report_scan_path(2)
report_scan_configuration(2)
```


report_dft_location

Reports the DFT Hierarchy location specification for the current design.

SYNTAX

```
integer report_dft_location  
[-all]
```

DESCRIPTION

The **report_dft_location1** command can be used to report DFT hierarchy location specification for the current design.

EXAMPLES

The following example reports DFT hierarchy location for design top.

```
prompt> current_design top  
prompt> set_dft_location core_inst/dft_inst  
prompt> report_dft_location  
Design Name          DFT Hierarchy Location  
=====
```

top	core_inst/dft_inst
1	

```
prompt>
```

SEE ALSO

```
insert_dft(2)  
set_dft_location(2)  
remove_dft_location(2)
```

report_dft_partition

Displays design partition information attached to a design.

SYNTAX

status **report_dft_partition**

ARGUMENTS

The **report_dft_partition** command has no arguments.

DESCRIPTION

The **report_dft_partition** command displays design partition information about a design.

EXAMPLES

The following example shows a design partition report for the CORE design:

```
prompt> report_dft_partition
*****
Report : DFT PARTITION Definition
Design : CORE
Version: B-2008.09-alpha5
Date   : Fri Jun 13 16:36:29 2008
*****

Cells or Designs defined in Partition 'part1':
    U1
    U2
Cells or Designs defined in Partition 'part2':
    U3
    U4
    U5
```

SEE ALSO

```
current_dft_partition(2)
define_dft_partition(2)
insert_dft(2)
preview_dft(2)
remove_dft_partition(2)
set_dft_signal(2)
set_scan_compression_configuration(2)
set_scan_configuration(2)
set_scan_path(2)
```

report_dft_signal

Displays options specified by the set_dft_signal command.

SYNTAX

```
integer report_dft_signal  
[-view spec | existing_dft]  
[-test_mode list_of_mode_names | all]  
[-port list_of_port_names]  
[-type type]
```

Data Types

<i>list_of_mode_names</i>	list
<i>list_of_port_names</i>	list

ARGUMENTS

-view spec | existing_dft
Specifies the view from which to report. Valid options are **spec** and **existing_dft**. If not specified, the **spec** view is the default.

-test_mode list_of_mode_names | all
Specifies the test mode or test modes to report. If not specified, the tool reports on the *current_test_mode* that can be displayed by the **current_test_mode** command. If no test modes were created, the tool defaults to InternalScan mode. If **all** is specified, the tool displays the scan configurations of all test modes.

-port list_of_port_names
Specifies the name of a port or a list of ports on which to report. If no ports are specified, the tool reports on all defined ports.

DESCRIPTION

This command displays options set by the **set_dft_signal** command on the current design.

The command reports on signal types, active states, hook up pins, and timing of the ports.

EXAMPLES

The following command specifies a report on the test mode named mymodeA:

```
prompt> report_dft_signal -test_mode mymodeA  
  
*****  
Report : DFT signals
```

```

Design : A
Version: 2003.12
Date   : Thu Aug 14 10:46:08 2003
*****

```

```

=====
TEST MODE: mymodeA
VIEW      : Specification
=====

```

Port	SignalType	Active	Hookup	Timing
u5/A	ScanEnable	1	-	
SE_A	ScanEnable	1	u5/A	
SE_ALL	ScanEnable	-	-	
SI	ScanDataIn	-	-	Delay 5

The following command specifies a report on a list of test modes:

```

prompt> report_dft_signal -test_mode {mymodeA mymodeB}

```

```

*****
Report : DFT signals
Design : A
Version: 2003.12
Date   : Thu Aug 14 10:46:08 2003
*****

```

```

=====
TEST MODE: mymodeA
VIEW      : Specification
=====

```

Port	SignalType	Active	Hookup	Timing
u5/A	ScanEnable	-	-	
SE_A	ScanEnable	-	u5/A	
SE_ALL	ScanEnable	-	-	

```

=====
TEST MODE: mymodeB
VIEW      : Specification
=====

```

Port	SignalType	Active	Hookup	Timing
SE_B	ScanEnable	-	-	
u5/A	ScanEnable	-	-	
SE_A	ScanEnable	-	u5/A	
SE_ALL	ScanEnable	-	-	

SEE ALSO

```

current_test_mode(2)
report_dft_configuration(2)
report_dft_signal(2)
report_scan_configuration(2)

```

```
report_scan_path(2)
set_dft_configuration(2)
set_dft_signal(2)
set_scan_configuration(2)
set_scan_path(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_dp_smartgen_options

Displays datapath strategies available to the current design.

SYNTAX

```
integer report_dp_smartgen_options
```

ARGUMENTS

The **report_dp_smartgen_options** command has no arguments.

DESCRIPTION

This command lists the enabled and disabled datapath strategies available to the current design. These strategies provide control over some of the optimizations done by datapath generators.

EXAMPLES

The following is an example of a report generated by **report_dp_smartgen_options**:

```
prompt> report_dp_smartgen_options
*****
Datapath smart generation options...
*****

      Smart generation:          enabled
      Default values:          auto
```

SEE ALSO

```
set_dp_smartgen_options(2)
synlib_dwgen_smart_generation(3)
```

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_process_scale 1.4 -min_process_scale 0.6
```

```
prompt> report_extraction_options
```

```
*****  
Report : extraction options  
Design : module_a  
Version: B-2008.09-beta2  
Date   : Fri Jul 18 12:17:20 2008  
*****
```

```
    Max process scaling factor : 1.4  
    Min process scaling factor : 0.6
```

SEE ALSO

`set_extraction_options(2)`

report_fsm

Displays state-machine attributes and information for the design of the current instance or for the current design.

SYNTAX

```
integer report_fsm  
[-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

This command displays state-machine attributes and information for the design of the current instance or for 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.

EXAMPLES

The following is an example of a state-machine report:

```
prompt> report_fsm  
  
*****  
Report : fsm  
Design : BUS_ARBITRATOR  
Version: v2.0  
Date   : Fri Mar 20 14:11:13 1991  
*****  
Clock           : CLK           Sense: rising_edge  
Asynchronous Reset: Unspecified  
  
Encoding Bit Length: 3  
Encoding style      : auto  
  
State Vector: {FF2 FF1 FF0}  
  
State Encodings and Order:  
  
Grant_A      : 001  
Wait_A       : 011  
Timeout_A1   : 111
```

```
Grant_B      : 010
Wait_B       : 110
Timeout_B1   : 101
```

Preserved States: Grant_A

Merged States: None

SEE ALSO

```
set_fsm_encoding(2)
set_fsm_encoding_style(2)
set_fsm_minimize(2)
set_fsm_order(2)
set_fsm_preserve_state(2)
set_fsm_state_vector(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 multi-CPU processing options as defined by the *set_host_options* command.

SYNTAX

```
int report_host_options
```

DESCRIPTION

Prints a report of the multi-CPU processing options defined by the *set_host_options* command.

SEE ALSO

```
set_host_options(2)  
remove_host_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
```

```
report_ideal_network
```

```
1022
```

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. This command is supported only in topographical mode.

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.

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_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)  
report_compile_options(2)
```

```
set_isolate_ports(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]
[-isolation_strategy isolation_strategy_names]
[-ports pins_ports]
[-verbose]
[-nosplit]
```

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.

-isolation_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.

-ports *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.

-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_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  
  
=====
```

```
report_isolation_cell  
1036
```


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_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]
[-noise]
[-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]
[-noise_arcs]
```

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.

-noise
Lists all detailed noise-related information in the specified library.
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.

-noise_arcs

Lists all cell noise arcs.

This option applies only to logic 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
pwrg - 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 Compiler Modeling Timing, Signal Integrity, and Power in Technology Libraries User Guide*.

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	
latch1	5	i,a	QN	D	
	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)
report_synlib(2)
update_lib(2)
write_lib(2)

```

report_logicbist_configuration

Displays options specified by the set_logicbist_configuration command.

SYNTAX

```
integer report_logicbist_configuration  
[-test_mode test_mode_names | all]
```

Data Types

test_mode_names list

ARGUMENTS

```
-test_mode test_mode_names | all  
    Reports the options for the specified list of test modes or for all test  
    modes.
```

DESCRIPTION

This command displays the options set by the **set_logicbist_configuration** command on the current design.

It shows the parameters that determine the configuration of the logic built-in self-test (BIST) hardware.

EXAMPLES

The following command reports the configuration options set on the current design:

```
prompt> report_logicbist_configuration  
  
*****  
Report : BIST configuration  
Design : CORE  
Version: 2005.09  
Date   : Fri Jun  3 03:25:12 2005  
*****  
  
=====
```

TEST MODE:	all_dft
VIEW	: Specification

```
=====
```

Integration	True
PrpgLength	257
CodecCount	2
MaxChainLength	10
AutoArchitect	True

BalanceBistSegments	True
PrpgShadowSi	7
SelectorShadowSi	5
ObserveOutput	9
InvertPrpgClock	True
ClockMISRDuringCapture	False

SEE ALSO

```
report_dft_configuration(2)
set_dft_configuration(2)
set_logicbist_configuration(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_multibit

Displays information about multibit components in the current design or the subdesign.

SYNTAX

```
status report_multibit
[-nosplit]
[object_list]
```

Data Types

object_list 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.

object_list

Specifies a list of cells and multibit components to show in the report. If this option is not specified, all multibit components in the current design are listed. For specified cells, the report is done on multibit components that contain the specified cells.

DESCRIPTION

The **report_multibit** command displays information and statistics about multibit components in the current design or *current_instance*. A multibit component exists in a design due to the instantiation of a multibit library cell, inference from the HDL source, or an invocation of the **create_multibit** command.

The data displayed by this command may not be accurate if the design cannot be linked.

EXAMPLES

The following example shows a report on all of the multibit components in the design:

```
prompt> report_multibit
```

```
*****
Report : multibit
Design : subtest
```

Version: 1998.02
Date : Fri Aug 29 10:01:19 1997

Attributes:
b - black box (unknown)
h - hierarchical
n - noncombinational
r - removable
u - contains unmapped logic

Multibit Component : y_reg					
Cell	Reference	Library	Area	Width	Attributes

y_reg[3]	**SEQGEN**		0.00	1	n, u
y_reg[2]	**SEQGEN**		0.00	1	n, u
y_reg[1]	**SEQGEN**		0.00	1	n, u
y_reg[0]	**SEQGEN**		0.00	1	n, u

Total 4 cells			0.00	4	

Multibit Component : z_reg					
Cell	Reference	Library	Area	Width	Attributes

z_reg[3]	**SEQGEN**		0.00	1	n, u
z_reg[2]	**SEQGEN**		0.00	1	n, u
z_reg[1]	**SEQGEN**		0.00	1	n, u
z_reg[0]	**SEQGEN**		0.00	1	n, u

Total 4 cells			0.00	4	

Total 2 Multibit Components

The following example shows a report on one multibit component:

```
prompt> report_multibit y_reg

*****
Report : multibit
Design : subtest
Version: 1998.02
Date   : Fri Aug 29 10:01:19 1997
*****
```

Attributes:
b - black box (unknown)
h - hierarchical
n - noncombinational
r - removable
u - contains unmapped logic

Multibit Component : y_reg					
Cell	Reference	Library	Area	Width	Attributes

y_reg[3]	**SEQGEN**	0.00	1	n, u
y_reg[2]	**SEQGEN**	0.00	1	n, u
y_reg[1]	**SEQGEN**	0.00	1	n, u
y_reg[0]	**SEQGEN**	0.00	1	n, u

Total 4 cells		0.00	4	

Total 1 Multibit Components

The following example shows a report generated with a cell as an argument:

```
prompt> report_multibit y_reg[0]
```

```
*****
Report : multibit
Design : subtest
Version: 1998.02
Date   : Fri Aug 29 10:01:19 1997
*****
```

Attributes:

```
  b - black box (unknown)
  h - hierarchical
  n - noncombinational
  r - removable
  u - contains unmapped logic
```

Multibit Component : y_reg

Cell	Reference	Library	Area	Width	Attributes
y_reg[3]	**SEQGEN**		0.00	1	n, u
y_reg[2]	**SEQGEN**		0.00	1	n, u
y_reg[1]	**SEQGEN**		0.00	1	n, u
y_reg[0]	**SEQGEN**		0.00	1	n, u

Total 4 cells			0.00	4	

Total 1 Multibit Components

The following example shows **report_multibit** with the *y_reg* multibit component from the *U1/BOT1* design instance:

```
prompt> report_multibit U1/BOT1/y_reg
```

```
*****
Report : multibit
Design : subtest
Version: 1998.02
Date   : Fri Aug 29 10:01:19 1997
*****
```

Attributes:

```
  b - black box (unknown)
```

h - hierarchical
 n - noncombinational
 r - removable
 u - contains unmapped logic

Multibit Component : y_reg					
Cell	Reference	Library	Area	Width	Attributes
U1/BOT1/y_reg[3]	**SEQGEN**		0.00	1	n, u
U1/BOT1/y_reg[2]	**SEQGEN**		0.00	1	n, u
U1/BOT1/y_reg[1]	**SEQGEN**		0.00	1	n, u
U1/BOT1/y_reg[0]	**SEQGEN**		0.00	1	n, u
Total 4 cells			0.00	4	

Total 1 Multibit Components

SEE ALSO

create_multibit(2)
 current_design(2)
 remove_multibit(2)
 report_cell(2)
 report_compile_options(2)
 report_reference(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:

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
```

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	Allowed Chars
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	Allowed Chars
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.

-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:

```
report_net_fanout  
1080
```

```
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

```
current_design(2)  
current_instance(2)  
report_buffer_tree(2)
```

report_operand_isolation

Reports the status of operand isolation cells in the current design.

SYNTAX

```
status report_operand_isolation  
[-instances]  
[-isolated_objects]  
[-unisolated_objects]  
[-all_objects]  
[-verbose]  
[-no_hier]  
[-nosplit]  
[object_list]
```

Data Types

object_list list

ARGUMENTS

- instances
Specifies that a breakdown per instance of the number of isolated and unisolated operators and hierarchical combinational cells should be reported. This option cannot be used in combination with **-no_hier**.
- isolated_objects
Specifies that the names of all isolated objects are to be reported.
- unisolated_objects
Specifies that the names of all operators and hierarchical cells that were not isolated are to be reported.
- all_objects
Specifies that all operators and hierarchical combinational cells are to be reported, whether they were isolated or not. Using this option is equivalent to using both **-isolated_objects** and **-unisolated_objects** options.
- verbose
Specifies that the details on the isolation gates should be reported for each isolated object. This option can only be used together with **-isolated_objects**, **-all_objects** or a specific list of objects.
- no_hier
Specifies that the operand isolation information is to be reported only for the top level of the current design or current instance. Without this option, all levels of hierarchy starting from the current design or instance are reported.
- 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 the information for a field exceeds the column width, the next field begins on a new line, starting at the correct column.

`object_list`

Specifies the list of objects for which the report should be generated. When a list of objects is specified, the **-isolated_objects**, **-unisolated_objects** and **-all_objects** options cannot be used.

DESCRIPTION

Use of the **report_operand_isolation** command is not recommended. This feature will be obsolete in a future release. The **report_operand_isolation** command reports the status of operand isolation cells in the current design. The report is divided into different sections: isolated objects, unisolated objects, breakdown per instance and global summary. The summary is generated when object-specific report is not requested. More details about the isolated operators and hierarchical combinational cells are reported when the **-verbose** option is used.

EXAMPLES

The following is a sample output of **report_operand_isolation** command when no options are specified. The output is limited to a summary of the operand isolation present in the entire design.

```
prompt>report_operand_isolation
```

```
*****
Report : isolation
Design : sample
Version: W-2004.12
Date   : Wed Oct 27 20:27:59 2004
*****
```

Library(s) Used:

power_lib (File: /root/libraries/power_lib.db)

Operand Isolation Summary

Isolation Style	adaptive
Isolation Method	automatic
Number of Isolation gates	22
Number of Isolated objects	3 (42.86%)
operators	1 (14.29%)
hierarchical cells	1 (14.29%)
ungrouped objects	1 (14.29%)
Number of Unisolated objects	4 (57.14%)
operators	3 (42.86%)
hierarchical cells	1 (14.29%)

The following example shows the report with all isolated and unisolated objects, as well as a summary of breakdown per instance. Note that the global summary is also reported.

```
prompt>report_operand_isolation -all_objects -instances
```

```
*****
Report : isolation
        -all_objects
        -instances
Design : sample
Version: W-2004.12
Date   : Wed Oct 27 20:34:03 2004
*****
```

Library(s) Used:

power_lib (File: /root/libraries/power_lib.db)

Isolated Objects Report

Isolated Object	Object Type	Method
add_21	operator	auto
U1	hierarchical	auto
sub_23	ungrouped	auto

Unisolated Objects Report

Unisolated Object	Object Type
add_25	operator
U2	hierarchical
U2/mult_5	operator
U1/mult_5	operator

Breakdown per Instance

Isol.	Isolated			Unisolated		
Gates	Oper	Hier	Ungr	Oper	Hier	Parent Instance
22	1	1	1	1	1	<top level>
0	0	0	0	1	0	U2
0	0	0	0	1	0	U1

Operand Isolation Summary

Isolation Style	adaptive

Isolation Method	automatic
Number of Isolation gates	22
Number of Isolated objects	3 (42.86%)
operators	1 (14.29%)
hierarchical cells	1 (14.29%)
ungrouped objects	1 (14.29%)
Number of Unisolated objects	4 (57.14%)
operators	3 (42.86%)
hierarchical cells	1 (14.29%)

In the following example a detailed report of the isolation for operator add_21 is generated. Note that in this case no global summary is reported because the **report_operand_isolation** is specified for a collection of objects.

```
prompt>report_operand_isolation [get_cells add_21] -verbose
```

```
*****
Report : isolation
       add_21
       -verbose
Design : sample
Version: W-2004.12
Date   : Wed Oct 27 20:40:18 2004
*****
```

Library(s) Used:

```
power_lib (File: /root/libraries/power_lib.db)
```

Isolated Objects Report

```
-----
Parent Instance: <top_level>
```

```
Isolated Object: add_21
```

```
Object Type: operator
```

```
Style: adaptive
```

```
Method: auto
```

```
Control Signal: n29
```

```
Gate Count : 12
```

Original Data Net	Isolated Pin	Isolation Gate	Type
a[5]	add_21/A[5]	C1	AND
a[4]	add_21/A[4]	C2	AND
a[3]	add_21/A[3]	C3	AND
a[2]	add_21/A[2]	C4	AND
a[1]	add_21/A[1]	C5	AND
a[0]	add_21/A[0]	C6	AND
d[5]	add_21/B[5]	C13	AND
d[4]	add_21/B[4]	C14	AND
d[3]	add_21/B[3]	C15	AND

d[2]	add_21/B[2]	C16	AND
d[1]	add_21/B[1]	C17	AND
d[0]	add_21/B[0]	C18	AND

SEE ALSO

set_operand_isolation_style(2)
do_operand_isolation(3)

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_partitions

Lists the hierarchical designs and their associated attributes and relative size (estimated in RTL).

SYNTAX

```
int report_partitions
[-nosplit]
```

ARGUMENTS

-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.

DESCRIPTION

This command displays the partition-related information, including estimated relative size (or cell area, for mapped designs), number of instantiations, and which instance is the master instance.

EXAMPLES

The following is an example of a partition report:

```
prompt> report_partitions
```

```
*****
Report      : partitions
Design      : vampire (unmapped)
Date        : Thu Feb  8 14:47:48 2001
*****
```

Partition attributes :

```
(*)  - partition.
%    - the percentage w.r.t total design area.
d    - dont_touch.
m(n) - multiple instantiated design (number of instantiation).
```

Designs	Attributes
vampire(*)	15.6%
tap_block(*)	6.8%, m(7), u1
add(*)	1.8%, m(8), u0/u0
mult(*)	1.5%, m(8), u0/u1
first_tap(*)	6.9%

<code>cascade</code>	1.5%
<code>accum(*)</code>	4.0%

SEE ALSO

`set_compile_partitions(2)`

report_pass_data

Reports the data files that are available for a design created by Automated Chip Synthesis.

SYNTAX

```
status report_pass_data  
[-hierarchy]  
[-pass_list pass_list]  
[design]
```

Data Types

<i>pass_list</i>	list
<i>design</i>	string

ARGUMENTS

-hierarchy

Requests data for all compile partitions in the hierarchy of the specified design. Because compile partitions with a **dont_touch** attribute are not modified, these partitions are not included in the report. By default, this command returns data only for the specified design.

-pass_list *pass_list*

Specifies a list of pass names. Pass names that contain the "*" wildcard character suffix are acceptable. Default pass names will be *default_prefix**, in which *default_prefix* is defined by *acs_default_pass_name*. An error occurs if a pass name does not exist.

design

Specifies the design to for which to report the data. You can specify only one design. An error occurs if the specified design does not exist in Design Compiler memory. By default, if you do not specify a design, Automated Chip Synthesis uses the current design.

DESCRIPTION

The **report_pass_data** command locates all passes in *pass_list* directories under the Automated Chip Synthesis working directory (as specified by **acs_work_dir**), and reports on the data files present in each pass for the specified design. Specifically, the command checks for the following files:

- pass/db/pre_compile/*design*.db
- pass/db/post_compile/*design*.db
- pass/scripts/*design*.autoscr

The file suffixes shown above are the default file suffixes. You can modify these suffixes using the **acs_db_suffix** variable for .db files, and the **acs_constraint_file_suffix** variable for the script file.

EXAMPLES

Assume you must run two passes (pass0 and pass1) for the design named my_design. To check on the data available for this design, execute one of the following commands:

```
prompt> report_pass_data my_design -pass_list {pass0 pass1}
```

```
prompt> report_pass_data my_design -pass_list {pass*}
```

This command generates the following report:

Design	Pass	Pre-compile Database	Post-compile Database	Script
my_design	pass0	YES	YES	YES
	pass1	YES	NO	YES

SEE ALSO

current_design(2)
acs_compile_script_suffix(3)
acs_db_suffix(3)
acs_work_dir(3)

report_path_budget

Displays budgeting information about a design.

SYNTAX

```
integer report_path_budget  
[-to to_list]  
[-from from_list]  
[-through through_list]  
[-nworst paths_per_endpoint]  
[-max_paths max_path_count]  
[-input_pins]  
[-nets]  
[-transition_time]  
[-significant_digits digits]  
[-nosplit]  
[-all]  
[-verbose]
```

Data Types

<i>to_list</i>	list
<i>from_list</i>	list
<i>through_list</i>	list
<i>paths_per_endpoint</i>	integer
<i>max_path_count</i>	integer
<i>digits</i>	integer

ARGUMENTS

- to *to_list***
Reports only the paths to pins, ports, or clocks specified by *to_list*. The default is to report the longest path to an output port, with the worst slack within each path group.
- from *from_list***
Reports only the paths from pins, ports, or clocks specified by *from_list*. The default is to report the longest path to an output port, with the worst slack within each path group.
- through *through_list***
Reports only the paths through pins specified by *through_list*. The default is to report the longest path to an output port, with the worst slack within each path group.
- nworst *paths_per_endpoint***
Specifies the number of paths per endpoint to report. The default is **1**.
- max_paths *max_path_count***
Specifies the number of paths per path group to report. The default is **1**.

-input_pins
Specifies that input pins are to be shown 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
Specifies that nets are to be shown 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
Specifies that the net transition time for each driving pin is to be shown in the path report. The default is not to show the net transition time for each driving pin.

-significant_digits *digits*
Specifies the number of reported digits to be to the right of the decimal point in the report. Valid values are **0** to **13**. The default value is **2**. Using this option overrides the value set by the **report_default_significant_digits** variable.

-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.

-all
Reports all hierarchical pins. By default, the command reports only the budget number on the hierarchical pins of the cell that have been budgeted. When the **-all** option is specified, the command reports budget numbers on all hierarchical pins on this path.

-verbose
Reports the budget number on every cell along the path.

DESCRIPTION

This command provides a report of budgeting information for the current design. The delay number shown for each block or cell shows the amount of budget allocated for that block or cell in the path. The overall budget for the path is the sum of these budgets (delays).

EXAMPLES

The following example shows a budgeting report using only the default values:

```
prompt> report_path_budget -th [get_pins i_driver/OUT]
```

```
*****
```

```
Report : budget
        -path full
        -delay max
```

```

-max_paths 1
Design : test
Version: 2002.05
Date   : Tue Mar 19 11:20:24 2002
*****

```

Attributes

```

-----
U   User Specified Budgets
A   Automatic Budgets
F   Fixed delay

```

```

Operating Conditions:
Wire Load Model Mode: top

```

```

Startpoint: i_driver/rIN_reg
            (rising edge-triggered flip-flop clocked by clk1)
Endpoint:   i_receiver/OUT_reg
            (rising edge-triggered flip-flop clocked by clk2)
Path Group: clk2
Path Type:  max

```

Des/Clust/Port	Wire Load Model	Library		
test	tc6a120m2	cba_core		
Point	Budget	Delay	Slack	Type

clock clk1 (rise edge)				
	200.00	200.00	0.00	F
clock network delay	0.00	0.00	0.00	F
i_driver/rIN_reg/CLK				
	0.00	0.00	0.00	A
i_driver/rIN_reg/Q	400.00	837.75	-437.75	U
i_driver/OUT	400.00	136.27	263.73	U
i_receiver/IN	0.00	0.00	0.00	F
i_receiver/OUT_reg/D				
	800.00	1024.02	-224.02	U

Total	1800.00	2198.04	-398.04	
Required time	708.44	708.44		

Slack	-1091.56	-1489.60		

The following example shows a budgeting report that displays the budget number on every cell along the path:

```

prompt> report_path_budget -verbose -th [get_pins i_driver/OUT]

```

```

*****
Report : budget
        -path full
        -delay max
        -max_paths 1

```

Design : test
Version: 2002.05-preprod
Date : Tue Mar 19 11:20:24 2002

Attributes

U User Specified Budgets
A Automatic Budgets
F Fixed delay

Operating Conditions:
Wire Load Model Mode: top

Startpoint: i_driver/rIN_reg
(rising edge-triggered flip-flop clocked by clk1)
Endpoint: i_receiver/OUT_reg
(rising edge-triggered flip-flop clocked by clk2)
Path Group: clk2
Path Type: max

Des/Clust/Port	Wire Load Model	Library		
test	tc6a120m2	cba_core		
Point	Budget	Delay	Slack	Type

clock clk1 (rise edge)				
	200.00	200.00	0.00	F
clock network delay	0.00	0.00	0.00	F
i_driver/rIN_reg/CLK				
	0.00	0.00	0.00	A
i_driver/rIN_reg/Q	400.00	837.75	-437.75	U
i_driver/U1/A	0.00	0.00	0.00	F
i_driver/U1/Y	400.00	136.27	263.73	U
i_driver/OUT	0.00	0.00	0.00	F
i_receiver/IN	0.00	0.00	0.00	F
i_receiver/U1/A	0.00	0.00	0.00	F
i_receiver/U1/Y	100.00	118.36	-18.36	U
i_receiver/U2/A	0.00	0.00	0.00	F
i_receiver/U2/Y	100.00	141.71	-41.71	U
i_receiver/U3/A	0.00	0.00	0.00	F
i_receiver/U3/Y	100.00	118.97	-18.97	U
i_receiver/U4/A	0.00	0.00	0.00	F
i_receiver/U4/Y	100.00	141.83	-41.83	U
i_receiver/U5/A	0.00	0.00	0.00	F
i_receiver/U5/Y	100.00	118.98	-18.98	U
i_receiver/U6/A	0.00	0.00	0.00	F
i_receiver/U6/Y	100.00	141.84	-41.84	U
i_receiver/U7/A	0.00	0.00	0.00	F
i_receiver/U7/Y	100.00	118.98	-18.98	U
i_receiver/U8/A	0.00	0.00	0.00	F
i_receiver/U8/Y	100.00	123.34	-23.34	U
i_receiver/OUT_reg/D				
	0.00	0.00	0.00	F

Total	1800.00	2198.04	-398.04
Required time	708.44	708.44	
Slack	-1091.56	-1489.60	

SEE ALSO

```
dc_allocate_budgets(2)
set_user_budget(2)
report_default_significant_digits(3)
```

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
1098
```

```

*****
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_physical_constraints

Reports the current physical constraints setting. This command is supported only in topographical mode.

SYNTAX

```
status report_physical_constraints  
[-no_site_row]  
[-pre_route]
```

ARGUMENTS

-no_site_row
Specifies that site rows are not reported. By default, the site rows are reported block by block.

-pre_route
Specifies that the pre-routes will be reported if they exist. By default, pre-routes are not reported.

DESCRIPTION

The **report_physical_constraints** command reports the physical constraints settings that are previously applied to the design. The command reports the following physical constraints.

- die area
- placement area
- utilization
- aspect ratio
- rectilinear outline
- port side
- port location
- cell location
- placement blockage
- wiring keepouts
- voltage area
- site rows
- bounds
- pre-routes

To report the physical constraints of pre-routes, specify the **-pre_route** option.

Note that because of priority considerations, the physical constraints with respect to the same type of object will not all be reported by **report_physical_constraints**. Two rules help determine which constraints are in effect and will be reported.

1. For the core area constraints set by the commands listed below, the placement area constraint has the highest priority, and rectilinear outline has a higher priority than utilization and aspect ratio:


```
set_placement_area
set_rectilinear_outline
set_utilization
set_aspect_ratio
```

2. Between the two constraints for port, port location has higher priority than port side. For example, assume the following constraints are set:

```
set_placement_area -coordinate {10 10 1000 1000}
set_utilization 0.9
set_port_location Clk -coordinate {100 200}
set_port_side -side top {Clk Reset}
```

When the **report_physical_constraints** command is invoked, the report shows the above constraints as follows:

Placement area	{0.000 0.000 100.000 100.000}
PORT LOCATION 1	LOCATION
Clk	{100.000 200.000}
PORT SIDE 1	SIDE
Reset	top

The utilization and the port side of the **Clk** port are omitted due to the constraints priority.

EXAMPLES

The following is an example of the command usage:

```
prompt> report_physical_constraints
```

SEE ALSO

```
create_bounds(2)
create_net_shape(2)
create_placement_blockage(2)
create_site_row(2)
create_wiring_keepouts(2)
create_voltage_area(2)
extract_physical_constraints(2)
set_aspect_ratio(2)
set_cell_location(2)
set_placement_area(2)
set_port_location(2)
set_port_side(2)
```

```
set_utilization(2)
write_physical_constraints(2)
```

report_pin_map

Displays package pin map information set by the `read_pin_map` command.

SYNTAX

```
status report_pin_map
```

ARGUMENTS

The **report_pin_map** command has no arguments.

DESCRIPTION

The **report_pin_map** command displays package pin map information set by the **read_pin_map** command on the current design.

EXAMPLES

The following is an example of the **report_pin_map** command:

```
prompt> report_pin_map

*****
Report : BSD Pin Map
Design : M1
Version: Z-2007.03
Date   : Fri Dec 21 08:44:55 2006
*****

Package Name      : M1
File Name         : /designs/bss_ut4_1.map

Port Name          Pin
-----
in[2]              6
in[1]              5
in[0]              1
clk                20
en                 21
out                22
out1[2]            25
out1[1]            24
out1[0]            23
tck                28
tms                29
tdi                30
tdo                32
trst_n             33
dummy1[0]          34
dummy1[1]          35
```

<code>dummy1[2]</code>	36
<code>dummy2[1]</code>	37
<code>dummy2[2]</code>	38
<code>dummy2[3]</code>	39

SEE ALSO

`read_pin_map(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_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
```

```
report_port
1108
```


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	--

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 Trans	Min Cap	Min Fanout	Cell Deg
	Rise	Fall				
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:

report_port
1110

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		Max		Related Clock	Max Fanout
	Rise	Fall	Rise	Fall		
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		Min Drive		Resistance		Min Cap	Min Fanout	Cell Deg
	Rise	Fall	Rise	Fall	Max	Min			
i1	--	--	--	--	--	--	0.548	--	--
i2	--	--	--	--	--	--	--	--	--
i3	--	--	--	--	--	--	--	--	--

Input Port	Max Tran		Min Tran	
	Rise	Fall	Rise	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		Max		Related Clock	Fanout Load
	Rise	Fall	Rise	Fall		

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)
 set_wire_load_mode(2)
 set_wire_load_model(2)
 set_wire_load_min_block_size(2)
 set_wire_load_selection_group(2)
 report_default_significant_digits(3)

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. In DC-Topographical mode, with **-cell** option the power report includes estimated clock tree power numbers if the **-only** option is not used, and if power prediction was turned on in the last **compile_ultra** run. This entry is marked *CLOCK_TREE_EST* in the cell column.

-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.

When invoked the **report_power** command 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_power** command, set the **power_keep_license_after_power_commands** variable to **true**.

In Topographical mode, if Power Prediction is enabled, **report_power** reports the correlated power. The reported power numbers reported include the consumption by design components as well as the predicted power of missing components, such as clock tree. The correlated power is not reported if any options to **report_power** are used.

When leakage power model is set to **channel_width** (**set_leakage_power_model**), the

total design power reported by **report_power** command includes the total weighted sum of the channel widths for the design.

Creation and usage of scenarios is available with Design Compiler Graphical.

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	

report_power

1118

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

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)

The following example shows a **report_power** summary report in DC-Topographical mode with power prediction on. Notice the PWR-620 message flagging this as correlated power.

```
prompt> set_power_prediction
prompt> compile_ultra -incr
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 :

report_power
1120

Voltage Units = 1V
Capacitance Units = 50.029999ff
Time Units = 1ns
Dynamic Power Units = 1mW (derived from V,C,T units)
Leakage Power Units = 1nW

Information: Reporting correlated power. (PWR-620)

Cell Internal Power	=	157.0383 nW	(86%)
Net Switching Power	=	26.1555 nW	(14%)

Total Dynamic Power	=	183.1938 nW	(100%)
Cell Leakage Power	=	1.0225 mW	

SEE ALSO

propagate_switching_activity(2)
set_power_prediction(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
```

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)
(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

```

```

      Z = A + B*X + C*Y + D*X*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 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

```
status report_power_domain
```

```
[power_domains]
```

Data Types

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 current scope.

In non-UPF mode, if this this option is not specified, the command reports all power domains in the current design.

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

report_power_domain

1128

- 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 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_domain

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.

```
prompt>report_power_gating F00
```

SEE ALSO

```
set_power_gating_style(2)  
power_enable_power_gating(3)
```


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_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
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

```

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  
[power_switch_name]  
[-verbose]
```

Data Types

power_switch_name string or list

ARGUMENTS

power_switch_name
Specifies a list of power switch names. The individual names must be hierarchical relative to the current scope. If the switch exists in the current scope, its simple name can be used. If no power switches specified absent, the tool reports all of the power switches in the current scope.

-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
```

```
report_power_switch  
1140
```

```
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)`

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_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]  
[-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.
- 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.

Creation and usage of scenarios is available with Design Compiler Graphical.

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_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_resources

Lists the resources and datapath blocks used in the design of the current instance or in the current design.

SYNTAX

```
integer report_resources  
[-nosplit]  
[-hierarchy]
```

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. 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

Reports information about all resources used in the hierarchy of the current design or the current instance. By default, resources used only in the top level of the current design or the current instance are reported.

DESCRIPTION

This command lists the resources and datapath blocks used in the design of the current instance or in the current design. The command displays information about the current design or about resources and datapath blocks used in the design. If the current instance is set, the report is generated for the design of that instance. Otherwise the report is generated for the current design.

A resource is an arithmetic or comparison operator read in as part of an HDL design. Resources can be shared when compiling the design and are reported after the compile operation completes.

A datapath block contains one or more resources that are grouped together and optimized by a datapath generator.

This report also has information on the parameters used to build the module, user-declared resources in each module, shared operations, and resources that are transformed into datapath blocks.

A detailed report is printed in the DC Ultra flow. The report displays the following information about the datapath blocks in the design:

- Design name
- Source file location
- Input and output connections for each datapath operand and bit width

- Operation name with source line number
- Datapath operation or expression

EXAMPLES

The following example displays resource information for the current design:

```
prompt> report_resources
```

```
*****
Report : resources
Design : REGCNT
Version: A-2007.12-SP1
Date   : Fri Jan  4 01:59:13 2008
*****
```

Resource Sharing Report for design REGCNT in file
/am/remote/cae7/testcases/DAC94/suite/hadv/src2/verilog/regcnt.v

Cell	Module	Parameters	Contained Operations
sub_x_0	DW01_dec	width=5	sub_6

Implementation Report

Cell	Module	Current Implementation	Set Implementation
sub_x_0	DW01_dec	rpl	

The following example shows the datapath extraction report for the design named mult_add. As shown in the example, the Datapath Report has two tables for each datapath block. The first table shows the datapath block along with the contained operations. The second table shows datapath expressions for the block along with the type (Primary Input, Primary Output or Internal FanOut), signage and bit width of each variable in the expression.

```
prompt> report_resources
```

```
*****
Report : resources
Design : mult_add
Version: A-2007.12-SP1
Date   : Fri Jan  4 01:59:13 2008
*****
```

Resource Report for this hierarchy in file
/remote/disks/user/dp_report/dp_test.v

Cell	Module	Parameters	Contained Operations
DP_OP_4_296	DP_OP_4_296		

Datapath Report for DP_OP_4_296

Cell	Contained Operations
DP_OP_4_296	mult_16 add_18 mult_17

Var	Type	Data Class	Width	Expression
I1	PI	Unsigned	4	
I2	PI	Unsigned	4	
I3	PI	Unsigned	4	
I4	PI	Unsigned	4	
T0	IFO	Unsigned	8	I1 * I2
T1	IFO	Unsigned	8	I3 * I4
O1	PO	Unsigned	8	T0 + T1

Implementation Report

Cell	Module	Current Implementation	Set Implementation
DP_OP_4_296	DP_OP_4_296	str	

The following example displays resource and datapath information for the current design:

prompt> **report_resources -hierarchy**

```
*****
Report : resources
Design : trunc3
Version: A-2007.12-SP1
Date   : Fri Jan  4 02:33:50 2008
*****
```

Resource Report for this hierarchy in file ./designs/trunc3.v

Cell	Module	Parameters	Contained Operations
add_x_9_0	DW01_add	width=6	add_9
DP_OP_6_296	DP_OP_6_296		

Datapath Report for DP_OP_6_296

Cell		Contained Operations		
DP_OP_6_296		sub_8 add_8		

Var	Type	Data Class	Width	Expression
I1	PI	Unsigned	5	
I2	PI	Unsigned	5	
I3	PI	Unsigned	5	
O1	PO	Signed	7	I1 - I2 + I3

Implementation Report

Cell	Module	Current Implementation	Set Implementation
DP_OP_6_296	DP_OP_6_296	str	
add_x_9_0	DW01_add	rpl	

No multiplexors to report

```
*****
Design : trunc3_DW01_add_0
*****
```

No resource sharing information to report.

No implementations to report

No multiplexors to report

```
*****
Design : trunc3_DP_OP_6_296_0
*****
```

No resource sharing information to report.

No implementations to report

No multiplexors to report

SEE ALSO

```
compile(2)
compile_ultra(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]
[-retention_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.

-retention_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 **-retention_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_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
```

```
report_saif
```

```
1162
```

```

-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

report_scan_compression_configuration

Displays options specified by the `set_scan_compression_configuration` command or the `reset_scan_compression_configuration` command.

SYNTAX

```
status report_scan_compression_configuration  
[-test_mode test_mode_names]
```

ARGUMENTS

The **report_scan_compression_configuration** command has no arguments.

DESCRIPTION

This command displays options specified by the **set_scan_compression_configuration** command on the current design.

The command reports the following information:

- Minimum compression
- Type of integration (if any)
- Level of xtolerance
- Compression chain synchronization
- Diagnosis accuracy
- Number of inputs
- Number of outputs
- Compression mode chain count
- Maximum chain length set

EXAMPLES

The following is an example of a scan compression configuration report:

```
prompt> report_scan_compression_configuration  
  
*****  
Report : Scan Compression Configuration  
Design : des_unit  
Version: Z-2007.03
```

Date : Mon Jan 22 09:22:02 2007

Minimum Compression	50
Integration	False
Hybrid	False
Xtolerance	high
force_diagnosis	False
Inputs	0
Outputs	0
Chain Count	0
Maximum Chain Length	0

1

SEE ALSO

current_test_mode(2)
set_scan_compression_configuration(2)
reset_scan_compression_configuration(2)

report_scan_configuration

Displays options specified by the `set_scan_configuration` command.

SYNTAX

```
status report_scan_configuration  
[-test_mode mode_names | all]
```

ARGUMENTS

```
-test_mode mode_names | all
```

Specifies the test mode to report. If not specified, the command applies to the current test mode that can be displayed by the **current_test_mode** command. If no test modes were created, the default is to use InternalScan. If **all** is specified, this option displays scan configuration information for all of the test modes.

DESCRIPTION

This command displays the options specified by the **set_scan_configuration** command on the current design.

The command reports the chain count, scan style, maximum scan chain length, rebalance scan chain length, preserve multibit segments, clock mixing, internal clocks, add lockup, insert terminal lockup, create dedicated scan out ports, shared scan in, and bidirectional mode.

EXAMPLES

The following is an example of a scan configuration report on the test mode named *mymodeA*:

```
prompt> report_scan_configuration -test_mode mymodeA  
  
*****  
Report : Scan configuration  
Design : A  
Version: 2003.12  
Date   : Thu Aug 14 10:46:07 2003  
*****  
  
=====
```

TEST MODE:	mymodeA
VIEW	: Specification

```
=====
```

Chain count:	Undefined
Scan Style:	Multiplexed flip-flop
Maximum scan chain length:	100
Rebalance scan chain length:	False

Preserve multibit segments:	True
Clock mixing:	Not defined
Internal clocks:	False
Add lockup:	True
Insert terminal lockup:	False
Create dedicated scan out ports:	False
Shared scan in:	0
Bidirectional mode:	No bidirectional type

The following is an example of a scan configuration report on all of the test modes:

```
prompt> report_scan_configuration -test_mode all
```

```
*****
Report : Scan configuration
Design : A
Version: 2003.12
Date   : Thu Aug 14 10:46:07 2003
*****
```

```
=====
TEST MODE: mymodeA
VIEW      : Specification
=====
Chain count:                Undefined
Scan Style:                 Multiplexed flip-flop
Maximum scan chain length:  100
Rebalance scan chain length: False
Preserve multibit segments: True
Clock mixing:               Not defined
Internal clocks:            False
Add lockup:                 True
Insert terminal lockup:     False
Create dedicated scan out ports: False
Shared scan in:             0
Bidirectional mode:         No bidirectional type
```

```
=====
TEST MODE: mymodeB
VIEW      : Specification
=====
Chain count:                Undefined
Scan Style:                 Scan style not defined
Maximum scan chain length:  200
Rebalance scan chain length: False
Preserve multibit segments: False
Clock mixing:               Not defined
Internal clocks:            False
Add lockup:                 True
Insert terminal lockup:     False
Create dedicated scan out ports: False
Shared scan in:             0
Bidirectional mode:         No bidirectional type
```

SEE ALSO

current_test_mode(2)
report_dft_configuration(2)
report_dft_signal(2)
report_scan_path(2)
report_scan_configuration(2)
set_dft_configuration(2)
set_dft_signal(2)
set_scan_configuration(2)
set_scan_path(2)

report_scan_group

Reports all scan group that were specified using the **set_scan_group** command.

SYNTAX

```
int report_scan_group  
scan_group_names
```

ARGUMENTS

scan_group_names

The names of scan groups that you want to report. The scan groups must be previously defined using the **set_scan_group** command.

DESCRIPTION

This command reports all scan groups that were specified using the **set_scan_group** command. If no names are specified, all previously defined scan groups are reported.

If you specify the name of a scan group to report, **report_scan_group** also reports the names of the elements within that group.

If the name you specify has not been previously defined as a scan group, the command issues a warning and exits.

EXAMPLES

The following example show how to use the **report_scan_group** command.

```
prompt> report_scan_group group1
```

```
prompt> report_scan_group all
```

```
prompt> report_scan_group
```

SEE ALSO

```
insert_dft(2)  
preview_dft(2)  
set_scan_group(2)  
remove_scan_group(2)
```


report_scan_link

Reports the scan links specified with the `set_scan_link` command.

SYNTAX

```
int report_scan_link  
-test_mode mode_name  
[link_type]
```

Data Types

mode_name string

ARGUMENTS

`-test_mode mode_name`
 Specifies the name of the mode for which the specifications are reported. The default is current mode.

DESCRIPTION

The **report_scan_link** command reports scan links specified with the **set_scan_link** command. Use the **-test_mode** option to report the scan links specified in a given mode. If the **-test_mode** option is not specified, the scan links in the current mode are reported.

Scan links connect scan cells, scan segments, and scan ports within scan chains. DFT Compiler supports scan links that are implemented as wires and scan-out lock-up latches.

Use the **report_scan_link** command to view the specified scan links.

To review your scan link specifications, use the **preview_scan** command with the **-show** option set to **cells**. To create scan links and add them to the current design, use the **insert_scan** command.

EXAMPLES

The following example declares a scan-out lock-up latch named *a_lckup* and then removes it with the **remove_scan_link**.

```
prompt> current_design WC66
```

```
prompt> set_scan_link a_lckup Lockup
```

```
prompt> report_scan_link
```

```
=====
TEST MODE: Internal_scan
VIEW      : Specification
```

```
=====
      LOCKUP              WIRES
      -----            -----
      my_lockup1          my_wire1
```

SEE ALSO

```
current_design(2)
remove_scan_link(2)
set_dont_touch(2)
set_scan_configuration(2)
set_scan_element(2)
set_scan_link(2)
set_scan_path(2)
write(2)
test_default_scan_style(3)
```

report_scan_path

Displays scan paths and scan cells specified by the `set_scan_path` command and displays scan paths inserted by the `insert_dft` command.

SYNTAX

```
integer report_scan_path  
[-view spec | existing_dft]  
[-chain chain_name | all]  
[-cell chain_name | all]  
[-test_mode list_of_mode_names | all]
```

Data Types

<i>chain_name</i>	string
<i>list_of_mode_names</i>	list

ARGUMENTS

- `-view spec | existing_dft`
Specifies the view from which to report. Valid options are **spec** and **existing_dft**. The **spec** view is used to report scan paths specified with the **set_scan_path** command. The **existing_dft** view shows scan paths already inserted by the **insert_dft** command. If not specified, the default is to use the **spec** view.
- `-chain chain_name | all`
Specifies the chain names to report scan-in, scan-out, scan-enable, scan-clock information. The default report shows only scan signal information. To see the scan cells use the **-cell** option. If **all** is specified, information on all scan chains is reported.
- `-cell chain_name | all`
Specifies chain names to report on scan cells and their order. By default the cells of the chains specified through the **-chain** option are not shown. With this option all cells in the chain are shown in the correct order. If **all** is specified, the scan cell order for all scan paths is reported.
- `-test_mode list_of_mode_names | all`
Specifies the test mode to report. If not specified, the tool reports on the current test mode that can be displayed by the **current_test_mode** command. If no test modes were created, the default is to use InternalScan. If **all** is specified, the tool displays the scan configurations of all of the test modes.

DESCRIPTION

This command displays scan paths specified by the **set_scan_path** command and the scan paths inserted by the **insert_dft** command.

The 3 report formats are as follows:

- Report on scan paths in the **spec** view contains information about non scan-in, scan-out, and scan-enable, with hook up pins if they exist.
- Report on scan paths in the **existing_dft** view contains information about scan paths defined by the **set_scan_path** command in the **existing_dft** view and on scan paths in the **existing_dft** view that are inserted by the **insert_dft** command.
- Report on scan cell order in scan paths in the **spec** view or the **existing_dft** view.

EXAMPLES

The following is an example of **report_scan_path** in the first format reporting on scan paths in the **spec** view. In this report, the *mychain1* scan path has SI (scan-in), Q (scan-out), and two scan enables SE1 and SE2, which are both hooked up to pin u5/A.

```
prompt> report_scan_path -view spec -chain mychain1 -test_mode mymodeA

*****
Report : Scan path
Design : A
Version: 2003.12
Date   : Thu Aug 14 10:46:10 2003
*****

=====
TEST MODE: mymodeA
VIEW      : Specification
=====

Scan_path      ScanDataIn (h)      ScanDataOut (h)      ScanEnable (h)
-----
mychain1       SI (-)              Q (-)               SE1 (u5/A)
                                   SE2 (u5/A)
```

The following is an example of **report_scan_path** in the second format reporting scan paths in the **existing_dft** view. In this report, the *mychain1* scan path is defined in the *mymodeA* test mode, and *mychain2* is defined in the *mymodeB* test mode. The *mychain2* scan chain has SI2 (scan-in), SO (scan-out), and SE2 & SE3 (scan-enable). Clocks MCLK1, MCLK2, and MCLK3 are master clocks for this scan chain, and SCLK is a slave clock.

```
prompt> report_scan_path -view existing_dft -chain all -test_mode all

*****
Report : Scan path
Design : A
Version: 2003.12
Date   : Thu Aug 14 10:46:10 2003
*****

=====
TEST MODE: mymodeA
```

```

VIEW      : Existing DFT
=====
Scan_path  Len  ScanDataIn  ScanDataOut  ScanEnable  MasterClock  SlaveClock
-----
mychain1   -    SI          Q            SE1         -            -

=====
TEST MODE: mymodeB
VIEW      : Existing DFT
=====
Scan_path  Len  ScanDataIn  ScanDataOut  ScanEnable  MasterClock  SlaveClock
-----
mychain2   5    SI2         SO           SE3         MCLK1        SCLK
                    SE2         MCLK2        -
                    -         MCLK3        -

```

The following is an example of **report_scan_path** in the third format reporting scan cells. In this report, 4 scan cells are reported from the *mychain1* scan chain:

```

prompt> report_scan_path -view existing_dft -cell mychain1 \
        -test_mode Internal_scan

```

```

*****
Report : Scan path
Design : top
Version: 2003.12
Date   : Thu Aug 14 10:46:29 2003
*****

=====
TEST MODE: Internal_scan
VIEW      : Existing DFT
=====
Scan_path  Cell_#  Instance_name  Clocks
-----
mychain1   0         r0/r0/q_reg1/q_reg
            1         r0/r0/q_reg2/q_reg
            2         r0/r1/q_reg1/q_reg
            3         r0/r1/q_reg2/q_reg

```

SEE ALSO

```

current_test_mode(2)
report_dft_configuration(2)
report_dft_signal(2)
report_scan_configuration(2)
report_scan_path(2)
set_dft_configuration(2)
set_dft_signal(2)
set_scan_configuration(2)
set_scan_path(2)

```

report_scan_register_type

Displays test-related information about the current design.

SYNTAX

integer **report_scan_register_type**

ARGUMENTS

None

DESCRIPTION

Displays the table of scan register types specified using the **set_scan_register_type** command.

EXAMPLES

The following is an example of the **report_scan_register_type** command:

```
prompt> report_scan_register_type

*****
Report : test
Design : level1
Version: A-2007.12
Date   : Thu Dec 28 17:46:57 2007
*****
Scan register on design level1 has '2' cells :
    FD3S
    FD4S
-exact false
```

SEE ALSO

current_design(2)
insert_dft(2)
set_scan_register_type(2)

report_scan_replacement

Displays the scan replacement table specified by the **set_scan_replacement** command.

SYNTAX

integer **report_scan_replacement**

ARGUMENTS

This command has no arguments.

DESCRIPTION

This command produces a report showing the complete scan replacement table specified using the **set_scan_replacement** command. The entries in this table are used by the **compile -scan**, **compile_ultra -scan**, and **insert_dft** commands to find scan equivalents for each instance of a non-scan flip-flop in the table.

To remove this table or entries in this table, use the **remove_scan_replacement** command.

EXAMPLES

The following is an example of **report_scan_replacement**:

```
prompt> report_scan_replacement

*****
Report : test
        -replacements
Design : level1
Version: A-2007.12
Date   : Thu Dec 28 18:18:38 2007
*****
Scan register on design level1 has '2' cells :
    FD3S
    FD4S
-exact false

Scan replacement table
Cell  LSSD  MUXD  CLOCKD_SCAN  AUX_CLOCKD_LSSD  COMB
-----
FD1   ____  FD1S  ____          ____          ____
1
```

SEE ALSO

current_design(2)

```
insert_dft(2)
remove_scan_replacement(2)
set_scan_replacement(2)
```


report_scan_state

Displays the scan state of the current design.

SYNTAX

```
status report_scan_state
```

ARGUMENTS

The **report_scan_state** command has no arguments.

DESCRIPTION

This command reports the current scan status description for the current design. The state of the design is either unknown, test_ready, or scan_existing.

EXAMPLES

The following example reports the scan state of the current design:

```
prompt> report_scan_state

*****
Report : test
        -state
Design : top
Version: V-2004.06-DFT
Date   : Thu Mar 25 17:42:10 2004
*****

Scan state           : scan cells replaced with loops
```

SEE ALSO

```
current_design(2)
insert_dft(2)
set_scan_state(2)
```

report_scan_suppress_toggling

Reports on the user specifications that were provided through the **set_scan_suppress_toggling** command in terms of the list of scan flip-flops to be gated by the **insert_dft** command.

SYNTAX

```
status report_scan_suppress_toggling
```

ARGUMENTS

The **report_scan_suppress_toggling** command has no arguments.

DESCRIPTION

The **report_scan_suppress_toggling** command reports on all of the existing specifications you provide through the **set_scan_suppress_toggling** command.

EXAMPLES

The following example shows how to issue the command:

```
prompt> report_scan_suppress_toggling
```

SEE ALSO

```
insert_dft(2)  
remove_scan_suppress_toggling(2)  
set_scan_suppress_toggling(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,

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.  
  
setup      : false  
hold       : true  
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.

The creation and usage of scenarios is available with Design Compiler Graphical.

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_supply_net

Reports all the supply nets in the current scope. This command is supported only in UPF mode.

SYNTAX

```
status report_supply_net  
[supply_net_name]  
[-include_exception]
```

Data Types

supply_net_name string list

ARGUMENTS

supply_net_name
 Specifies the supply nets that are to be reported. If the supply net does not exist in the current scope, the command fails. If the option is not specified, all the supply nets in the current scope are reported.

-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
[supply_port_name]
```

Data Types

supply_port_name string or list

ARGUMENTS

supply_port_name
Specifies the supply port or list of ports to be reported. If a specified supply port does not exist in the current scope, the command fails. If this argument is not specified, the tool reports on all supply ports in the current scope.

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_synlib

Displays information about synthetic libraries.

SYNTAX

```
int report_synlib  
library [module_list]
```

Data Types

```
library      string  
module_list  list
```

ARGUMENTS

library
Specifies the name of the library to be reported; this library must already be loaded into dc_shell. or must be in the **search_path**.

module_list
Specifies a list of modules about which information is to be reported. The default is to report information about all modules in the synthetic library.

DESCRIPTION

Displays information about synthetic libraries.

A synthetic library report displays, first, a list of all operators and their pins; next, a list of all modules with their pins, parameters, attributes, implementations and bindings; next, a list of all external implementations and external bindings; and finally, a list of DesignWare subblocks declared in the library.

By default, all modules are reported. If *module_list* is specified, only the listed modules are reported, along with any associated external implementations and external bindings. All implementations are annotated according to the attributes specified for them along with priority formulas and set ids.

Library objects that were added using the **update_lib** command are marked with an asterisk (*) following the name, to identify those objects added since the initial library was created with the **read_lib** command.

For a report to be generated, the specified library must be loaded into dc_shell. unless it is found in the **search_path**. The command **list -libraries** displays the libraries that are currently loaded.

EXAMPLES

The following command displays the **DW01_add** and **DW01_sub** modules and their implementations in the "standard.sldb" library. Notice at the end of the report that the implementations "super" and "fast" are marked with asterisks, indicating that

they were added using the **update_lib** command.

```
prompt> report_synlib standard.sldb {DW01_add DW01_sub}
```

```
*****
```

Report : library

Library: standard.sldb

Version: v3.1

Date : Thu Jan 28 11:46:00 1993

```
*****
```

Library Type : Synthetic

Tool Created : v3.1

Date Created : Oct. 29, 1992

Library Version : 3.0a

Synthetic Modules:

Module	Pins	Dir	Width	Parameters
DW01_add	A	in	width	
	B	in	width	
	CI	in	1	
	SUM	out	width	
	CO	out	1	
				width = max(width('A'), width(B))
DW01_sub	A	in	width	
	B	in	width	
	CI	in	1	
	DIFF	out	width	
	CO	out	1	
				width = max(width('A'), width('B'))

Module Attributes:

Attributes:

u - dont_use

d - design_library

Module	Attributes

DW01_add	d = DW01

DW01_sub	d = DW01

Module Implementations:

Attributes/Parameters:

v - verify_only

u - dont_use

r - regular_licenses

l - limited_licenses

d - design_library

report_synlib

1192

s - priority_set_id
p - priority
leg - legal

Module	Implementations	Attributes/Parameters
DW01_add	cla	
	rpl	
	sim	v
DW01_sub	cla	
	rpl	
	sim	v

Module Bindings:

Module	Bindings	Operators	Pin Associations	Constraints	Unbound Oper	P
DW01_add	b1	ADD_UNNS_OP	A, A B, B CI, "0" SUM, Z			
	b2	ADD_TC_OP	A, A B, B CI, "0" SUM, Z			
	b3	ADD_UNNS_OP	A, B B, A CI, "0" SUM, Z			
DW01_sub	b1	SUB_UNNS_OP	A, A B, B CI, "0" DIFF, Z			
	b2	SUB_TC_OP	A, A B, B CI, "0" DIFF, Z			

External Implementations:

Attributes/Parameters:
v - verify_only
u - dont_use
r - regular_licenses
l - limited_licenses

d - design_library
s - priority_set_id
p - priority
leg - legal

Module Implementations Attributes/Parameters -----
----- DW01_add fast * r = SynLib-ALU bar

DW01_sub super * l = SynLib-Eval p = "width > 8 ? 2 : 6"

SEE ALSO

compare_lib(2)
read_lib(2)
report_lib(2)
write_lib(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

```
compile(2)  
set_target_library_subset(2)  
remove_target_library_subset(2)  
check_target_library_subset(2)  
set_operating_conditions(2)  
target_library(3)
```

report_test_assume

Displays the value set on the pins by the **set_test_assume** command.

SYNTAX

```
status report_test_assume
```

ARGUMENTS

The **report_test_assume** command has no arguments.

DESCRIPTION

This command displays the pins and the value they are set to by the **set_test_assume** command on the current design. It reports the test assume value of either 1 or 0.

EXAMPLES

The following is an example of the **report_test_assume** command:

```
prompt> report_test_assume

*****
Report : test assume
Design : sub
Version: W-2004.12
Date   : Thu Mar 31 15:48:46 2005
*****

PIN NAME                                Test Assume Value
-----                                -
ff1/CD                                  0
ff6/Q                                   1
ff2/CD                                  1
```

SEE ALSO

set_test_assume(2)

report_test_model

Displays the test model information attached to a design.

SYNTAX

```
int report_test_model
[-design design_name]
```

Data Types

design_name string

ARGUMENTS

-design *design_name*
 Specifies the design on which to report test model information.

DESCRIPTION

The **report_test_model** command displays the model information about a design. Specify the design with the **-design** option. If no design is specified, the model information of the current design is displayed.

Test modes are modes of operation of the scan design. The model information contains the list of test modes associated with that model and their type. If the design has no corresponding modes or model, the "No test modes" message appears.

EXAMPLES

The following example generates a test model report for the *M1* design:

```
prompt> report_test_model -design M1

*****
Report : test model
Design : M1
Version: 2001.08
Date   : Fri Oct 19 09:07:32 2001
*****
```

Test Mode	Purpose	Spec	Impl	Model	Source
Internal_scan	InternalTest			Yes	
Mission_mode	Normal			Yes	
wrp_if	InternalTest			Yes	
wrp_of	ExternalTest			Yes	
wrp_safe	Isolate			Yes	

SEE ALSO

`current_design(2)`
`insert_dft(2)`

report_test_point_element

Displays options specified by set_test_point_element command.

SYNTAX

```
status report_test_point_element
[list_of_design_pin_objects]
[-type test_point_type]
```

Data Types

<i>list_of_design_pin_objects</i>	list
<i>test_point_type</i>	string

ARGUMENT

list_of_design_pin_objects
Specifies the list of design pin objects to which the **report_test_point_element** command applies.

-type test_point_type
Indicates the type to which the **report_test_point_element** command applies. You must specify one of the following values: **control_0**, **control_z0**, **control_1**, **control_z1**, **control_01**, **control_z01**, **force_0**, **force_z0**, **force_1**, **force_z1**, **force_01**, **force_z01**, or **observe**.

DESCRIPTION

This command displays the options specified by the **set_test_point_element** command on the current design.

EXAMPLES

The following is an examples of **report_test_point_element**:

```
prompt> report_test_point_element

*****
Report : User-defined test point element
Design : UReset
Version: W-2004.12
Date   : Tue Aug 17 17:21:49 2004
*****

=====
TEST MODE: all_dft
VIEW      : Specification
=====
Type:                                           observe
```

Configuration ID:	0
Pins that need a test point:	CurrState_reg_0/
Q CurrState_reg_1/Q	
Control signal:	MY_TM (TestMode)
Clock signal:	MY_CLOCK (MasterClock)
Test points per source or sink scan cell:	2
Scan for source or sink:	Enable
Power saving:	Enable

SEE ALSO

```
remove_test_point_element(2)
set_test_point_element(2)
```

report_testability_configuration

Displays options specified by the **set_testability_configuration** command.

SYNTAX

```
integer report_testability_configuration  
[-type control | observe | control_and_observe]
```

ARGUMENT

-type control | observe | control_and_observe
Indicates the type to which the report testability configuration command applies.

DESCRIPTION

This command displays the options specified by the **set_testability_configuration** command on the current design.

EXAMPLES

The following are examples of the report_testability_configuration.

```
prompt> report_testability_configuration  
  
*****  
Report : Testability configuration  
Design : A  
Version: V-2004.06  
Date   : Fri May  7 11:00:29 2004  
*****  
  
=====
```

TEST MODE: all_dft	
VIEW : Specification	
=====	
Testability type:	Control
Control signal	TMA (TestMode)
Clock signal:	CLKA (ScanMasterClock)
Clock type:	Dedicated
Maximum number of test points:	500
Maximum number of test points per scan register:	6
Testability type:	Observe
Control signal	TMB (TestMode)
Maximum number of test points per scan register:	8
Clock type:	Dominant
Power saving:	Enable
Maximum additional logic area (%)	1

SEE ALSO

```
reset_testability_configuration(2)  
set_testability_configuration(2)
```

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

```

compile(2)
set_attribute(2)
set_max_leakage_power(2)

```


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. The option is disabled in Design Compiler.

-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
```

Creation and usage of scenarios is available with Design Compiler Graphical.

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 (A07)	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 (A07)	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
cell124/n22 (net)	0.00	0.54 f
u0/A (NR2)	0.00	0.54 f
u0/Z (NR2)	1.20	1.74 r
cell124/n21 (net)	0.00	1.74 r
u8/A (IVA)	0.00	1.74 r
u8/Z (IVA)	0.43	2.17 f
cell124/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)
set_wire_load_min_block_size(2)
set_wire_load_mode(2)
set_wire_load_model(2)
set_wire_load_selection_group(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. This command is supported only in topographical mode.

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_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_use_test_model

Reports test model usage for each subdesign under the current design.

SYNTAX

status **report_use_test_model**

ARGUMENTS

The **report_use_test_model** command has no arguments.

DESCRIPTION

Displays the list of sub-designs currently in dc_shell that are instantiated within the current design. For each sub-design, it indicates whether a test model or gates are used.

EXAMPLES

SEE ALSO

use_test_model(2)

report_voltage_area

Reports the voltage areas in the design. This command is supported only in topographical mode.

SYNTAX

```
int report_voltage_area  
-all | patterns  
[-name list]  
[-verbose]  
[-nosplit]
```

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
*****
Voltage area name va1
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)
remove_voltage_area(2)
```

report_wire_load

Displays the characteristics of the wire load models set on a design or in a library.

SYNTAX

```
int report_wire_load  
[-design design_name]  
[-name model_name]  
[-libraries]  
[-nosplit]
```

Data Types

<i>design_name</i>	string
<i>model_name</i>	string

ARGUMENTS

-design *design_name*
Specifies the design name for which to report wire load models. The default is to report wire load models set on the current design.

-name *model_name*
Specifies the name of the wire load model to report. By default, the tool reports all of the wire load models set on the specified design.

-libraries
Specifies to also report on wire load models in the libraries used to link the current design.

-nosplit
Prevents line-splitting and facilitates writing software to extract information from the report output. Most of the wire load model 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_wire_load** command displays characteristics of wire load models set on a design, a cluster, or stored in a library. Wire load models are set on a design or cluster with the **set_wire_load_model** command. Wire loads models are created for designs and clusters with the **create_wire_load** command.

The wire load report displays where the model is stored under the Location field. This can be the current design, one of its subdesigns or clusters, or a library linking the current design. The Location is a design, subdesign, or cluster if the model was created with **create_wire_load**, otherwise the Location is a library.

If the wire load model is in the wire_load format, a wire load report displays the

coefficients to calculate net resistance, capacitance, and area from the net length. The report displays the statistics on how the model was generated. Points indicates how many nets were used to calculate the average capacitance. Standard Deviation is the standard deviation indicating the accuracy of the length and average capacitance. A standard deviation close to zero indicates that all data points reside very close to the average. The standard deviation is the average delta between the model's capacitance for a fanout and each annotated capacitance for the same fanout. The weighted standard deviation is the average percentage of all standard deviation weighted with the number of points for each fanout. This measure characterizes the accuracy of the model created. The wire_load format is generated with the **create_wire_load** command when the **compile_create_wire_load_tablevariable** is set to **true**.

If **create_wire_load** is used to generate the wire load, and the **compile_create_wire_load_table** variable is set to **true**, the report for the wire load table does not have the statistics information. The report has the length, capacitance, resistance, and area value listed for each fanout.

EXAMPLES

The following is an example of a wire load report for the current design. The design has the wire_load format model. The **compile_create_wire_load_table** variable was set to **false** when the **create_wire_load** command was executed.

```
prompt> report_wire_load
```

```
*****
```

```
Report : wire loads
```

```
Design : counter
```

```
Version: v3.1-development
```

```
Date   : Mon Jan 25 10:47:24 1993
```

```
*****
```

```
Wire load model: counter_wl
```

```
Location      : counter
```

```
Resistance    : 0
```

```
Capacitance   : 1
```

```
Area         : 0
```

```
Slope        : 1.333
```

Fanout	Length	Points	Average Cap	Standard Deviation	% Standard Deviation
1	0.58	1565	0.58	0.01	0.99
2	1.46	168	1.46	0.04	2.78
3	2.40	64	2.40	0.12	4.80
4	3.33	56	3.33	0.15	4.59
5	4.03	30	4.03	0.24	5.88
6	5.31	23	5.31	0.35	6.58
7	6.58	14	6.58	0.78	11.87
8	7.86	11	7.86	0.94	11.91
9	14.80	8	14.80	2.97	20.07
10	18.93	2	18.93	9.93	52.47
11	23.06	3	23.06	8.75	37.94
12	27.19	1	27.19	12.85	47.28

13	31.32	4	31.32	6.82	21.77
16	31.74	6	31.74	5.84	18.40
17	32.89	16	32.89	3.16	9.62
18	34.04	6	34.04	4.51	13.26
19	35.18	6	35.18	3.67	10.43

Weighted Average Standard Deviation: 2.08

The following is an example of a wire load report for the current design. The design has the `wire_load_table` format model. The **`compile_create_wire_load_table`** variable was set to **`true`** when the **`create_wire_load`** command was executed.

```
prompt> report_wire_load
```

```
*****
Report : wire loads
Design : TOP
Version: 1997.01-SI1
Date   : Mon Sep  9 17:59:34 1996
*****
```

```
Wire load model:  U3/U7_wl
Location         :  CONVERTOR_1 (design)
```

Fanout	Length	Capacitance	Resistance	Area
1	0.28	0.28	.06	0
2	0.52	0.52	.09	0
3	0.71	0.71	.11	0

SEE ALSO

```
set_wire_load_mode(2)
set_wire_load_model(2)
set_wire_load_min_block_size(2)
set_wire_load_selection_group(2)
report_design(2)
```

report_wrapper_configuration

Reports the wrapper configuration of the current design.

SYNTAX

```
status report_wrapper_configuration
```

ARGUMENTS

The **report_wrapper_configuration** command has no arguments.

DESCRIPTION

This command reports the wrapper configuration in the following format:

Wrapper Structures	Status
-----	-----
Class:	<class_name>
Style:	<style_name>
Dedicated Cell Type:	<cell_type>
Shared Cell Type:	<cell_type>
Dedicated Cell Design Name:	<design_name>
Shared Cell Design Name:	<design_name>
Register IO Implementation:	<impl_name>
Use Dedicated Wrapper Clock:	<boolean>
Safe State:	<state>
Delay Test:	<boolean>

EXAMPLES

The following example reports the wrapper configuration with a dedicated cell type WC_D1_S and safe state 1:

```
prompt> set_wrapper_configuration class core_wrapper \  
        -dedicated_cell_type WC_D1_S -safe_state 1
```

1

```
prompt> report_wrapper_configuration  
*****  
Report : Wrapper Configuration  
Design : M1  
Version: W-2004.12  
Date   : Thu Sep 16 09:46:34 2004  
*****
```

Wrapper Structures	Status
-----	-----
Class:	Core Wrapper
Style:	Dedicated Wrapper

Dedicated Cell Type:	WC_D1_S
Shared Cell Type:	WC_S1
Dedicated Cell Design Name:	
Shared Cell Design Name:	
Register IO Implementation:	Swap
Use Dedicated Wrapper Clock:	False
Safe State:	1
Delay Test:	False

1

SEE ALSO

`insert_dft(2)`
`preview_dft(2)`
`reset_wrapper_configuration(2)`
`set_wrapper_configuration(2)`

reset_autofix_configuration

Resets the autofix configuration on a type basis for the current design.

SYNTAX

```
status reset_autofix_configuration  
[-type autofix_type]
```

Data Types

autofix_type string

ARGUMENTS

-type autofix_type
Indicates the type to which the reset autofix configuration command applies.
You must specify one of the following values: **clock**, **set**, **reset**,
xpropagation, **internal_bus**, **external_bus**, or **bidirectional**.

DESCRIPTION

The **reset_autofix_configuration** command resets the current autofix configuration for a particular type from the design.

EXAMPLES

The following example resets the autofix configuration for the clock type:

```
prompt> reset_autofix_configuration -type clock
```

SEE ALSO

```
insert_dft(2)  
report_autofix_configuration(2)  
set_autofix_configuration(2)
```

reset_autofix_element

Resets the autofix configuration for a particular type and a list of design objects for the current design.

SYNTAX

```
int reset_autofix_element  
list_of_design_objects  
[-type clock | set | reset | xpropagation | internal_bus | external_bus]
```

Data Types

list_of_design_objects list

ARGUMENTS

list_of_design_objects
 Specifies the list of design objects to which the configuration applies.

-type clock | set | reset | xpropagation | internal_bus | external_bus
 Indicates which type the reset autofix element command applies to.

DESCRIPTION

Use the **reset_autofix_element** command to reset the current autofix configuration from the design for a particular type and a list of design objects.

EXAMPLES

The following example shows how to reset the autofix configuration for the type reset and for the cells u0 and u1

```
prompt> reset_autofix_configuration [list u0 u1] -type reset
```

SEE ALSO

```
set_autofix_element(2)  
report_autofix_element(2)  
insert_dft(2)
```

reset_bsd_configuration

Removes the IEEE 1149.1 specifications from a boundary-scan design.

SYNTAX

status **reset_bsd_configuration**

ARGUMENTS

The **reset_bsd_configuration** command has no arguments.

DESCRIPTION

The **reset_bsd_configuration** command removes the IEEE 1149.1 specifications from the current design.

EXAMPLES

The following is an example of the **reset_bsd_configuration** command:

```
prompt> reset_bsd_configuration
```

SEE ALSO

```
define_dft_design(2)  
set_boundary_cell(2)  
set_bsd_compliance(2)  
set_bsd_instruction(2)  
set_bsd_linkage_port(2)  
set_scan_path(2)
```

reset_clock_gate_latency

Resets all clock latency values previously specified for or applied to clock gating cells.

SYNTAX

```
int reset_clock_gate_latency  
[-clock clock_list]
```

Data Types

clock_list list

ARGUMENTS

-clock *clock_list*
Indicates that only latency values with respect to the specified clocks must be removed. If **clock_list** is not specified, latency is removed from all clocks that are currently defined.

DESCRIPTION

This command resets all clock latency values previously specified for or applied to clock gating cells. Following two effects are produced by issuing this command:

- It removes the attributes from the source ports of clocks where some clock latency values may have been previously specified by using **set_clock_gate_latency** command.
- It completely removes all clock latency information from clock gating cells which already have annotated latency values. In this case it does not differentiate between latencies set using **set_clock_latency** command and those specified using **set_clock_gate_latency** command.

SEE ALSO

```
set_clock_gate_latency(2)  
apply_clock_gate_latency(2)  
report_clock_gating(2)  
set_clock_latency(2)  
remove_clock_latency(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_dft_configuration

Resets the DFT configuration for the current design specified by the `set_dft_configuration` command.

SYNTAX

integer **reset_dft_configuration**

ARGUMENTS

The **reset_dft_configuration** command has no arguments.

DESCRIPTION

The **reset_dft_configuration** command resets the DFT configuration for the current design.

EXAMPLES

In the following example, the command resets the current design to the default value:

```
prompt> reset_dft_configuration
```

SEE ALSO

```
insert_dft(2)  
report_dft_configuration(2)  
set_dft_configuration(2)
```

reset_dft_drc_rules

Resets the severity to the default value on the allowed set of violations.

SYNTAX

```
integer reset_dft_drc_rules  
[-violation drc_error_ID]  
[-cell cell_list]
```

Data Types

<i>drc_error_ID</i>	string
<i>cell_list</i>	string

ARGUMENTS

-violation *drc_error_ID*
Specifies explicitly to change the severity of the violation to the default value.

-cell *cell_list*
Specifies the cells for which the violation severity is to be changed to the default value. If no cells are specified, the violation severity is changed for all the cells in the design.

DESCRIPTION

Use the **reset_dft_drc_rules** command to change the severity of the violations reported by **dft_drc** to their default values. You can use the **reset_dft_drc_rules** command to change the severity only if it has been set by using the **set_dft_drc_rules** command. A list of valid violations is shown below.

```
DRC violation ID  
-----  
TEST-504  
TEST-505
```

Use the **report_dft_drc_rules** command to display the violations whose severity has been changed. To set the severity of the violation, use the **set_dft_drc_rules** command.

EXAMPLES

The following example changes the severity of the DRC violation TEST-504 for all cells and TEST-505 for cells reg3 and reg4 to the default ERROR.

```
prompt> reset_dft_drc_rules -violation {TEST-504}  
prompt> reset_dft_drc_rules -violation {TEST-505} -cell {reg3 reg4}
```

SEE ALSO

dft_drc(2)
insert_dft(2)
report_dft_drc_rules(2)
set_dft_drc_rules(2)

reset_dft_insertion_configuration

Resets the DFT insertion configuration for the current design.

SYNTAX

```
int reset_dft_insertion_configuration
```

DESCRIPTION

Resets the existing insertion configuration to its defaults. The map effort is by default medium, synthesis optimization is all, route scan enable is TRUE, route_scan_clock is TRUE, route_scan_serial is TRUE, preserve_design_name is FALSE, unscan option is FALSE. Use the **report_dft_insertion_configuration** command to display the current DFT insertion configuration of the design.

Use the **reset_dft_insertion_configuration** command to remove the current DFT insert configuration from the design.

EXAMPLES

The following example shows how to reset the insert dft configuration

```
prompt> reset_dft_insertion_configuration
```

SEE ALSO

```
set_dft_insertion_configuration(2)  
report_dft_insertion_configuration(2)  
insert_dft(2)
```

reset_logicbist_configuration

Resets the current design's logic BIST configuration previously defined by the **set_logicbist_configuration** command.

SYNTAX

```
integer reset_logicbist_configuration  
[-test_mode test_mode_names]
```

DESCRIPTION

Use the **reset_logicbist_configuration** command to reset the logic BIST configuration for the current design. The command resets the logic BIST configuration to its default values.

EXAMPLES

The following example shows how to reset the current design's logic BIST configuration to the default value.

```
prompt> reset_logicbist_configuration
```

SEE ALSO

```
insert_dft(2)  
report_logicbist_configuration(2)  
set_logicbist_configuration(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  
1252
```


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_physical_constraints

Resets all current physical constraints.

SYNTAX

integer **reset_physical_constraints**

ARGUMENTS

This command has no arguments.

DESCRIPTION

The **reset_physical_constraints** command resets the physical constraints settings that were previously applied to the design. After this command is executed, the netlist does not have any physical information.

EXAMPLES

The following command resets the physical constraints:

```
prompt> reset_physical_constraints
```

SEE ALSO

```
extract_physical_constraints(2)  
report_physical_constraints(2)  
write_physical_constraints(2)
```

reset_pipeline_scan_data_configuration

Resets the pipeline scan data configuration specified by the **set_pipeline_scan_data_configuration** command.

SYNTAX

integer **reset_pipeline_scan_data_configuration**

ARGUMENTS

This command has no arguments.

DESCRIPTION

Use the **reset_pipeline_scan_data_configuration** command to reset the pipeline scan data configuration for the current design.

EXAMPLES

The following example resets the pipeline scan data configuration of the current design to the default value:

```
prompt> reset_pipeline_scan_data_configuration
```

SEE ALSO

set_pipeline_scan_data_configuration(2)

reset_scan_compression_configuration

Resets the scan compression configuration for the current design.

SYNTAX

```
int reset_scan_compression_configuration
```

ARGUMENT

The **reset_scan_compression_configuration** command has no arguments.

DESCRIPTION

Use the **reset_scan_compression_configuration** command to reset the current scan compression configuration on the design.

EXAMPLES

```
prompt> reset_scan_compression_configuration  
Accepted scan compression specification for design 'des_unit'.  
1
```

SEE ALSO

```
set_scan_compression_configuration(2)  
report_scan_compression_configuration(2)  
insert_dft(2)  
preview_dft(2)
```


reset_scan_configuration

Resets the scan configuration for the current design.

SYNTAX

```
int reset_scan_configuration  
[-test_mode mode_name]
```

Data Types

mode_name string

ARGUMENT

`-test_mode mode_name`
Indicates which test mode the reset scan configuration command applies to.

DESCRIPTION

Use the **reset_scan_configuration** command to reset the current scan configuration from the design.

EXAMPLES

The following example shows how to reset the scan configuration

```
prompt> reset_scan_configuration
```

SEE ALSO

```
set_scan_configuration(2)  
report_scan_configuration(2)  
insert_dft(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
```

```
reset_switching_activity  
1262
```

```
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_test_mode

Resets the test mode for the current design.

SYNTAX

```
status reset_test_mode
```

ARGUMENTS

The **reset_test_mode** command has no arguments.

DESCRIPTION

The **reset_test_mode** command can be used to reset the current test mode of the current design to the default test mode.

EXAMPLES

The following example resets the current test mode:

```
prompt> reset_test_mode  
Current test mode is reset to default  
1
```

SEE ALSO

```
current_test_mode(2)  
define_test_mode(2)  
remove_test_mode(2)
```

reset_testability_configuration

Resets the testability configuration on a type basis for the current design.

SYNTAX

```
status reset_testability_configuration  
[-type control | observe | control_and_observe]
```

ARGUMENTS

```
-type control | observe | control_and_observe  
    Indicates the type to which the reset_testability_configuration applies.
```

DESCRIPTION

The **reset_testability_configuration** command resets the current testability configuration for a particular type from the design.

EXAMPLES

The following example shows how to reset the testability configuration for the **observe** type.

```
prompt> reset_testability_configuration -type observe
```

SEE ALSO

```
insert_dft(2)  
report_testability_configuration(2)  
set_testability_configuration(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_wrapper_configuration

Resets the wrapper configuration for the current design.

SYNTAX

integer **reset_wrapper_configuration**

ARGUMENTS

The **reset_wrapper_configuration** command has no arguments.

DESCRIPTION

This command resets the wrapper configuration with the following values:

Wrapper Structures	Status
-----	-----
Class:	Core Wrapper
Style:	Dedicated Wrapper
Dedicated Cell Type:	WC_D1
Shared Cell Type:	WC_S1
Dedicated Cell Design Name:	
Shared Cell Design Name:	
Register IO Implementation:	Swap
Use Dedicated Wrapper Clock:	False
Safe State:	None
Delay Test:	False

EXAMPLES

The following example resets the wrapper configuration for the current design. A report command follows that shows the wrapper configuration values after the reset.

```
prompt> reset_wrapper_configuration
1
prompt> report_wrapper_configuration
*****
Report : Wrapper Configuration
Design : M1
Version: A-2007.12
Date   : Fri Dec 28 09:46:28 2007
*****
```

Wrapper Structures	Status
-----	-----
Class:	Core Wrapper
Style:	Dedicated Wrapper
Dedicated Cell Type:	WC_D1
Shared Cell Type:	WC_S1
Dedicated Cell Design Name:	
Shared Cell Design Name:	

Register IO Implementation:	Swap
Use Dedicated Wrapper Clock:	False
Safe State:	None
Delay Test:	False

1

SEE ALSO

`insert_dft(2)`
`preview_dft(2)`
`report_wrapper_configuration(2)`
`set_wrapper_configuration(2)`

rewire_clock_gating

Changes the clock-gating cell implemented by the tool for a particular gated cell.

SYNTAX

```
status rewire_clock_gating  
[-gating_cell new_clock_gating_cell]  
[-gated_objects gated_objects_list]  
[-balance_fanout]  
[-undo]  
[-verbose]
```

Data Types

<i>new_clock_gating_cell</i>	string
<i>gated_objects_list</i>	list

ARGUMENTS

-gating_cell *new_clock_gating_cell*
Specifies a new clock-gating cell to be used to gate the cells or pins on the *gated_objects_list*.

-gated_objects *gated_objects_list*
Specifies that all registers, clock gates, and clock pins of gated modules in the *gated_objects_list* are gated by the *new_clock_gating_cell*.

-balance_fanout
Specifies that the clock gates and registers must be rewired to meet the minimum and maximum fanout constraints specified with the **set_clock_gating_style** command.

-undo
Indicates that directives specified by previously-executed **rewire_clock_gating** commands are to be removed.

-verbose
Specifies that additional information is to be displayed as the command executes.

DESCRIPTION

This command changes the clock-gating cell for the specified gated cell. The cell can be a register, a clock gate, or a gated module.

The tool performs clock gating at the RTL level with the use of the **compile_ultra -gate_clock** command. If you use the **set_clock_gating_style** command with the **-max_fanout** option, the tool limits the fanout of individual clock-gating elements created by **compile_ultra -gate_clock**. The tool duplicates clock-gating logic to ensure that the fanout of each element is bound by the constraint specified with the **-max_fanout** option. The clock-gating cells created to satisfy the **-max_fanout**

constraint are logically equivalent. The **rewire_clock_gating** command directs the tool to move a clock-gated cell from one clock-gating cell to another logically-equivalent clock-gating cell.

The **rewire_clock_gating** command directs the tool to perform the actual circuit modifications to rewire clock gating in the design. You can undo the effect of any prior **rewire_clock_gating** command by using the **-undo** option. The **-undo** option deletes the directives previously specified by the **rewire_clock_gating** command.

In some cases, the tool could delete or add registers. This might alter the balanced fanout of clock gates. When invoked with the **-balance_fanout** option, **rewire_clock_gating** attempts to reinstate the clock gate fanout limits by splitting, merging or rewiring. When using this option, running **compile -incremental** might not be necessary. Under some specific circumstances, a few GTECH cells might be introduced. In those case, the **rewire_clock_gating** command prints a message that a **compile -incremental** is necessary. When the command is going to be invoked from a script (non-interactive mode) it is best to add **compile -incremental** after **rewire_clock_gating -balance_fanout**. This option can also be used if you require the clock-gate fanout limits to be changed after the **compile_ultra -gate_clock** command. It is recommended to invoke this command after the **compile** command. Note that this command has no effect on multilevel and module clock gates.

Because rewiring gated registers alters the connections of the clock-gating cell that gates the registers, any path-based timing exception that passes through the old clock-gating cell to a gated register is no longer relevant and is lost.

If either the *new_clock_gating_cell* or any of the *gated_register_list* is marked for removal of clock gating with the **remove_clock_gating** command, the rewiring directive is not honored.

EXAMPLES

The following example provides a directive to change the clock-gating cell gating the *pipeline_reg_A[0]* and *pipeline_reg_A[3]* registers to the *clk_gate_pipeline_reg_A_0* clock-gating cell:

```
prompt> rewire_clock_gating -gating_cell \  
        clk_gate_pipeline_reg_A_0 -gated_registers \  
        {pipeline_reg_A[0], pipeline_reg_A[3]}
```

The following example provides a directive to remove the previous directive and to change the clock-gating cell gating the *pipeline_reg_A[0]* and *pipeline_reg_A[3]* registers:

```
prompt> rewire_clock_gating -undo \  
        -gated_registers {pipeline_reg_A[0] pipeline_reg_A[3]}
```

The following example directs the **rewire_clock_gating** command to change the minimum and maximum fanout limits to new values:

```
prompt> set_clock_gating_style -min 2 -max 128
```

```
prompt> rewire_clock_gating -balance_fanout
```

SEE ALSO

```
compile(2)  
compile_ultra(2)  
remove_clock_gating(2)  
report_clock_gating(2)  
set_false_path(2)  
set_max_delay(2)  
set_min_delay(2)  
set_multicycle_path(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
```

```
rp_group_inclusions  
1272
```

```
{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)
```

rtl2saif

Creates a SAIF forward-annotation file starting from the top level of the design.

The rtl2saif command will be obsolete in a future release as RTL forward SAIF is now obsolete.

SYNTAX

```
int rtl2saif  
[-output file_name]  
[-design design_name]
```

Data Types

<i>file_name</i>	string
<i>design_name</i>	string

ARGUMENTS

-output *file_name*
Specifies the name of the file to which the forward-annotation SAIF file created by **rtl2saif** is to be written; the file must be writable. The default file name is *power_rtl.saif*, which can be changed by the **power_rtl_saif_file** variable.

-design *design_name*
Specifies the name of the design for which the forward-annotation SAIF file is to be created. The default is the current design. You must link all designs before calling **rtl2saif**, because **rtl2saif** traverses the subdesigns within all hierarchies.

DESCRIPTION

The rtl2saif command will be obsolete in a future release as RTL forward SAIF is now obsolete.

Creates a SAIF forward-annotation file starting from the top hierarchy of the design. The SAIF forward-annotation file created contains synthesis invariant information, which is needed to capture switching activity at the RTL level. The command traverses into the subdesigns within all hierarchies.

This command works only with a pre-mapped design netlist.

The dc_shell variable **power_preserve_rtl_hier_names** must be set to *true* before analyzing and elaborating the designs.

For details about SAIF, refer to the SAIF specification.

For details about capturing RTL switching activity, refer to the *Power Compiler Reference Manual*.

EXAMPLE

The following example shows how to use the command.

```
prompt> rtl2saif -out forward_annot.saif
```

SEE ALSO

`power_preserve_rtl_hier_names(3)`

saif_map

Manages the SAIF name mapping mechanism for reading SAIF files.

SYNTAX

```
string saif_map
[-start]
[-end]
[-reset]
[-report]
[-get_name]
[-set_name names]
[-add_name names]
[-remove_name names]
[-clear_name]
[-get_object_names name]
[-create_map]
[-write_map filename]
[-read_map filename]
[-type type]
[-inverted]
[-instances objects]
[-hierarchical]
[-no_hierarchical]
[-columns columns]
[-sort columns]
[-rtl_summary]
[-missing_rtl]
[-input SAIF_file]
[-source_instance SAIF_instance_name]
[-target_instance target_instance_name]
[-review]
[-preview]
[-hsep character]
[object_list]
[-verbose]
[-non_verbose]
[-nosplit]
```

Data Types

<i>names</i>	string
<i>name</i>	string
<i>filename</i>	string
<i>objects</i>	list
<i>columns</i>	string
<i>SAIF_instance_name</i>	string
<i>target_instance_name</i>	cell
<i>character</i>	string
<i>object_list</i>	list

ARGUMENTS

-start

Initializes the name mapping database. The features of the name mapping mechanism are not available unless it is initialized. Use **saif_map -start** before reading RTL source files when you need to use the SAIF name mapping mechanism.

-end

Uninitialises the name mapping database.

-reset

Resets the name mapping information by clearing all SAIF names from the name mapping database and sets all the original names to an initial value depending on the object's name and scope.

-report

Outputs a report of the name mapping information. By default, the report lists the design objects and their SAIF names. You can use the **-columns** option to include more information, such as the object's original name. You can use the **-sort** option to specify sorting criteria. By default, all the objects in the current instance hierarchy are considered for the report; otherwise, you can specify a list of objects explicitly as the *object_list* argument or implicitly by using the **-instances**, **-hierarchical**, and **-no_hierarchical** options.

-get_name

Returns the name of the specified object in the name mapping database. The name returned depends on the value of the optional **-type** option. If the **-type** option is "saif_names" (default value), a list of SAIF names on that object is returned. If the **-type** option is "orig_name", the original name is returned. If the **-type** option is "extended_orig_name", the original name is returned in extended syntax.

-set_name *names*

Sets the name of the specified object in the name mapping database. The name to be set depends on the value of the optional **-type** option. If the **-type** option is "saif_names" (default value), the SAIF names on that object are set. If the **-type** option is "orig_name", the original name is set. If the **-type** option is "extended_orig_name", the original name is set and the *names* argument is expected to be in extended syntax.

-add_name *names*

Adds the specified SAIF names to the specified object in the name mapping database.

-remove_name *names*

Removes the specified SAIF names from the specified object in the name mapping database.

-clear_name

Clears the names of the specified object in the name mapping database. The name that is cleared depends on the value of the optional **-type** option. If the **-type** option is "saif_names" (default value), the SAIF names on that object are cleared. If the **-type** option is "orig_name" or

"extended_orig_name", the original name is cleared. If the **-type** option is "all_names", both the SAIF names and the original name are cleared. You can specify the objects explicitly as the *object_list* argument or implicitly by using the optional **-instances**, **-hierarchical**, and **-no_hierarchical** options. If you do not specify the **-instances** option, the current instance is used.

-get_object_names *name*

Returns the object names of the design objects that have the specified name in the name mapping database. The name used for finding the design objects in the name mapping database depends on the value of the **-type** option. If the **-type** option is "saif_names" (default value), the *name* argument is assumed to be a SAIF name. If the value of the **-type** option is "orig_name", the *name* argument is assumed to be an original name. If the value of the **-type** option is "extended_orig_name", the *name* argument is assumed to be an original name in extended syntax.

-create_map

Updates the name mapping database automatically with new SAIF names by reading a SAIF file and matching design objects with the names in the SAIF file. The SAIF file must be specified by using the **-input** option, and the instance name of the current design in the SAIF file must be specified by using the **-source_instance** option.

-write_map *filename*

Outputs the name mapping database into the specified name mapping file. You can read the name mapping file back into Power Compiler by using the **-read_map** option of **saif_map**. If the **-type ptpx** option is used, the SAIF names in the name mapping database are output into a file with a list of **set_rtl_to_gate_name** commands that can be read by PrimeTime PX.

-read_map *filename*

Reads the name mapping information stored in a name mapping file generated using the **-write_map** option of **saif_map**.

-type *type*

Specifies an optional type argument with some of the **saif_map** command features. When used with the **-get_name**, **-set_name**, or **-get_object_names** options, the following values are the valid type keywords:

- * **saif_names** (default)
Specifies that SAIF names are used.
- * **orig_name**
Specifies that original names are used.
- * **extended_orig_name**
Specifies that original names in extended syntax format are used.

In addition to these values, you can use the **all_names** keyword with the **-clear_name** option to specify that both SAIF names and original names are cleared.

When used with the **-write_map** option, the following values are the valid type keywords:

- * **name_map** (default)
Specifies that a name mapping file is created.
- * **ptpx**
Specifies that a set of PrimeTime PX instructions is generated.

-inverted

Specifies that the name to be set maps to a logically inverted object. This option is valid only with the **-set_name** option. For example, you can use the **-inverted** option when specifying that the object A_reg/QN maps to the SAIF name "A" but its logical value is inverted. However, note that for such a simple example, it is generally sufficient to specify that the register cell A_reg maps to the SAIF name "A", and the fact that A_reg/QN is logically inverted to "A" is deduced automatically when the SAIF file is read.

-instances *objects*

Restricts the effect of some of the **saif_map** command features to the specified instances. A number of **saif_map** options (including **-report** and **-clear**) act on a number of objects that can be specified explicitly as the *object_list* argument or implicitly by using the optional **-instances**, **-hierarchical**, or **-no_hierarchical** options. The *objects* argument is a list of hierarchical cells, and the default value is the current instance. If you do not explicitly specify the list of objects, all the objects in the current instance are used (if you use **-no_hierarchical**, only the top-level objects are used).

-hierarchical

Specifies that all objects in the hierarchy under the current instance or the instances specified with the **-instance** option are used by the **saif_map** command. A number of **saif_map** options (including **-report** and **-clear**) act on a number of objects that can be specified explicitly as the *object_list* argument or implicitly by using the optional **-instances**, **-hierarchical**, or **-no_hierarchical** options. When selecting objects implicitly, the **-hierarchical** flag is assumed by default, so it is never required. It is included as an option because it is a common option in the shell commands.

-no_hierarchical

Specifies that all objects in the top-level of the current instance or the instances specified with the **-instance** option are used by the **saif_map** command. A number of **saif_map** options (including **-report** and **-clear**) act on a number of objects that can be specified explicitly as the *object_list* argument or implicitly by using the optional **-instances**, **-hierarchical**, or **-no_hierarchical** options.

-columns *columns*

Specifies a list of column names that are used when reporting the name mapping database information. The possible values of the elements in the *columns* argument are

- * **type**

Specifies the type (pin, port, cell, or net) of the object.

- * **object_names**

Specifies the name of the design object.

- * **saif_names**

Specifies that SAIF names are included in the report.

- * **orig_names**

Specifies that original names are included in the report.

- * **extended_orig_names**

Specifies that original names in extended format are included in the report.

- * **attributes**

Specifies that attributes, or flags, are included in the report.

The default value of *columns* is {type object_names saif_names attributes}.

-sort *columns*

Specifies the sorting criteria that is used when reporting the name mapping database information. The *columnsP* argument is a list of column names as specified under the **-columns** description above.

-rtl_summary

Specifies that a summary report of which RTL or synthesis invariant objects have SAIF name mapping information. This option is valid only when you also use the **-report** option. The report contains the number of synthesis invariant design objects that have SAIF name information obtained automatically or set by the user. It also contains the number of design objects that do not have the SAIF name information annotated directly on them, but annotated to some of the object's connecting nets, pins, or ports. The synthesis invariant objects included in the report are: design ports, hierarchical cell pins, sequential cells, and tristate cells.

-missing_rtl

Displays a list of RTL or synthesis invariant objects that do not have SAIF name mapping information. This option is valid only when you also use the **-report** and **-rtl_summary** options.

-input *SAIF_file*

Specifies the name of the SAIF file that is used to create the name mapping information when used with the **-create_map** option.

-source_instance *SAIF_instance_name*

Specifies the name of the current design instance in the SAIF file that is used to create the name mapping information when used with the **-create_map** option.

-target_instance *target_instance_name*

Specifies the instance in the current design for which the name mapping information is created when used with the **-create_map** option. The default target instance is the current instance.

-review

Displays a SAIF match report showing which design objects are matched with which SAIF file objects after the SAIF name mapping information is created from a SAIF file. This option is valid only with the **-create_map** option.

-preview

Displays a SAIF match report without creating the actual SAIF name mapping. The report shows which design objects would be matched with which SAIF file objects when the name mapping is created. This option is valid only used with the **-create_map** option.

-hsep *character*

Specifies the hierarchical separator when parsing or reporting SAIF names and original names. The default value is "/". Other possible values are "." and " ". The **-hsep " "** argument can be useful when the "/" character is part of the identifier names of the design objects.

object_list

Specifies the list of objects used by the **saif_map** command.

-verbose

Specifies that more verbose output messages are used. You can use this option with other **saif_map** options or on its own to turn on verbose messaging for all future calls of the **saif_map** command.

-non_verbose

Specifies that verbose output messages are not used. You can use this option with other **saif_map** options or on its own to turn off verbose messaging for all future calls of the **saif_map** command.

DESCRIPTION

The **saif_map** command manages and uses a SAIF name mapping database that provides a mapping between design objects and the names in SAIF files. The name mapping database can be created automatically and you can query and modify the information in the database. You can use the name mapping database when reading SAIF files generated by RTL simulation, where the names in the SAIF file do not exactly match the names of the design objects.

The information stored in the name mapping database consists of the following possible information on design ports, nets, pins and cells:

1. A list of SAIF names, representing the possible names in the SAIF file that match the design object.
2. An original name, representing the original name of the design object as created by the synthesis front-end tool (Presto).
3. A number of flags that specify, for example, whether the SAIF and original names have been created automatically or set by the user.

When the name mapping mechanism is active (the **saif_map -start** command activates the name mapping mechanism), the original names are automatically created when the design RTL source is read and linked. The SAIF names and original names are full instance names.

The SAIF names can be created automatically when processing an RTL SAIF file by using the **saif_map -create_map** command, which matches the original names of the design objects with the names in the SAIF file. The SAIF names information can then be used when reading the SAIF file by using the **-map_names** option of the **read_saif** command. Also, the name mapping mechanism can be created automatically and used to read a SAIF file by using the **-auto_map_names** option of the **read_saif** command.

The **saif_map** command provides several options for reporting, querying, and modifying the SAIF name mapping database information (see the list of arguments above). You might need to set the SAIF names of design objects explicitly by using the **-create_map** option in cases where they cannot be created automatically. You can report, query, and modify the original names of the design objects that can be represented in a simple format by using the **-type orig_name** option or in an extended syntax format by using the **-type extended_orig_name** option, which is close to the internal representation of the original names information and provides more information on the scope of the design objects.

You can write the name mapping database information to a file by using the **-write_map** option. You can read the information back into the design by using the **-read_map** option. Note that the name mapping database is stored on the design and is therefore automatically stored in the .ddc files. If you read the design as a gate-

level source file, you might need to read the name mapping information read back into the design.

EXAMPLES

The following example shows how you can automatically create and use the name mapping mechanism when reading an RTL SAIF file into a compiled design.

```
saif_map -start
read_verilog ex.v
current_design ex
link
compile
read_saif -input ex.rtl.saif -instance tb/u1 -auto_map_names
```

The following example shows how you can create the name mapping mechanism, report it, modify it slightly, and use it to read an RTL SAIF file.

```
saif_map -start
read_verilog ex.v
current_design ex
link
compile
saif_map -create_map -input ex.rtl.saif -source tb/u1
saif_map -report
saif_map -get_name [get_port clk_1]
saif_map -set_name "clk1" [get_port clk_1]
read_saif -input ex.rtl.saif -instance tb/u1 -map_names
```

The following example shows how the SAIF name mapping information can be stored to a file and used later. Because the **saif_map -write** command is used before the SAIF name mapping is created by using **saif_map -create_map**, there are no SAIF names in the database and only the original names are stored in the name mapping file.

```
saif_map -start
read_verilog ex.v
current_design ex
link
compile
change_name -rules verilog
write -f verilog -hier -out ex.gate.v
saif_map -write_map ex.namemap

...

read_verilog ex.gate.v
saif_map -read_map ex.namemap
read_saif -input ex.rtl.saif -instance tb/u1 -auto_map_names
```

SEE ALSO

[read_saif\(2\)](#)

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

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.

Creation and usage of scenarios is available with Design Compiler Graphical.

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 for all flows except the timing-driven congestion flow.

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 for all flows except the timing-driven congestion flow. The default value for the timing-driven congestion flow is medium.

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 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.

set_ahfs_options

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.

```
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

report_ahfs_options(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)
```

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.

set_annotated_delay

-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_aspect_ratio

Specifies the placement aspect ratio for the core area. This command is supported only in topographical mode.

SYNTAX

```
int set_aspect_ratio  
y_x_ratio
```

Data Types

```
y_x_ratio      float
```

ARGUMENTS

```
y_x_ratio
    Specifies aspect ratio constraint for core area, it is defined as y:x ratio.
    For example, specifying 1.2 as the aspect ratio will result a core area whose
    y dimension is 1.2 times of x dimension. The default value for aspect ratio
    is 1.0.
```

DESCRIPTION

The **set_aspect_ratio** command specifies the design-level physical constraint for aspect ratio, which is used to generate the coarse floorplan during synthesis. Use this command before running synthesis commands, and after loading the physical library and the design.

There are also other commands that set constraints on core area, such as **set_placement_area**, **set_rectilinear_outline** and **set_aspect_ratio**. Among the four commands, **set_placement_area** has the highest priority, while **set_rectilinear_outline** has higher priority than **set_utilization** and **set_aspect_ratio**. This means if the placement area constraint is set by **set_placement_area** or rectilinear outline constraint is set by **set_rectilinear_outline** on the current design, then the aspect ratio constraint will not be used during synthesis. Furthermore, in this case, aspect ratio will be neither reported by the command **report_physical_constraints**, nor written out by the command **write_physical_constraints**.

EXAMPLES

The following example shows how to set the aspect ratio to 2.0.

```
prompt> set_aspect_ratio 2.0
```

SEE ALSO

```
set_placement_area(2)  
report_physical_constraints(2)  
write_physical_constraints(2)
```

set_attribute

Sets an attribute to a specified value on the specified list of objects.

SYNTAX

```
collection set_attribute
object_list
attribute_name
attribute_value
[-type boolean | integer | float | string]
[-bus]
[-quiet]
```

Data Types

<i>object_list</i>	list
<i>attribute_name</i>	string
<i>attribute_value</i>	string

ARGUMENTS

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.

-type boolean | integer | float | string
Specifies the data type of the *attribute_value*. This argument is required when creating new attributes; otherwise, it is optional. If the attribute data type is not specified, the tool uses either the specified *attribute_value* or data type of the existing attribute.

-bus
Set attributes on the bus instead of bus members.

-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_compile_options(2)
report_design(2)
reset_design(2)
set_dont_touch(2)
```

```
set_dont_touch_network(2)
set_ideal_net(2)
set_ideal_network(2)
```

set_autofix_configuration

Controls automatic fixing of violations when running the **preview_dft** or **insert_dft** command.

SYNTAX

```
int set_autofix_configuration
[-
type clock | set | reset | xpropagation | internal_bus | external_bus | bidirection
al]
[-test_data test_data_name]
[-method mux | gate | enable_one | disable_all | input | output | no_disabling]
[-fix_data enable | disable]
[-fix_latch enable | disable]
[-include_elements list_of_design_objects]
[-exclude_elements list_of_design_objects]
[-control_signal control_name]
```

Data Types

<i>test_data_name</i>	string
<i>list_of_design_objects</i>	list
<i>control_name</i>	string

ARGUMENTS

-type clock | set | reset | xpropagation | internal_bus | external_bus | bidirectional

| external_bus | bidirectional

Specifies the type of fix to which the configuration applies. Specify one type with each invocation of the command.

-test_data *test_data_name*

Specifies the name of the test data signal for fixing violations. This signal is used as a source by the autofix test_points. Predefine the signal as having a **clock** signal type for the **clock** type and a **Reset** signal type for the **set** and **reset** types. Predefine this signal as having a **TestData** signal type. If no test data signal is specified, Autofix creates a new Clock or Reset port depending on the fixing type.

-method mux | gate | enable_one | disable_all | input | output | no_disabling

| input | output | no_disabling

Specifies whether to use muxes or gates for the **set** and **reset** types to autofix sets and resets. By default, Autofix inserts muxes to fix asynchronous signals. For the **internal_bus**, **external_bus**, and **bidirectional** types, the command specifies how to insert disabling logic. By default, the fix method for **internal_bus** is **enable_one**, for **external_bus** it is **disable_all**, and for

bidirectional it is **input**.

-fix_data enable | disable

Enables or disables fixing of clock-used-as-data violations. The default value is **disable**.

-fix_latch enable | disable

Enables or disables fixing of uncontrollable **set** and **reset** pins on latches. The default value is **disable**. This does not affect the enable signal of the latch.

-include_elements *list_of_design_objects*

Specifies the design objects to include for fixing Clock, Set and/or Reset violations. If this option is used then only the specified design objects are fixed. By default, all violated design objects are autofixed. Hierarchical cells are not accepted and leaf level cells must be specified. Any specified hierarchical cells will be discarded.

-exclude_elements *list_of_design_objects*

Specifies the design objects to exclude from fixing Clock, Set and Reset violations. By default, the list of excluded design objects is empty. Hierarchical cells are not accepted and leaf level cells must be specified. Any specified hierarchical cells will be discarded.

-control_signal *control_name*

Specifies the name of the control signal for fixing violations. Predefine the signal as having either the **TestMode** or **ScanEnable** signal type. The **ScanEnable** signal type is valid only for the **set** and **reset** types. If no control signal is specified, the autofix test points are controlled by any signal with the **TestMode** signal type. If no **TestMode** signal is found, Autofix creates a new TestMode port.

DESCRIPTION

The **set_autofix_configuration** command makes specifications on a type basis when running the **preview_dft** or **insert_dft** command. These specifications are only made if one of the Autofix features (**fix_clock**, **fix_set**, **fix_reset**, **fix_xpropagation**, or **fix_bus**), is enabled using the **set_dft_configuration** command.

The Autofix configuration is specific to the current design. If you change the Autofix configuration and then change the current design, your changes are no longer visible.

Use the **report_autofix_configuration** command to display the current Autofix configuration for the design.

EXAMPLES

The following command specifies that Autofix fixes uncontrollable clock violations, but not asynchronous preset_clear violations. The inserted test points are controlled by the *MY_TM* TestMode signal and the test data is the *MY_CLK* clock. From the clock violations, the *U1* cell is not autofixed.

set_autofix_configuration

```
prompt> set_dft_configuration -fix_clock enable \  
-fix_set disable -fix_reset disable
```

```
prompt> set_autofix_configuration -type clock -control_signal MY_TM \  
-test_data MY_CLK -exclude_elements U1
```

The following commands specify that Autofix fixes uncontrollable clock violations and uses data clock_autofix_clock_s as the new Autofix clock and control clock_autofix_mode as the test mode enable pin. Note that the test mode and clock pins to be used must be defined with set_dft_signal as is appropriate for the type of fixing.

```
prompt> set_dft_signal -view existing_dft -type ScanMasterClock -timing {45 55} -  
port clock_autofix_clock_s  
prompt> set_dft_signal -view spec -type TestData -port clock_autofix_clock_s  
prompt> set_dft_signal -view spec -type TestMode -port clock_autofix_mode  
prompt> set_autofix_configuration -type clock \  
-include_elements [get_object_name [get_cells -  
hierarchical *]] \  
-control_signal clock_autofix_mode \  
-test_data clock_autofix_clock_s
```

SEE ALSO

```
insert_dft(2)  
preview_dft(2)  
report_autofix_configuration(2)  
report_dft_configuration(2)  
reset_autofix_configuration(2)  
set_dft_configuration(2)
```

set_autofix_element

If one of the Autofix features is enabled (`fix_clock`, `fix_set`, `fix_reset`, `fix_xpropagation` or `fix_bus`), the command makes specifications on a type and design object basis during execution of **preview_dft** or **insert_dft**.

SYNTAX

```
int set_autofix_element  
list_of_design_objects  
[-  
type clock | set | reset | xpropagation | internal_bus | external_bus | bidirection  
al]  
[-control_signal control_name]  
[-test_data test_data_name]  
[-method mux | gate | enable_one | disable_all | input | output | no_disabling]  
[-fix_data enable | disable]  
[-fix_latch enable | disable]
```

Data Types

list_of_design_objects list

ARGUMENTS

list_of_design_objects

Specifies the list of design objects to which the configuration applies. Hierarchical cells are not accepted by this command and leaf level cells must be specified. Any specified hierarchical cells will be discarded.

-type clock | set | reset | xpropagation | internal_bus | external_bus | bidirectional

Specifies the type of fix to which the configuration applies. Only one type must be specified with each invocation of this command.

-control_signal *control_name*

Specifies the name of the control signal for fixing violations. The signal should be predefined as having either TestMode or ScanEnable signal type. The ScanEnable signal type is valid only for the types set and reset. If no control signal is specified, the autofix test points will be control by any signal with the TestMode signal type. If no TestMode signal is found, Autofix will create a new TestMode port.

-test_data *test_data_name*

Specifies the name of the test_data signal for fixing violations. This signal will be used as a source by the autofix test points. The signal should be predefined as having a clock signal type for the type clock and a Reset signal type for the types set and reset. This signal should also be predefined as having a TestData signal type. For the types clock, set and reset, when no test_data signal is specified, Autofix examines the fanin cone of the combinational logic of the uncontrollable clock or asynchronous signal violations for an existing port defined respectively as a clock or asynchronous signal. If no clock or asynchronous port is found, Autofix creates a new Clock or Reset port depending on the fixing type.

`-method mux | gate | enable_one | disable_all | input | output | no_disabling`
 For the types set and reset, specifies whether to use muxes or gates to autofix sets and resets. By default, Autofix inserts muxes to fix asynchronous signals. For the types `internal_bus`, `external_bus` and `bidirectional`, specifies how to insert disabling logic. By default, the fix method for internal_busses is `enable_one`, the fix method for external_busses is `disable_all` and the default for `bidirectional` is `input`.

`-fix_data enable | disable`
 Enables or disables fixing of clock-used-as-data violations. The default value is `disable`.

`-fix_latch enable | disable`
 Enables or disables fixing of set and reset of latches. The default is `disable`.

DESCRIPTION

If Autofix is enabled, this command specifies a list of elements and indicates whether or not **preview_dft** and **insert_dft** are to automatically fix the specified scan rule violation for those elements. By default, Autofix automatically fixes all uncontrollable clock violations and asynchronous preset/clear, bus and xpropagation violations detected by **check_test**. You can override this default behavior and disable automatic fixing of all violations using the **set_autofix_configuration -clock false, -async false, -bus false** or **-xprop false** options. The **set_autofix_element** command overrides both the default behavior and the **set_autofix_configuration** settings for the specified cells.

This command has no effect unless Autofix is enabled. To enable Autofix, execute **set_dft_configuration** with required options. See **set_dft_configuration** command for details.

Specifying **set_autofix_element** on a cell that is free of violations does not affect **insert_dft**.

To reset to the default behavior for the specified objects, set the appropriate option to true.

EXAMPLES

The following example specifies that **preview_dft** and **insert_dft** are not to automatically fix the specified three sequential cells in the current design.

```
prompt> set_autofix_element [list U3_1 U3_2 U3_3] -clock false -async false
```

SEE ALSO

```
current_design(2)
insert_dft(2)
preview_dft(2)
set_autofix_configuration(2)
report_autofix_configuration(2)
set_autofix_element(2)
```

```
report_autofix_element(2)
```

set_balance_registers

Sets the **balance_registers** attribute on the specified designs or on the current design, so that the design is retimed during compile.

SYNTAX

```
integer set_balance_registers  
[true | false]  
[-design design_list]  
[design_list]
```

Data Types

design_list list

ARGUMENTS

true | false
Specifies the value with which to set the **balance_registers** attribute. The default value is true, which allows the **compile** command to automatically invoke the **balance_registers** command. When the value is set to false, **balance_registers** is not invoked.

-design *design_list*
Specifies a list of designs to retime. The default is the current design.

DESCRIPTION

This command sets the **balance_registers** attribute on the specified designs or on the current design, so that the design is retimed during compile. If the **balance_registers** attribute is set to true (the default) on a design, **compile** automatically invokes the **balance_registers** command, which moves registers to minimize the maximum register-to-register delay. Subdesigns in the hierarchy are ungrouped into the design, unless the **dont_touch** attribute is set.

It is an error to invoke **balance_registers** on a design that contains generic logic. If the **balance_registers** attribute is set, **compile** attempts to optimize the design by invoking **balance_registers**. Ensure that your design contains no generic logic at the instant **balance_registers** is called during **compile**.

For more details on retiming during optimization, see the Design Compiler documentation.

To remove **balance_registers**, use the **remove_attribute** or **reset_design** command. You can achieve the same effect by setting the **balance_registers** attribute to false; execute **set_balance_registers false**.

Licensing during compile when using the **balance_registers** attribute can be controlled by using the **compile_retime_license_behavior** dc_shell variable.

EXAMPLES

In the following example, **balance_registers** is enabled on the design named TEST:

```
prompt> set_balance_registers -design TEST
```

In the following example, **balance_registers** is enabled on the current design and **balance_registers** is disabled on the design named OLD:

```
prompt> read TEST
```

```
prompt> set_balance_registers
```

```
prompt> set_balance_registers false -design OLD
```

SEE ALSO

```
balance_registers(2)  
compile(2)  
current_design(2)  
remove_attribute(2)  
reset_design(2)
```

set_boundary_cell

Sets the boundary cell configuration for the specified ports and core cells.

SYNTAX

```
status set_boundary_cell
-class core_wrapper | shadow_wrapper | bsd
[-
function input | output | control | bidir | observe | input_inverted | output_inver
ted | bidir_inverted | receiver_p | receiver_n | ac_select]
[-ports port_list]
[-core_cells core_list]
[-
type WC_D1 | WC_D1_S | WC_S1 | WC_S1_S | BC1 | BC2 | BC4 | BC7 | BC8 | BC9 | AC1 |
AC2 | AC7 | AC_SEL | AC_SELX | none]
[-design design_name]
[-register_io_implementation swap | in_place]
[-use_dedicated_wrapper_clock true | false]
[-safe_state 0 | 1 | none]
[-share true | false]
[-name bcell_name]
[-shift_clk bcell_shift_clk]
[-instance hier_cell_list]
[-include cell_list]
[-exclude cell_list]
```

Data Types

<i>port_list</i>	list
<i>core_list</i>	list
<i>design_name</i>	string
<i>bcell_name</i>	string
<i>bcell_shift_clk</i>	port or pin
<i>hier_cell_list</i>	list of hierarchical cells
<i>cell_list</i>	list of IO Registers

ARGUMENTS

-class core_wrapper | shadow_wrapper | bsd
Specifies the name of the class for which the configuration applies. Valid values are **core_wrapper**, **shadow_wrapper**, and **bsd**. This argument is required.

-function input | output | control | bidir | observe | input_inverted | output_inverted | bidir_inverted | receiver_p | receiver_n | ac_select

Specifies the function of the specified ports to which the local configuration applies.
Valid values are as follows:
input

```

output
control
bidir
observe
input_inverted
output_inverted
bidir_inverted
receiver_p
receiver_n
ac_select

```

There is no default value for this option.

-ports *port_list*

Specifies the list of ports for which the configuration applies.
There is no default value for this option. Either the **-ports** option or the **-core** option is required.

-type WC_D1 | WC_D1_S | WC_S1 | WC_S1_S | BC1 | BC2 | BC4 | BC7 | BC8 | BC9 | AC1 | AC2 | AC7 | AC_SEL | AC_SELX | none

Specifies the cell type to be used for the boundary cell in DFT insertion.
Valid values for this option are as follows:

```

WC_D1
WC_D1_S
WC_S1 |
WC_S1_S
BC1
BC2
BC4
BC7
BC8
BC9
AC1
AC2
AC7
AC_SELU
AC_SELX
none

```

There is no default value for this option. If this option is not specified, the tool uses the **set_wrapper_configuration** command to get the type of DFT design to use for the specified ports.

-design *design_name*

Specifies the name of the user-defined DFT design as specified by the **define_dft_design** command to be used for the boundary cell.
This option is mutually exclusive with the **-type** option.
There is no default value for this option. By default, the tool uses a DesignWare cell for DFT insertion.
Either the **-type** option or the **-design** option is required.

-register_io_implementation swap | in_place

Specifies the implementation to use for synthesizing shared wrapper cells in core wrapping the current design.
When the option is set to **swap**, registers connected to the specified ports

set_boundary_cell

of the design are replaced with equivalent shared wrapper cells. When the option is set to **in_place**, additional glue logic is added to registers attached to the specified ports of the design to reuse them in core wrapper chains.

There is no default value for this option. If the option is not specified, the tool uses the **set_wrapper_configuration** command to get the implementation style to use for the specified ports.

-use_dedicated_wrapper_clock true | false

Specifies whether the dedicated wrapper clock is used for shared wrapper. When the option is set to true, core wrapper application uses the dedicated wrapper clock for all shared wrapper cells of the specified ports of the design. When the option is set to false, functional clocks are used for shared wrapper cells connected to the specified ports of the design.

-safe_state 0 | 1 | none

Specifies the safe state to be used for wrapper cells added by core wrapping. Valid values for this option are **0**, **1**, and **none**.

There is no default value for this option. If the option is not specified, the tool uses the **set_wrapper_configuration** command to get the safe state to use for the specified ports.

-share true | false

Specifies whether the boundary cells should be shared. Sharing boundary cells is allowed only for control boundary cells. If the value of the option is set to true, the boundary cells for the specified ports are shared. If the value of the option is set to false, the boundary cells for the specified ports are not shared.

The default value of this option is true for function value control and false for all other types of boundary cells.

-name *bcell_name*

Specifies the name of the boundary cell. The name of the boundary cell is mandatory if the **-share** option is set to true.

There is no default value for this option.

-shift_clk *bcell_shift_clk*

Specifies the shift clock to be used for the boundary cell.

This option is applicable only with the **-class core_wrapper** and either of the **-type WC_D1** or **-type WC_D1_S** options.

The value of this option must be a port or an internal pin. The specified port or pin is connected to the shift clock pin of the dedicated boundary cell.

The timing information of the specified clock is obtained from DRC.

There is no default value for this option.

-instance *hier_cell_list*

This option is applicable only when the class option is set to **core_wrapper** and **maximize_reuse** option of **set_wrapper_configuration** command is enabled. This option specifies a list of hierarchical cells. If a port is connected to a register inside one of these hierarchical cells, the tool would add the specified type of boundary cell to the port.

The valid types of boundary cells supported with this option are **none** or **WC_D1**.

If the type of the boundary cell is chosen as **none**, no wrapper cell is added

to the port.

If the type of the boundary cell is chosen as WC_D1, a dedicated wrapper cell is added to the port.

Option -port is mutually exclusive with this option.

There is no default value for this option.

-include *cell_list*

This option is applicable only when the class option is set to core_wrapper and maximize_reuse option of set_wrapper_configuration command is enabled. This option specifies the list of IO register cells to be included for the specified port.

There is no default value for this option.

-exclude *cell_list*

This option is applicable only when the class option is set to core_wrapper and maximize_reuse option of set_wrapper_configuration command is enabled. This option specifies the list of IO register cells to be excluded for the specified port.

To specify a list of cells to be excluded from all the ports, don't specify the port names with the command.

There is no default value for this option.

DESCRIPTION

The **set_boundary_cell** command overrides the default configuration for the specified ports and core cells of the current design. The ports and core cells must exist in the current design.

To set the default core wrapper configuration, use the **set_wrapper_configuration -class core_wrapper** command.

To set the default shadow wrapper configuration, use the **set_wrapper_configuration -class shadow_wrapper** command.

To set the default BSD configuration, use the **set_bsd_configuration** command.

EXAMPLES

The following example specifies that the DesignWare WC_D1 cell is to be used with safe value 1 on *port1* and *port2* in core wrapping:

```
prompt> set_boundary_cell -class core_wrapper -type WC_D1 \  
       -safe_state 1 -port_list {port1 port2}
```

The following example specifies that the DesignWare BC_1 cell is to be used for ports *port1* and *port2* in BSD insertion:

```
prompt> set_boundary_cell -class bsd -type BC_1 \  
       -function output -port_list {port1 port2}
```

The following example specifies that the DesignWare BC_2 cell is to be used as a control BSR cell for ports *port1* and *port2* in BSD insertion: The control BSR cells for these ports are not shared.


```
prompt> set_boundary_cell -class bsd -type BC_2 \  
-function control -port_list {port1 port2} -name CTRL1 -share false
```

The following example specifies that the DesignWare BC_2 cell is to be used as a control BSR cell for ports *port1* and *port2* in BSD insertion. The control BSR cell is shared for these ports.

```
prompt> set_boundary_cell -class bsd -type BC_2 \  
-function control -port_list {port1 port2} -name CTRL1 -share true
```

The following example specifies that the DesignWare AC_SELU cell is to be used as the AC selection BSR cell for ports *ac_port1* and *ac_port2* in BSD insertion. The AC selection BSR cell is shared for these ports.

```
prompt> set_boundary_cell -class bsd -type AC_SELU \  
-function ac_select -port_list {ac_port1, ac_port2} -name SELECT1
```

The following example specifies a user defined implementation of BC_1 BSR cell to be used for ports *port1* and *port2* in BSD insertion.

```
prompt> define_dft_design -type BC_1 -design my_bc_1 -  
interface {capture_clk cclk h \  
update_clk uclk h shift_dr se h mode tm h si si h data_in di h data_out  
do h so so h}  
  
set_boundary_cell -class bsd -design my_bc_1 -port_list {port1 port2}
```

The following example specifies that no boundary cell to be added to the ports connected to hierarchical cell *core1_inst/core_inst*.

```
prompt> set_boundary_cell -class core_wrapper -type none \  
-instance core1_inst/core2_inst
```

The following example specifies that a dedicated boundary cell to be added to the ports connected to hierarchical cell *core1_inst/core_inst*.

```
prompt> set_boundary_cell -class core_wrapper -type WC_D1 \  
-instance core1_inst/core2_inst
```

The following example specifies that I2_reg, I2_1_reg to be included as boundary cells for the port I2.

```
prompt> set_boundary_cell -class core_wrapper -port {I2} \  
-include {I2_reg I2_1_reg}
```

The following example specifies that I2_reg, I2_1_reg to be excluded as boundary cells for the port I2.

```
prompt> set_boundary_cell -class core_wrapper -port {I2} \  
-exclude {I2_reg I2_1_reg}
```

The following example specifies that I2_reg, I2_1_reg to be excluded as boundary cells for the design.

```
prompt> set_boundary_cell -class core_wrapper \
```

```
-include {I2_reg I2_1_reg}
```

SEE ALSO

```
insert_dft(2)  
preview_dft(2)  
remove_boundary_cell(2)  
report_boundary_cell(2)  
set_bsd_configuration(2)  
set_wrapper_configuration(2)
```

set_boundary_cell_io

Specifies the primary inputs (PIs) and primary outputs (POs) of a boundary cell in the current design.

SYNTAX

```
int set_boundary_cell_io  
-access {pin_type pin_name}  
-type boundary | wrapper  
-cell shift_flop_name
```

Data Types

<i>pin_type</i>	string
<i>pin_name</i>	string
<i>shift_flop_name</i>	string

ARGUMENTS

-access {*pin_type pin_name*}

Specifies a list of pairs, with each pair consisting of the pin type and the hierarchical pin name. Valid values for *pin_type* are **in_pi**, **in_po**, **out_pi**, **out_po**.
This argument is required.

-type boundary | wrapper

Specifies the type of cell as either **boundary** or **wrapper**.
This argument is required.

-cell *shift_flop_name*

Specifies the names of the shift flop of the boundary cell.
This argument is required.

DESCRIPTION

The **set_boundary_cell_io** command specifies the primary inputs (PIs) and primary outputs (POs) of boundary cells. For input cells, specify only **in_pi** and **in_po**. For output and control cells, specify **out_pi** and **out_po**. For merged cells, specify all four values (**in_pi**, **in_po**, **out_pi**, **out_po**). The cell name indicates the name of the shift flop of the cell.

EXAMPLES

The following example instructs **check_bsd** to use the specified PIs and POs for the BSR cell *bs_d_oe2/q2bscan/inst1/inst4*:

```
prompt> set_boundary_cell_io -type boundary \  
      -access {out_pi bs_d_oe2/U1/**logic_0** out_po U8/U1/A} \  
      -cell bs_d_oe2/q2bscan/inst1/inst4/q_reg \  
      \
```

SEE ALSO

`check_bsd(2)`
`write_bsd1(2)`

set_boundary_optimization

Sets the **boundary_optimization** attribute on specified cells, references, or designs, thus allowing for optimization across hierarchical boundaries.

SYNTAX

```
int set_boundary_optimization  
obj_list  
[true | false]
```

Data Types

obj_list list

ARGUMENTS

obj_list
Specifies a list of cell, reference, or design names for which to enable boundary optimization. Cell or reference names in *obj_list* must be from the *current design*. If more than one object are specified, they must be enclosed in quotes (") or braces ({}).

true | false
Specifies the value with which to set the **boundary_optimization** attribute. When this option is set to **true** (the default), boundary optimization is enabled.

DESCRIPTION

Sets the **boundary_optimization** attribute on *obj_list*. This attribute is used to specify the cells, references or designs to be optimized across hierarchical boundaries. The **compile** command uses this information to create smaller designs. This might change the function of the object, so the object should not be used in any other context.

If a cell with a specified name is found in the *current design*, the **boundary_optimization** attribute is set to the specified value in the cell. If no cell with a specified name is found, the tool searches for a reference. If a reference is found, the attribute is set for the reference. Otherwise, the tool searches for a design and the attribute set for the design.

Occasionally, cells use an input signal only in its complemented form. In this case, it is best to invert the signal and its ports. The **set_boundary_optimization** command considers these optimizations. When this occurs, port names are changed according to the **port_complement_naming_style** variable.

To remove this attribute, use the **remove_attribute** command.

EXAMPLES

The following example shows how to ignore hierarchical boundaries during optimization for cells named *U0* and *U1* and how to preserve them for a cell named *U2*.

```
prompt> set_boundary_optimization {U0 U1}  
prompt> set_boundary_optimization U2 false
```

SEE ALSO

```
compile(2)  
find(2)  
get_attribute(2)  
remove_attribute(2)  
reset_design(2)  
uniquify(2)  
port_complement_naming_style(3)
```

set_bsd_ac_port

Identifies the AC ports in your design.

SYNTAX

```
int set_bsd_ac_port  
-port_list list_of_ports
```

Data Types

list_of_ports list

ARGUMENTS

```
-port_list list_of_ports  
           Specifies the ports of the current design to be considered as AC ports.
```

DESCRIPTION for all modes

The **set_bsd_ac_port** command identifies the ports of the current design to be considered as AC ports. Boundary Scan Insertion adds AC BSR cells for such ports if IEEE1149.6_2003 standard mode is enabled.

This specification is honored only Boundary Scan Synthesis.

To preview the boundary-scan implementation in XG mode use *preview_dft -bsd cells*. To implement the boundary-scan logic, use **insert_dft**. To delete boundary-scan AC port specifications, use **remove_bsd_specification -ac_port**.

EXAMPLES for all modes

The following example shows how the ports A_1, A_2, and A_3 are identified as AC ports.

```
prompt> set_bsd_ac_port -port_list {A_1, A_2, A_3}  
prompt> set_bsd_ac_port -port_list {A_1 A_2 A_3}
```

SEE ALSO

```
insert_dft(2)  
preview_dft(2)
```

set_bsd_compliance

Specifies an IEEE 1149.1 compliant pattern for a boundary-scan design.

SYNTAX

```
status set_bsd_compliance  
-name pattern_name  
-pattern signal_port_bit_value_pairs
```

Data Types

```
pattern_name                string  
signal_port_bit_value_pairs list
```

ARGUMENTS

```
-name pattern_name  
    Specifies the name of the pattern. Patterns having the same name merge and  
    overwrite earlier versions.  
    This argument is required.  
  
-pattern signal_port_bit_value_pairs  
    Lists signal port and binary value pairs that specify the compliance-enable  
    pattern. The specified signal ports must be input ports and cannot include  
    IEEE 1149.1 test access ports. Valid bit values specified at each signal port  
    are 0 or 1.  
    This argument is required.
```

DESCRIPTION

The **set_bsd_compliance** command specifies the compliance-enable ports and patterns that enable the IEEE 1149.1 boundary-scan functionality for a design.

The functionality of the IEEE 1149.1 boundary-scan logic in certain designs depends on the logic values at the compliance-enable ports for the designs. Compliance-enable ports are defined in ANSI/IEEE Std. 1149.1, section 3-8.

The **set_bsd_compliance** command is mandatory for all IEEE 1149.1 designs that contain at least one compliance-enable port.

EXAMPLES

The following example specifies one compliance-enable pattern named *p1* for the current design:

```
prompt> set_bsd_compliance -name p1 \  
        -pattern [get_ports c_en*, 1, ntest, 0]
```


SEE ALSO

`reset_bsd_configuration(2)`

set_bsd_configuration

Specifies the boundary-scan configuration for a design.

SYNTAX

```
status set_bsd_configuration
[-asynchronous_reset true | false]
[-default_package package_name]
[-instruction_encoding binary | one_hot]
[-ir_width instruction_register_length]
[-style synchronous | asynchronous]
[-check_pad_designs none | all | pad_design_list]
[-control_cell_max_fanout max_fanout]
[-ieee1149.1_1993 enable | disable]
[-std std_version_name_list]
```

Data Types

<i>package_name</i>	string
<i>instruction_register_length</i>	integer
<i>max_fanout</i>	integer
<i>std_version_name_list</i>	list of strings

ARGUMENTS

- asynchronous_reset true | false**
Specifies whether or not the boundary-scan circuitry synthesized by the **insert_dft** command is to have an asynchronous test logic reset port. When set to **true** (the default), an asynchronous test logic reset port is used and when set to **false** there is no asynchronous test logic reset port. You must provide a power-up mechanism.
- default_package *package_name***
Specifies the default package for a boundary-scan design. The *package_name* must correspond to one of the packages for the design read in using the **read_pin_map** command.
- instruction_encoding binary | one_hot**
Specifies the instruction encoding scheme to use to generate the decoding logic for the instruction register and the instruction code assignment. Setting the value to **binary** (the default) causes the **insert_dft** command to use the binary code to encode the instructions, using a minimum number of bits for the instruction register. Setting the value to **one_hot** causes the **insert_dft** command to use codes that have only one bit set to logic 1 with all other bits set to logic 0. Exceptions are the BYPASS and EXTEST instructions, which have bits set to logic 1.
- ir_width *instruction_register_length***
Specifies the length of the instruction register, in bits, for the boundary-scan design. Allowed values are the integers **2** through **16**, inclusive. The default value is the minimum value calculated by BSD Compiler to fit the set of instructions you set.

-style *synchronous* | *asynchronous*
 Specifies the style of boundary-scan to implement. Valid values are **asynchronous** and **synchronous**. The default style is **synchronous**.

-check_pad_designs *none* | *all* | *pad_design_list*
 Specifies the pad designs to check. When set to **none**, the **preview_dft** command does not validate any pad design.
 When set to **all** (the default), the **preview_dft** command validates all pad designs instantiated in the design.
 When given a *pad_designs_list*, the **preview_dft** command validates all pad designs specified in the list.
 Signal type **port** must be specified in the access list of all pad designs that are to be validated. See the man page for the **set_bsd_pad_design** command for details.

-control_cell_max_fanout *max_fanout*
 Specifies the maximum number of fanouts allowed for the control cell when **-share true** is specified in the **set_boundary_cell** command.

-ieee1149.1_1993 *enable* | *disable*
 Specifies the mode in which BSD Compiler performs. This option is disabled by default, so BSD Compiler runs in the IEEE1149.1-2001 standard mode.
 When set to **enable**, BSD Compiler runs in the old IEEE1149.1-1993 standard mode.
 This option is obsolete, use **-std {ieee1149.1_1993}** to enable IEEE1149.1-1993 standard mode.

-std *std_version_name_list*
 Specifies the modes in which BSD Compiler performs. You can specify one or more of the following values: **ieee1149.1_1993**, **ieee1149.1_2001**, and **ieee1149.6_2003**.
 When set to **-std ieee1149.1_1993**, BSD Compiler runs in the old IEEE1149.1-1993 standard mode.
 When set to **-std ieee1149.6_2003**, BSD Compiler runs in the IEEE1149.6-2003 standard mode.
 When set to **-std ieee1149.1_1993 ieee1149.6_2003**, BSD Compiler runs in the IEEE1149.1-1993 and IEEE1149.6-2003 standard modes.
 By default BSD Compiler runs in the IEEE1149.1-2001 standard mode.
 The **insert_dft** command generates the RTL output file with this name.

DESCRIPTION

The **set_bsd_configuration** command specifies the boundary-scan configuration for a design. Customize the implementation by selecting the synchronization style, the instruction register length, the reset mechanism, and the port-to-pin mapping.

When the **asynchronous** style is selected, the asynchronous signals coming from the TAP controller cell, *capture_dr* and *update_dr*, are connected to the boundary-scan register cells. The asynchronous version of the TAP controller is used.

When the **synchronous** style is selected, the synchronous signals coming from the TAP controller cell, *capture_en*, *update_en*, and *TCK* are connected to the boundary-scan register cells. The synchronous version of the TAP controller is used. By default, BSD Compiler generates a synchronous boundary scan design.

By specifying the instruction length, you can control the opcode length of the implemented instruction. By default, BSD Compiler calculates the minimum length of the instruction register needed to encode both the mandatory instruction and those instructions specified using the **set_bsd_instruction** command. This also depends on the encoding scheme selected with the **-instruction_encoding** option.

The mapping of logical signals onto the physical pins of a component package is described through a port-to-pin map file. Multiple packages for a design can be read in, using the **read_pin_map** command. The sequence of ports listed in the pin mapping file after the line declaring the package name drives the order of the corresponding cells in the boundary-scan register. The default package name specified with the command **set_bsd_configuration -default_package** appears in the generic parameter statement of the BSDL file produced by the **write_bsd1** command.

Use the **report_test -bsd_configuration** command to display the current default package for the design.

Use the **reset_bsd_configuration** command to remove the settings for the design.

EXAMPLES

The following example enables ieee1149.1_1993:

```
prompt> set_bsd_configuration -ieee1149.1_1993 enable
```

SEE ALSO

```
read_pin_map(2)
remove_pin_map(2)
report_bsd_configuration(2)
reset_bsd_configuration(2)
set_bsd_instruction(2)
```

set_bsd_instruction

Specifies boundary-scan instructions used by the **insert_dft** command for the current design or used by the **check_bsd** command in the verification flow.

SYNTAX

```
status set_bsd_instruction
[-view existing_dft | spec]
instruction_list
[-code inst_code_list]
[-register register_name]
[-input_clock_condition clock_conditioning]
[-output_condition BSR | HIGHZ | NONE]
[-internal_scan pin_name]
[-capture_value capture_value_list]
[-private]
[-clock_cycles clock_cycle_list]
[-signature pattern]
[-high pin_name_list]
[-low pin_name_list]
[-time real_time]
[-excluded_bsr_condition CLAMP | NONE]
```

Data Types

<i>instruction_list</i>	list
<i>inst_code_list</i>	list
<i>register_name</i>	string
<i>clock_conditioning</i>	string
<i>pin_name</i>	string
<i>capture_value_list</i>	list
<i>clock_cycle_list</i>	list
<i>pattern</i>	string
<i>pin_name_list</i>	list
<i>real_time</i>	float

ARGUMENTS

-view existing_dft | spec

Specifies the view to which the specification applies. Valid views are **existing_dft** and **spec**.

Set the view to **existing_dft** to refer to the existing instructions in the design. Use the **existing_dft** value when working with BSD-inserted designs. **For example, use the following command to specify the instruction for the tool to implement:**

```
prompt> set_bsd_instruction bypass -view existing_dft \  
          -register BYPASS -code 1111
```

Set the view to **spec**, the default, to refer to instructions that the tool must use when running **check_bsd**. Use this view when boundary scan synthesis is not performed by **insert_dft** (in the verification flow). For example, to direct the tool that the

my_instr instruction must be implemented as targeted for the register **BOUNDARY**, use the following command:

```
prompt> set_bsd_instruction my_instr -view spec \  
-register BYPASS -code 1001
```

instruction_list

Specifies a set of boundary instructions.

-code inst_code_list

Specifies the list of binary codes corresponding to the instructions specified by *instruction_list*. A single instruction can be associated with an opcode. This means that the *instruction_list* must contain a single identifier if **-code inst_code_list** is used. The opcode must have a length equal to the number set by **set_bsd_configuration -ir_width**. By default, the opcode is assigned automatically by BSD Compiler.

When using this option with the **check_bsd** command in the verification flow, opcodes should be specified. Only **BYPASS** and **EXTEST** reserved opcodes are inferred automatically.

-register register_name

Selects a standard data register or a user-defined register, previously declared with the **set_dft_signal** and the **set_scan_path** commands, to be connected for serial access between TDI and TDO when the instruction is active. Valid standard data registers are **BOUNDARY** and **BYPASS**. The **BOUNDARY** value is automatically selected for **EXTEST**, **SAMPLE**, **PRELOAD** and **INTEST**. The **BYPASS** value is automatically selected for **BYPASS**, **CLAMP** and **HIGHZ**. User-defined instructions must specify a register. You cannot specify **BOUNDARY** as a data register for the standard instructions **BYPASS**, **HIGHZ**, **CLAMP**, **IDOCDE**, **USERCODE**. Similarly, **BYPASS** cannot be specified as the data register for the standard instructions **EXTEST**, **INTEST**, **SAMPLE_PRELOAD**, **SAMPLE**, **PRELOAD**, **IDCODE**, **USERCODE**.

-input_clock_condition clock_conditioning

Specifies the value that drives the clock signal going into the system when the instruction is active. If you are clocking the TDR with the system clock, set the value to **PI**. If you are using TCK clock, set the value to **TCK**. Valid values are **PI** or **TCK**. The default is **PI**.

-output_condition BSR | HIGHZ | NONE

Specifies how the system outputs are driven when the instruction is active. Valid conditions are as follows:

- **BSR** puts the boundary scan register into **EXTEST** mode.
- **HIGHZ** puts the output pins into high impedance mode.
- **NONE** puts the boundary scan register into transparent mode (the default).

-internal_scan pin_name

Specifies a hierarchical pin in the design to which the TAP controller's shift-dr signal gated with the decoded instruction is connected.

-capture_value capture_value_list

Defines the capture value of the DEVICE-ID register for the **IDCODE**

set_bsd_instruction

instruction. The capture value contains 32 digits. Ensure that the least significant bit (LSB) of the capture value is set to 1 for the IDCODE capture value.

-private

Specifies that the instructions are considered private. Private instructions have the **INSTRUCTION_PRIVATE** attribute in the BSDL file generated by the tool. The data registers of private instructions are not shown in the BSDL file. No test patterns are generated for private instructions. By default, the instructions are not considered private.

-clock_cycles *clock_cycle_list*

Lists the clock port and integer pairs that specify the minimum number of clock cycles on the clock port required for the design to stay in the Run-Test/Idle TAP controller state to ensure completion of the INTEST or the RUNBIST instruction.

-signature *pattern*

Specifies the state of the boundary-scan register after execution of the RUNBIST instruction. The *pattern* argument correlates to the <det pattern> argument defined in section B.8.15 of the Supplement to IEEE Std. 1149.1.

-high *pin_name_list*

Specifies a list of pins that must be active high when the specified instructions are selected. The *pin_name_list* contains the hierarchical names of the pins. There is no default for this option. This option can be used with any instruction, including standard instructions.

-low *pin_name_list*

Specifies a list of pins that must be active low when the specified instructions are selected. The *pin_name_list* variable contains the hierarchical names of the pins. There is no default for this option and this option can be used with any instruction including standard instructions.

-time *real_time*

Specifies a real number that indicates the time duration in nanoseconds for the EXTEST_TRAIN and EXTEST_PULSE instructions in the Run-Test-Idle (RTI) state, as well as for INTEST and RUNBIST. The tool returns an error if this option is used with other instructions.

-excluded_bsr_condition CLAMP | NONE

Specifies the condition of BSR cells excluded from the specified short BSR chain. The valid values for this option are as follows:

- **CLAMP** all BSR cells that are not part of the specified BSR chain are conditioned as if CLAMP instruction is active.
- **NONE** all BSR cells that are not part of the specified BSR chain are kept transparent while the user instruction is active.

Note: The error, Invalid value '%s' specified for option '-excluded_bsr_condition'., occurs when a value other than CLAMP or NONE is used with this option.

DESCRIPTION

The **set_bsd_instruction** command specifies one or more boundary-scan instructions that the **insert_dft** command implements for the current design. The **check_bsd** command infers these instructions in the verification flow when the **-infer_instructions** switch is set to **true** or if the **insert_dft** command does not perform boundary-scan synthesis. Alternately, all boundary-scan synthesis decoded opcodes need to be listed. The default value for the **-infer_instructions** switch is false, in which case, it checks compliance of the instructions specified in the run script. You can specify a list of IEEE Std 1149.1 instructions. The keywords for the standard instructions are BYPASS, EXTEST, SAMPLE, PRELOAD, IDCODE, CLAMP, HIGHZ and USERCODE.

For all other standard and private instructions, including INTEST and RUNBIST, you can specify their corresponding binary code, input clock conditioning, user-defined register, and output conditioning.

If you do not specify any binary code, the **insert_dft** command selects one automatically in the design flow. However, you must specify opcodes in the Verification flow for the BYPASS, SAMPLE, PRELOAD, and EXTEST instructions when the **set_bsd_instruction** command is used with the **check_bsd** command and the **-infer_instructions** switch is set to **true**. The input clock conditioning and output conditioning (BSR or HIGHZ) specifications are required for the standard instructions INTEST and RUNBIST.

By default, the capture and update clock pins of the Test Data Register(TDR) specified with instructions are not gated, that is, the pins are always driven by TCK.

To gate the clock pins of a TDR with a MUX so that the pins are driven by TCK when the instruction that selects the TDR is active and driven by system logic when the instruction is not active, set the following variable to TRUE prior to BSD insertion:

```
set test_bsd_synthesis_gated_tck TRUE
```

The **-clock_cycle** option specifies a list of clock port and integer pairs, each pair specifies the minimum number of clock cycles on the specified clock port that the design needs to stay in the Run-Test or the Idle TAP controller state to ensure completion of the RUNBIST, INTEST, EXTEST_PULSE, and EXTEST_TRAIN instructions.

The **-time** option applies to EXTEST_PULSE and EXTEST instructions. For the EXTEST_PULSE instruction, the *real_time* value specifies the minimum wait time in nanoseconds in the Run-Test or the Idle TAP controller state. Note that you can specify only one of the options **-clock_cycles** or **-time** for this instruction.

For the EXTEST_TRAIN instruction, the *real_time* value specifies the maximum time in nanoseconds prior to the exit of the Run-Test or the Idle TAP controller state within which the specified minimum number of pulses specified with the **-clock_cycles** option should occur for EXTEST_TRAIN instruction. Note that the **-time** option should always be used along with **-clock_cycles** for this instruction.

The **-signature** option specifies a binary string that represents the signature produced by the RUNBIST instruction.

The **capture_value** option allows you to specify a 32-bit binary or hexadecimal value and pins in the design, that provide the capture value, to be connected to the DIR register. This option also allows you to specify a mix of bits and design pins.

Specify a valid user code value for the USERCODE instruction. The user code value can only be specified for the USERCODE instruction. Specify binary or hexadecimal bit-patterns using the size-constant Verilog syntax as shown in the following example:

```
<number of binary bits>'[b|h|B|H]<binary or hex value>
```

Specify pin names with the full hierarchical pin name. Specify a bus representing a collection of pins using the following syntax:

```
<name of the bus>[<upper bit>:< lower bit>]
```

To review your boundary-scan instruction specifications, use **preview_dft -bsd all**. To implement the boundary-scan instruction set, use **insert_dft**. To delete boundary-scan instruction specifications, use **remove_bsd_instruction**.

EXAMPLES

The following example directs **insert_dft** to implement the HIGHZ and CLAMP instructions as a part of the synthesized boundary-scan circuitry:

```
prompt> set_bsd_instruction -view spec {HIGHZ CLAMP}
```

The following example directs **insert_dft** to implement the user-defined instruction named *UDI1*, whose binary code is 1010, and to select the boundary-scan register to be connected for serial access between TDI and TDO. This puts the outputs in high-impedance instructions as a part of the synthesized boundary-scan circuitry.

```
prompt> set_bsd_instruction -view spec UDI1 -register BOUNDARY \  
-code {1010} -output_conditioning HIGHZ
```

The following example directs **insert_dft** to implement the INTEST instructions, and to drive the system clock by TCK:

```
prompt> set_bsd_instruction -view spec INTEST \  
-input_clock_condition TCK
```

The following example directs **insert_dft** to implement the USERCODE instruction:

```
prompt> set_bsd_instruction -view spec USERCODE -code {0110} \  
-user_code_val {version_reg/v3 version_reg/v2 \  
version_reg/v1 version_reg/v0 16'b1111111110000000 12'h2ab}
```

SEE ALSO

check_bsd(2)
insert_dft(2)
preview_dft(2)
remove_bsd_instruction(2)
set_bsd_configuration(2)
set_scan_path(2)

set_bsd_linkage_port

Identifies the linkage ports in your design.

SYNTAX

```
int set_bsd_linkage_port  
-port_list list_of_ports
```

Data Types

list_of_ports list

ARGUMENTS

```
-port_list list_of_ports  
           Specifies the ports of the current design to be considered as linkage ports.
```

DESCRIPTION for all modes

The **set_bsd_linkage_port** command identifies the ports of the current design to be considered as linkage ports. Such ports may be analog ports, power or ground ports, or any port driven by a black box cell that you do not wish to consider for boundary-scan insertion.

The following will not occur for ports declared as linkage ports by the **set_bsd_linkage_port** command:

- Checking for pad cells connected to the port during boundary-scan insertion.
- Insertion of a boundary-scan cell to be connected to the port.
- Compliance checking on the port.
- Toggling of the port in test vector generation.

The linkage ports will be listed as linkage bits in the BSDL.

To preview the boundary-scan implementation in dctl and DB (dcsh) modes, use **preview_dft -bsd cells**, in XG mode use *preview_dft -bsd all*. To implement the boundary-scan logic, use **insert_dft**. To delete boundary-scan linkage port specifications, use **remove_bsd_specification -linkage_port**.

EXAMPLES for all modes

The following example shows how the ports PLL_1, PLL_2, and PLL_3 are identified as linkage bits.

```
prompt> set_bsd_linkage_port -port_list {PLL_1, PLL_2, PLL_3}  
prompt> set_bsd_linkage_port -port_list {PLL_1 PLL_2 PLL_3}
```

SEE ALSO

insert_dft(2)

preview_dft(2)

set_bsd_power_up_reset

Specifies and characterizes the power-up reset cell for the current design.

SYNTAX

```
integer set_bsd_power_up_reset  
-cell_name cell_name  
-reset_pin_name reset_pin_name  
-active high | low  
[-delay power_up_reset_delay]
```

Data Types

<i>cell_name</i>	string
<i>reset_pin_name</i>	string
<i>power_up_reset_delay</i>	integer

ARGUMENTS

-cell_name *cell_name*
Specifies the instance name of the power-up reset cell.

-reset_pin_name *reset_pin_name*
Specifies the pin name of the power-up reset cell at which a reset pulse is generated upon power-on.

-active high | low
Specifies whether the power-up reset pulse is active **high** or active **low**.

-delay *power_up_reset_delay*
Specifies the initial power-up reset delay, which is measured from the time power is switched on to the time TAP controller resets to the test-logic-reset state.

DESCRIPTION

This command specifies and characterizes the power-up reset cell instance for the design. You must use the **-delay** option with the *power_up_reset_delay* value if the design does not have the (optional) TRST test access port.

EXAMPLES

The following example shows how to specify the instance *PUR* as the power-up reset with a pin named *reset_out* as the pin at which an active-high reset pulse is generated upon power-up with a delay of *1300* units:

```
prompt> set_bsd_power_up_reset -cell_name PUR \  
       -reset_pin_name reset_out -active high -delay 1300
```

SEE ALSO

check_bsd(2)
create_bsd_patterns(2)
insert_dft(2)
preview_dft(2)
set_bsd_configuration(2)
write_test(2)
test_default_period(3)

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_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)  
compile(2)  
remove_attribute(2)  
report_constraint(2)  
set_max_capacitance(2)  
compile_fix_cell_degradation(3)
```

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, orientation, and dont_touch status for leaf cells. This command is supported in Design Compiler Topographical mode only.

SYNTAX for DC Topographical Mode

```
int set_cell_location
```

```
object_list  
-coordinates {X Y}  
[-orientation string]  
[-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.

-orientation *string*
Specifies the target orientation. Valid values are specified either in DEF syntax (**N**, **FN**, **S**, **FS**, **W**, **FW**, **E**, **FE**) or PDEF syntax (**0**, **90**, **180**, **270**, **0-mirror**, **90-mirror**, **180-mirror**, and **270-mirror**).
DEF syntax definitions are the following:
N is nominal orientation (North).
S is 180 degrees rotation (South).
E is 270 degrees counter clockwise rotation (East).
W is 90 degrees counter clockwise rotation (West).
FN is the reflection in the y-axis (flipped North).
FS is 180 degrees rotation followed by reflection in the y-axis (flipped South).
FE is 90 degrees rotation counter clockwise followed by reflection in the y-axis (flipped East).
FW is 270 degrees rotation counter clockwise followed by reflection in the y-axis (flipped West).
The PDEF syntax definitions are the following:
0 is 0 degrees rotation.
90 is 90 degrees counter clockwise rotation.
180 is 180 degrees rotation.
270 is 90 degrees counter clockwise rotation.
0-mirror is reflection in the y-axis.
90-mirror is 90 degrees counter clockwise rotation followed by

reflection in the y-axis.

180-mirror is 180 degrees rotation followed by reflection in the y-axis.

270-mirror is 270 degrees counter clockwise rotation followed by reflection in the y-axis.

-fixed

Indicates that the command **set_cell_location** sets the **dont_touch** attribute on the cells in the list *object_list*, to prevent modification or replacement of these cells during optimization.

DESCRIPTION

The **set_cell_location** command specifies the physical location, orientation, and dont_touch status 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

report_physical_constraints(2)
write_physical_constraints(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}]
[-validate {timing}]
[-analyze {nominal_vs_sigma | table_trend}]
[-criteria {criteria_spec}]
[-group_attribute {groups_or_attributes}]
[-report_format {csv[=csv_dir] nosplit sort_by_cell | sort_by_group_type}]
[-leq]
[-physical]
[-logic_vs_physical]
[-logic]
[-reset]
[-all]
```

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
<i>criteria_spec</i>	list
<i>groups_or_attributes</i>	list

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).

`-validate {timing}`

Specifies logic library consistency checking for library characterization between different timing models.

`-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.

`-criteria {criteria_spec}`

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.

-group_attribute {groups_or_attributes}

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.

-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.

-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
- mcmm
- 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

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_clock_gate_latency

Specifies clock network latency values to be used for clock gating cells, as a function of clock domain, clock gating stage, and fanout.

SYNTAX

```
status set_clock_gate_latency  
[-clock clock_list]  
[-overwrite]  
-stage cg_stage  
-fanout_latency cg_fanout_list
```

Data Types

<i>clock_list</i>	list
<i>cg_stage</i>	integer
<i>cg_fanout_list</i>	string

ARGUMENTS

-clock *clock_list*
Specifies that the latency must be applied with respect to the specified clocks. If **-clock** is not specified, the latency is applied with respect to all clock domains to which the clock gating cell belongs.

-overwrite
Specifies that clock latency values previously set on clock gating cells should be overwritten.

-stage *cg_stage*
Specifies the clock gating stage to which the clock latency data from the fanout range is applied. Registers are considered as stage 0.

-fanout_latency *cg_fanout_list*
Specifies the list of clock gating cells fanout and clock latency values; for example, {1-5 0.9, 6-20 0.5, 21-inf 0.3}. A fanout of 1 to 5 has a latency of 0.9; a fanout of 21 or larger has a latency of 0.3. If the same latency value is desired for the entire fanout range, it can be specified as {1-inf 0.9}.

DESCRIPTION

The **set_clock_gate_latency** command allows you to specify clock network latency values for clock gating cells, as a function of clock domain, clock gating stage, and fanout. These latency values will be annotated on the clock pins of clock gating cells when the **compile_ultra** command is run, or by running the **apply_clock_gate_latency** command.

The clock gate latency settings specified using the **set_clock_gate_latency** command are stored as an attribute on the source port of each specified clock. Invoke this command after defining all clocks of the design. Those stored latency values will be

annotated on the clock pins of clock gating cells by means of one of the following processes:

- Using the **apply_clock_gate_latency** command, which applies latency on any existing clock gating cells
- During **compile_ultra** activity, which applies latency on any existing clock gating cells or those inserted or modified during compile.

Use the **set_clock_gate_latency** command to specify the latency for gated registers by specifying the command for stage 0. Since in this case the fanout parameter is meaningless, the only permitted syntax for **-fanout_latency** *cg_fanout_list* for stage 0 is as follows:

```
prompt> set_clock_gate_latency -stage 0 -fanout_latency \  
         {1-inf <value>}
```

When clock latency settings are provided for stage 0, the values are annotated on the output pin (enabled clock pin) of each clock gating cell that is directly driving some gated registers. This allows the clock latency information to be considered in the timing analysis done for those registers by the timer.

Alternatively, if no setting is provided for stage 0, but a clock latency has been set on the clock object by using **set_clock_latency** command, then this value is set by the tool on the output pin (enabled clock pin) of clock gating cell with directly-driven gated registers, allowing the timer to use the clock latency from clock object for timing analysis of gated registers. For ungated registers, the latency is propagated by the timer from the clock object latency.

You can use the **-clock** and **-overwrite** options with the **-stage 0** option.

If you specify fanout ranges in the *cg_fanout_list* that do not cover all values from 1 to inf, the fanout ranges with missing latency information are filled with the smallest available latency value from the adjacent ranges. The PWR-740 warning message is printed during the processing of the **set_clock_gate_latency** command.

If the specified clock latency values are not decreasing with respect to fanout ranges, the PWR-742 warning message is generated.

If clock latency settings are not specified for a certain clock gating stage and a clock gating cell with this stage is actually found, the PWR-745 warning message is generated. In this case, the latency is propagated from the driving clock gating cell or from clock object latency, if available. These actions are performed during the annotation process; such as when running the **apply_clock_gate_latency** or **compile_ultra** command.

If specified clock latency values are not decreasing with respect to clock gating stages, the PWR-744 warning message is generated during the processing of the **set_clock_gate_latency** command to inform you that this inconsistent setting may produce an incorrect timing analysis. In addition, if an inconsistent clock latency annotation is detected when running the **apply_clock_gate_latency** or **compile_ultra** command, the tool attempts to fix it by assuming the smallest value from the clock latencies annotated on gated clock gating cells or registers. This ensures that clock latencies are monotonically increasing values in the path going from the clock source to the gated registers. The following actions can be performed by the tool to

achieve this:

- If clock gating cell A is directly driving one or more registers, and some clock latency settings have been provided for the timing of these gated registers (by using value 0 for the **-stage** option of **set_clock_gate_latency** command or by propagation of latency specified for clock object), and the latency annotated on cell A is higher than the latency provided for gated registers, then the latency on clock gating cell A is overwritten with the latency provided for gated registers. If this fix is done, the PWR-746 warning message is generated.
- If clock gating cell A is driving another clock gating cell B, and the clock latency set on A is higher than the one set on B, the tool resolves it by overwriting the clock latency on A with the value set on B. If this fix is done, the PWR-746 warning message is generated.

To remove the settings specified using **set_clock_gate_latency**, use the **reset_clock_gate_latency** command.

EXAMPLES

The following example specifies the clock latency values for the complete fanout range of clock gating cells for stages 1, 2, and 3. This latency data applies to the clock gating cells whose clock pins belongs to clock clk1.

```
prompt> set_clock_gate_latency -clock [get_clocks clk1] -stage 1 \  
-fanout_latency {1-30 2.1, 31-100 1.7, 101-inf 1.1}  
  
prompt> set_clock_gate_latency -clock [get_clocks clk1] -stage 2 \  
-fanout_latency {1-5 0.9, 6-20 0.5, 21-inf 0.3}  
  
prompt> set_clock_gate_latency -clock [get_clocks clk1] -stage 3 \  
-fanout_latency {1-10 0.28, 11-inf 0.11}
```

In the following example, fanout ranges specified in the **-fanout_latencycg_fanout_list** do not cover all values from 1 to inf. In this case, the tool assumes a clock latency value of 1.7 to be applied to fanout range 11-30 and a clock latency of 1.1 for fanouts higher than 80.

```
prompt> set_clock_gate_latency -stage 1 \  
-fanout_latency {1-10 2.1, 31-80 1.7, 101-300 1.1}
```

SEE ALSO

```
apply_clock_gate_latency(2)  
remove_clock_latency(2)  
report_clock_gating(2)  
reset_clock_gate_latency(2)  
set_clock_latency(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.

set_clock_gating_check

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

```
compile(2)  
compile_ultra(2)  
report_clock_gating(2)  
set_clock_gating_style(2)
```

set_clock_gating_style

Sets the clock-gating style for the clock-gate insertion and replacement.

SYNTAX

```
status set_clock_gating_style
[-sequential_cell none | latch]
[-minimum_bitwidth minsize_value]
[-setup setup_value]
[-hold hold_value]
[-positive_edge_logic {cell_list | integrated [active_low_enable] [invert_gclk] }
[-negative_edge_logic {cell_list | integrated [active_low_enable] [invert_gclk] }
[-control_point none | before | after]
[-control_signal scan_enable | test_mode]
[-observation_point true | false]
[-observation_logic_depth depth_value]
[-max_fanout max_fanout_count]
[-num_stages num_stages_count]
[-no_sharing]
[-gicg_pos_cell {[cell_library/]cell_name}]
[-gicg_neg_cell {[cell_library/]cell_name}]
[-gicg_pos_auto {[cell_library/]cell_name}]
[-gicg_neg_auto {[cell_library/]cell_name}]
```

Data Types

<i>minsize_value</i>	integer
<i>setup_value</i>	float
<i>hold_value</i>	float
<i>depth_value</i>	integer
<i>max_fanout_count</i>	integer
<i>num_stages_count</i>	integer

ARGUMENTS

-sequential_cell none | latch

Specifies the cell that is used to delay the enable signal gating the clock. To select a latch-based style (the default), use **latch**. By default, the tool selects which latch cell to use. You can specify a specific latch cell by using the following syntax:

latch:*lib_cell*

To select a latch-free style, use **none**.

The latch style must be compatible with the clock-gating circuitry as chosen by the **-positive_edge_logic** and **-negative_edge_logic** options. If you need to use different latches for positive-edge logic and negative-edge logic, use the **-positive_edge_logic** and **-negative_edge_logic** options, rather than using the **-sequential_cell** option. If you specify either of the following, the tool infers a latch-based clock gating style:

```
-sequential_cell none -positive_edge_logic {latch and}
-sequential_cell none -negative_edge_logic {latch or}
```

set_clock_gating_style

See the examples below for this usage.

`-minimum_bitwidth minsize_value`

Specifies the minimum size of a register bank for which the clock can be gated.

`-setup setup_value`

Specifies the setup time constraint at the enable input of the 2-input clock gate in the clock-gating circuitry.

`-hold hold_value`

Specifies the hold time constraint at the enable input of the 2-input clock gate in the clock-gating circuitry.

`-positive_edge_logic {cell_list | integrated [active_low_enable] [invert_gclk]}`

Specifies the circuitry used to gate the clock of a flip-flop that is inferred by a positive edge clock construct in the HDL code. You can specify the circuitry in one of the following ways:

- *cell_list*

Specifies a list of 1 to 4 strings that describe the gates inserted in the clock network. The circuitry must be compatible with the latch style specified by the **-sequential_cell** option (for example, AND or NAND functionality for a latch-based style and OR or NOR functionality for a latch-free style).

If the circuitry inverts the clock signal, an additional inverter is inferred between the clock output pin of the clock-gating circuitry and the clock input pin of the register. In contrast to an inverter specified in the clock-gating circuitry, you can map this inverter with the generic sequential cell to a negative-edge triggered library flip-flop.

- {integrated [active_low_enable] [invert_gclk]}

Uses a single special integrated cell instead of the clock-gating circuitry.

`-negative_edge_logic {cell_list | integrated [active_low_enable] [invert_gclk]}`

Specifies the circuitry used to gate the clock of a flip-flop that is inferred by a negative edge clock construct in the HDL code. You can specify the circuitry in one of the following ways:

- *cell_list*

Specifies a list of 1 to 4 strings that describe the gates inserted in the clock network. The circuitry must be compatible with the latch style specified by the **-sequential_cell** option (for example, OR, NOR functionality for a latch-based style and AND, NAND functionality for a latch-free style). If the circuitry does not invert the clock signal, an additional inverter is inferred between the clock output pin of the clock-gating circuitry and the clock input pin of the register. In contrast to an inverter specified in the clock-gating circuitry, you can map this inverter with the generic sequential cell to a negative-edge triggered library flip-flop.

- {integrated [active_low_enable] [invert_gclk]}

Uses a single special integrated cell instead of the clock-gating circuitry.

`-control_point none | before | after`

Specifies where to insert a control point in the clock gating circuitry. With *before* or *after* control styles, the tool implements an OR gate with the original register enable signal and a test control signal as inputs. If you specify a latch-based style, the *before* or *after* control styles determine the location of the OR gate with respect to the latch. The tool creates a new input port to provide the test signal. The control points must be hooked up to the design level `test_mode` or `scan_enable` port using the **insert_dft** command.

`-control_signal scan_enable | test_mode`

Specifies the test control signal. If an input port is created and the argument of the **-control_signal** option is *scan_enable*, the name of the port is determined by the **test_scan_enable_port_naming_style** variable, and a **test_scan_enable** signal_type attribute is put on the new port. If an input port is created and the argument of the **-control_signal** option is *test_mode*, the name of the port is determined by the **test_mode_port_naming_style** variable, and a **test_hold** attribute is put on the new port.

`-observation_point true | false`

Specifies whether or not to implement observability logic. The observed signals are the enable signal inputs to the control point OR gates. A new input port is created to provide the test signal, which blocks activity to the observability logic during mission mode.

`-observation_logic_depth depth_value`

Specifies the maximum depth of an XOR tree built in the observability circuitry.

`-max_fanout max_fanout_count`

Specifies the maximum fanout of a single clock-gating cell.

`-num_stages num_stages_count`

Specifies the maximum number of stages for multistage clock gating.

`-no_sharing`

Limits the registers gated by a clock-gating cell to a single register bank.

DESCRIPTION

This command sets the clock-gating style to be used for clock-gating with the **compile_ultra -gate_clock** and **replace_clock_gates** commands, as well as power-driven and gate-level clock gating enabled by specifying the **-gate_clock** option to **compile** or **compile_ultra** commands.

Clock gating of register banks is an alternative implementation of load-enable registers. In comparison to the traditional implementation using recirculating feedback and multiplexers, power consumption is reduced on the clock tree, in the registers, and in the combinational logic (multiplexers). Clock gating has implications on testability and physical design.

The clock gating style specifies three aspects of clock gating:

`set_clock_gating_style`

- The conditions when clock gating is applied
- The clock-gating circuitry that is implemented
- Additional circuitry to improve testability.

Clock-gating Conditions

A number of conditions must be satisfied to gate a register clock in a latch-based style. First, the register must be a load enable flip-flop. Second, the register belongs to a bank with a bit width equal to or larger than the value specified by the **-minimum_bitwidth** option. Because a clock gating circuitry, and possibly some testability improving gates, is implemented for each register bank, a register bank should have a reasonable size to be considered for clock gating. The default value is 3. For power-driven clock gating the default minimum width is 1. During compilation, the optimization algorithm decides if a register bank is large enough to be clock gated based on the switching activity and the dynamic power of the bank; therefore it is recommended not to set an minimum width when performing power-driven clock gating.

An additional condition (known as the setup condition) must be satisfied for latch-free clock gating. All of the inputs to the combinational logic generating the load enable signal of a register bank must be flip-flops that are triggered by the same clock and on the same clock edge as the register bank. For hierarchical designs it may be necessary to derive the related clock for the inputs to the combinational logic from the clock connectivity in parent hierarchies. This is enabled by setting the variable **power_cg_derive_related_clock** to true. Use the **set_clock_gating_registers** command to force clock gating of registers that do not satisfy these conditions, or disable the setup condition for all registers by setting the variable **power_cg_ignore_setup_condition** to true.

Specifying the Clock-Gating Circuitry

The clock-gating circuitry is a hierarchical cell. The cell is attributed such that its structure is maintained during compilation, which means you have tight control over the structure of the clock-gating circuitry.

There are several options to determine the clock-gating circuitry. The most important option is **-sequential_cell**, which determines latch-based or latch-free clock gating. The basic effect of latch-based clock gating is to prevent a leading clock edge by maintaining the value of the clock signal after the trailing edge. For example, for positive-edge triggered flip-flops, the clock signal is forced and kept to Logic0 after the negative edge, preventing a rising edge that loads new data into the register. To achieve this, the clock-gating circuitry must have AND or NAND functionality for positive-edge triggered flip-flops, and OR or NOR functionality for negative-edge triggered flip-flops. To prevent problems on the gated clock signal, the load enable signal is delayed by a latch that is transparent and it is between the trailing and the leading clock edge. Additionally, setup and hold time constraints are set at the enable input of the clock gate. The **-setup** option specifies the setup time value; the **-hold** option specifies the hold time value. The **-sequential_cell** option can also be used to specify a particular D-latch from a

particular library:

```
string seq_cell = none | latch[:[library_name/]cell_name]
string library_name
string cell_name
```

Latch-free clock gating prevents a leading clock edge by maintaining the value of the clock signal before the trailing edge. For example, for positive-edge triggered flip-flops, the clock signal is forced and kept to Logic1 before the negative edge. To achieve this, the clock-gating circuitry must have OR or NOR functionality for positive-edge triggered flip-flops, and AND or NAND functionality for negative-edge triggered flip-flops. Because the load enable signal must arrive before the trailing edge of the clock, there is no latch. Additionally, setup and hold time constraints are set at the enable input of the clock gate. The setup time value is given by the **-setup** option; the hold time value by the **-hold** option.

Before compilation, you must propagate the setup and hold timing constraints to the current design using the **propagate_constraints -gate_clock** command.

Note that the term "positive-edge triggered flip-flop" here means that the register is inferred by a positive edge clock construct in the HDL code.

Because the clock-gating circuitry for positive-edge triggered flip-flops differs from that for negative-edge triggered flip-flops, you have two options for specifying the clock gating circuitries: **-positive_edge_logic** (for flip-flops inferred by a positive edge construct in the HDL code) and **-negative_edge_logic** (for flip-flops inferred by a negative edge construct in the HDL code). Both options have as arguments a list of one to four strings that describe the gates inserted in the clock network. If you give four strings, the first specifies the latch, the third string specifies the 2-input clock gate (and/nand/or/nor gate), the second string specifies a 1-input gate (inverter or buffer) in the fanin of the clock gate on the clock network, and the fourth string specifies a 1-input gate (inverter or buffer) in the fanout of the clock gate. The specification of the 1-input gates in the fanin and fanout of the clock gate is optional. There can be four strings, each of which can contain the specification of a library cell:

```
string gate = gate_type[:[library_name/]cell_name]
string gate_type = latch | and | nand | or | nor | buf | inv
string library_name
string cell_name
```

Note that the **-positive_edge_logic** and **-negative_edge_logic** options must be compatible with the latch style as specified by the **-sequential_cell** option or in the gate-level list. For latch-based clock gating, the **-positive_edge_logic** option must specify AND or NAND functionality, and the **-negative_edge_logic** option must specify OR or NOR functionality. For latch-free clock gating, the **-positive_edge_logic** option must specify OR or NOR functionality, and the **-negative_edge_logic** option must specify AND or NAND functionality.

AND functionality is given by one of the following lists:

```
{and}
```

```
{buf and}
{and buf}
{buf and buf}
{nand inv}
{buf nand inv}
{inv or inv}
{inv nor}
{inv nor buf}
```

NAND functionality is given by one of the following lists:

```
{and inv}
{buf and inv}
{nand}
{buf nand}
{nand buf}
{buf nand buf}
{inv or}
{inv or buf}
{inv nor inv}
```

OR functionality is given by one of the following lists:

```
{or}
{buf or}
{or buf}
{buf or buf}
{nor inv}
{buf nor inv}
{inv and inv}
{inv nand}
{inv nand buf}
```

NOR functionality is given by one of the following lists:

```
{or inv}
{buf or inv}
{nor}
{buf nor}
{nor buf}
{buf nor buf}
{inv and}
{inv and buf}
{inv nand inv}
```

You can designate certain library cells to be used exclusively or preferentially for gating clocks. Such cells can be used for the two-input clock gate, inverters, buffers, or D-latches in the clock-gating circuitry. To use a specific cell for clock gating and to preclude its use in other areas of the design, set the **dont_use** and the **is_clock_gating_cell** attributes to true in the library description. To preferentially use a specific cell in the clock gating circuitry, set only the **is_clock_gating_cell** attribute to true. The **is_clock_gating_cell** attribute is a

user-defined attribute of type Boolean for the cell group.

You can also designate a specific input pin of the two-input clock gate as the enable input. If the clock attribute is set on one of the input pins of the two-input clock gate, the other input is recognized as the enable pin. Otherwise, if the **clock_gate_enable_pin** attribute is set to true on an input, this pin is recognized as the enable pin. The **clock_gate_enable_pin** attribute is a user-defined attribute of type Boolean for the pin group.

Besides specifying simple 2-input gates, you also have the option to specify that a single complex clock gating cell be used for the clock gating logic. This is done by specifying "integrated" for the gate list to the **-positive_edge_logic** or **-negative_edge_logic** options. Note that if these options are specified as "integrated", the strings after it are of integrated clock gating cell types. No inverters or buffers are allowed before or after the integrated cell. This string can contain the specification of a library cell:

```
string gate = {integrated[:[library_name/]cell_name] \
               [active_low_enable] [invert_gclk]}
string library_name
string cell_name
```

If a library cell is specified for use as an integrated clock gating cell, the cell must have the **clock_gating_integrated_cell** attribute. Specify the attribute value as "generic". Library Compiler infers the functionality of the library cell based on the cell pins' logic functions. For backward compatibility, this string attribute can still have the value and specifies the functionality of the cell as a clock-gating cell. Based on the presence or absence of a latch, whether it is to be used as **positive_edge_logic** or **negative_edge_logic**, whether the control point is present and whether the observability port is present, the **clock_gating_integrated_cell** can have the following values:

```
latch_posedge
latch_posedge_precontrol
latch_posedge_postcontrol
latch_posedge_precontrol_obs
latch_posedge_postcontrol_obs
latch_negedge
latch_negedge_precontrol
latch_negedge_postcontrol
latch_negedge_precontrol_obs
latch_negedge_postcontrol_obs
none_posedge
none_posedge_control
none_negedge
none_negedge_control
```

Also, the pins of the integrated cell must have appropriate attributes, which include:

```
clock_gate_enable_pin attribute on the enable input pin
clock_gate_clock_pin attribute on the clock input pin
```

set_clock_gating_style

clock_gate_out_pin attribute on the gated-clock output pin
clock_gate_test_pin attribute on the scan-enable or test-mode pin
clock_gate_obs_pin attribute on the observability output pin

Specifying Additional Circuitry to Improve Testability

The next sections are concerned with the specification of additional circuitry to improve testability of designs with gated clocks. The last four options specify circuitry that is introduced to improve the testability. The impact of clock gating on testability is twofold. First, due to the clock gate, the controllability of the gated clock signal is reduced. Therefore, the clock-gating style allows the insertion of an OR gate, which forces the enable signal to Logic1 during testing. The **-control_point** option specifies if and where the OR gate is implemented. The **-control_signal** option specifies if a scan enable or a test mode signal is used to drive the OR gate. (The test mode signal is held constant during testing. The scan enable signal is guaranteed to be Logic1 only during scan-in and scan-out.) A port is created in the design according to the chosen test control signal.

Second, due to the control point OR gate, the enable signal is not observable if a test mode signal has been chosen as input to the control point OR gate. To improve the observability, the **-observation_point** option enables the insertion of observation points on the enable signals. The enable signals are grouped according to the clock signal and the clock edge of the register with which they are associated. An XOR gate tree fed by enable signals and providing input to an observability register is implemented for each group. The maximum depth of the XOR tree is determined by the **-observation_logic_depth** option. To minimize power consumption in the observability circuitry, the enable signals and the observability register's clock signal are gated by the test control signal.

The **-max_fanout** option enables you to specify a number for the maximum fanout of clock-gating cell at the RTL level. Based on the **max_fanout_count**, Power Compiler splits the register banks that are to be clock-gated, and creates as many clock-gating cells as required to drive them.

This option does not override the **max_fanout** design rule constraint during the compile. The **max_fanout_count** value is an integer number greater than 0. It should not be less than the **minimum_bitwidth** value. The default value is a huge number (unlimited for all practical purposes).

The **-num_stages** option enables multistage clock gating with the **compile_ultra -gate_clock** command. The **num_stages_count** value is an integer number greater than 0. It determines the maximum number of stages that are introduced by factoring of the inserted clock gates. The default value is 1, so no factoring is performed.

The **-no_sharing** option can be used only with the **-max_fanout** option. When you use this option, Power Compiler does not share any clock-gating cells between register banks. When you do not use this option, Power Compiler enables you to share clock-gating cells between register banks with similar control logic to meet the **max_fanout_count** value.

If **compile_ultra -gate_clock** or **replace_clock_gates** is issued without explicitly setting the clock-gating style beforehand (such as when the **set_clock_gating_style** is not issued) a default style is applied.

The default style is derived from the *target_library* that is set when the **compile_ultra -gate_clock** or **replace_clock_gates** command is issued. The style is chosen such that the best available clock gating configuration is used with the current *target_library*. The "best available" style corresponds to the first legal style in context of the current *target_library* of the following combinations:

- (a) **set_clock_gating_style -pos integrated -neg integrated **
-control_point before -control_signal scan_enable
- (b) **set_clock_gating_style -pos integrated -neg integrated **
-control_point after -control_signal scan_enable
- (c) **set_clock_gating_style -pos integrated -neg integrated **
-control_point before -control_signal test_mode \
-observation_point true
- (d) **set_clock_gating_style -pos integrated -neg integrated **
-control_point after -control_signal test_mode \
-observation_point true
- (e) **set_clock_gating_style -pos integrated -neg integrated**
- (f) **set_clock_gating_style -pos integrated -neg or **
-control_point before -control_signal scan_enable
- (g) **set_clock_gating_style -pos integrated -neg or **
-control_point after -control_signal scan_enable
- (h) **set_clock_gating_style -pos integrated -neg or **
-control_point before -control_signal test_mode \
-observation_point true
- (i) **set_clock_gating_style -pos integrated -neg or **
-control_point after -control_signal test_mode \
-observation_point true
- (j) **set_clock_gating_style -pos integrated -neg or**
- (k) **set_clock_gating_style -pos and -neg integrated **
-control_point before -control_signal scan_enable
- (l) **set_clock_gating_style -pos and -neg integrated **
-control_point after -control_signal scan_enable
- (m) **set_clock_gating_style -pos and -neg integrated **
-control_point before -control_signal test_mode \
-observation_point true
- (n) **set_clock_gating_style -pos and -neg integrated **
-control_point after -control_signal test_mode \
-observation_point true
- (o) **set_clock_gating_style -pos and -neg integrated**

(p) **set_clock_gating_style -pos and -neg or**

EXAMPLES

Default values are assumed for all unspecified options. In particular, option values from previous clock-gating style settings are not stored.

The following example sets the clock-gating style such that banks with two or more flip-flops are implemented with gated clocks:

```
prompt> set_clock_gating_style -minimum_bitwidth 2
```

The following example specifies the clock-gating circuitry, including the library cells for gating the clock of registers inferred by a positive edge clock HDL code construct. Note that for the negative edge registers, the default is used. The latch-based style is chosen, therefore, the circuitry must have AND functionality. A particular latch LD2 is chosen from the target library.

```
prompt> set_clock_gating_style -sequential_cell latch:LD2 \  
      -positive_edge_logic {nand:NAND2 inv:IV3}
```

The following example specifies the clock-gating circuitries for registers inferred by positive and negative edge clock HDL code constructs. By default, latch-based clock gating is assumed. Three gates are specified on the clock network. The mapping to library cells happens automatically during compilation. A hold time constraint of 0.5 is specified. By default, the setup time constraint is 0.

```
prompt> set_clock_gating_style -pos {buf nand inv} \  
      -neg {buf or inv} -hold 0.5
```

The following example specifies the clock-gating circuitries for registers inferred by positive and negative edge clock HDL code constructs (latch-free clock gating). Notice that an inverter is inferred for those registers that are inferred by a negative edge clock construct in order to obtain an inverted clock signal at the flip-flops clock pin.

```
prompt> set_clock_gating_style -sequential_cell none \  
      -pos {or} -neg {and}
```

The following example specifies that clock gating is performed with the insertion of control point OR gates:

```
prompt> set_clock_gating_style -control_point after \  
      -control_signal scan_enable
```

The following example specifies that clock gating is performed with the insertion of control point OR gates, as well as the insertion of observation points:

```
prompt> set_clock_gating_style -control_point before \  
        -control_signal test_mode -observation_point true \  
        -observation_logic_depth 4
```

The following example specifies a latch based clock-gating style and limits the fanout of individual clock-gating cells to 8. The **-no_sharing** option ensures that a single clock-gating cell gates registers from only one register bank.

```
prompt> set_clock_gating_style -sequential latch \  
        -max_fanout 8 -no_sharing
```

The following example specifies the clock-gating circuitries for registers inferred by positive and negative edge clock HDL code constructs. Notice that an active low enabled and inverting gclk output integrated clock gating cell is chosen for registers that are inferred by a positive edge clock construct. An inverting gclk output integrated clock gating cell is chosen for registers that are inferred by a negative edge clock construct.

```
prompt> set_clock_gating_style \  
        -pos {integrated active_low_enable invert_gclk} \  
        -neg {integrated invert_gclk}
```

The following example specifies two different latches for positive and negative edge logic:

```
prompt> set_clock_gating_style \  
        -positive_edge_logic {latch:lsi_10k/LD2 and} \  
        -negative_edge_logic {latch:lsi_10k/LD1 or}
```

SEE ALSO

```
compile(2)  
compile_ultra(2)  
insert_dft(2)  
propagate_constraints(2)  
replace_clock_gates(2)  
set_clock_gating_registers(2)  
power_cg_derive_related_clock(3)  
power_cg_ignore_setup_condition(3)
```

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_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)  
translate(2)  
target_library(3)
```

set_compile_directives

Controls the application of high-level optimization operations on cells, references, designs, and library cells.

SYNTAX

```
status set_compile_directives  
object_list  
[-delete_unloaded_gate true | false]  
[-constant_propagation true | false]  
[-local_optimization true | false]  
[-critical_path_resynthesis true | false]
```

ARGUMENTS

object_list

Applies the **set_compile_directive** command on the list of objects (cells, references, designs, and library cells) you specify. This command is not applied on classes of cells that can only be internally generated (for example, generic logic, synthetic operators, and control logic to selector operators). If you include more than one object, the objects must be enclosed in quotes or braces ({}). For more information about object names, refer to the **get_object_name** command man page.

-delete_unloaded_gate true | false

Allows you to delete unloaded gates from the cells specified in the *object_list* when set to **true** (the default). When **false**, cells referenced by *object_list* are not deleted, even if they do not drive any load.

-constant_propagation true | false

Allows constants to be propagated through the cells specified in *object_list* when set to **true** (the default). When **false**, constant propagation, within or across hierarchies, is disabled for the specified cells.

-local_optimization true | false

Allows you to perform operations on the cells specified in the *object_list* when set to **true** (the default). When **false**, operations that involve the immediate neighborhood of cells referenced by the object list are not performed. Such operations include buffering, phase shift operations, and so on.

-critical_path_resynthesis true | false

Allows you to perform critical path resynthesis on cells specified in the *object_list*. when set to **true** (the default). When **false**, critical path resynthesis is not performed on cells referenced by *object list*.

DESCRIPTION

You must have a DC-Ultra license to use this command.

The **set_compile_directives** command controls switching on or off of certain high-

level optimization steps on cells, references, designs, and library cells. The high-level optimization steps controlled by this command include deletion of unloaded gates, propagation of constants, local optimization (buffering, phase operations, and so on), and critical path resynthesis. Note that changing the default value (**true**) of any of these command-line options also implies that the cells referenced by the object list are not structured.

An important application of this command is to direct synthesis to perform only sizing operations on a cell. You can achieve this by setting all command options to **false**. The **set_size_only** command achieves the same result. The **set_size_only** command does not require a DC-Ultra license.

Setting **set_compile_directives** on a hierarchical cell implies **set_compile_directives** on all cells below it. Setting **set_compile_directives** on a library cell implies **set_compile_directives** on all instances of that cell. Setting **set_compile_directives** on a reference implies **set_compile_directives** on all cells of that reference during subsequent optimizations of the design.

Setting the **set_compile_directives** attribute on a design has an effect only when the design is instantiated within another design as a level of hierarchy. In this case, **set_compile_directives** on the design implies that all cells under that level of hierarchy acquire the same directives. Setting **set_compile_directives** on the top-level design has no effect because the top-level design is not instantiated within any other design.

NOTE: Synthetic parts and generic cells ignore the **set_compile_directives** command (for example, many of the cells read in from an HDL description). Control logic feeding selector operator cells also ignore this command. Thus, the command is not applied on any class of cells that can only be internally generated.

When you specify an object name, **set_compile_directives** first looks within the current design for a cell that matches that name. If it finds a match, **set_compile_directives** attributes are assigned to that cell. If no match is found, the command then looks for a reference, then a design in the system, and finally for a library_cell in the system. This command assigns **set_compile_directives** attributes to the first object for which it finds a match.

The **report_cell** command prints all attributes set by the command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command specifies to size the "block1" and "analog1" cells during **compile** only.

```
prompt> set_compile_directives -del false -const false -local false \  
-crit false [get_cells {block1 analog1}]
```

SEE ALSO

`compile(2)`
`get_object_name(2)`
`report_cell(2)`
`set_size_only(2)`

set_compile_partitions

Specifies the compile partitions for the current design.

SYNTAX

```
status set_compile_partitions
-level level
    | -designs design_list
    | -all
    | -auto
[-force]
[-no_reset]
```

Data Types

<i>level</i>	int
<i>design_list</i>	list

ARGUMENTS

-level *level*

Creates partitions from all designs at the specified hierarchical level. Level one selects the subdesigns of the top-level design. The specified level must be greater than or equal to one. You must specify one, but not more than one, of the following options: **-level**, **-designs**, **-all**, or **-auto**.

-designs *design_list*

Creates partitions from the specified designs. You must specify one, but not more than one, of the following options: **-level**, **-designs**, **-all**, or **-auto**.

-all

Creates partitions from all subdesigns throughout the hierarchy of the current design. You must specify one, but not more than one, of the following options: **-level**, **-designs**, **-all**, or **-auto**.

-auto

Creates partitions from all designs automatically. You must provide the threshold for the partition size. See the *Automated Chip Synthesis User Guide* for threshold setting. You must specify one, but not more than one, of the following options: **-level**, **-designs**, **-all**, or **-auto**.

-force

Enables partitioning, even if the design contains multiple instances. If the specified design contains multiple instances, Automated Chip Synthesis selects as the master instance the instance with the hierarchical name that is first alphabetically. For more information about master instances, see the **MasterInstance** attribute man page.

By default, the **set_compile_partitions** command fails if the design contains multiple instances. You must either resolve these instances before running the command, or use the **-force** option.

-no_reset
 Adds the specified partitions to the existing partition definitions.
 By default, the **is_partition** attributes are removed from all subdesigns before creating the specified partitions.

DESCRIPTION

The **set_compile_partitions** command creates partitions from the specified designs by setting the **is_partition** attribute on the designs. When Automated Chip Synthesis (ACS) compiles a top-level design, it creates parallel compile jobs for each partition.

EXAMPLES

The following example assumes the design RISC_CORE is already in memory, and that the current design is RISC_CORE. It creates partitions from all designs automatically and uses **-force** to enable the partition since there are multiple instantiated designs.

```
prompt> set_compile_partitions -auto -force
```

In this example, the **report_partitions** command is used to see the results.

```
prompt> report_partitions
```

```
*****
Report      : partitions
Design      : RISC_CORE (unmapped)
Date        : Thu May 10 15:37:37 2001
*****
```

Partition attributes :

```
(*)  - partition.
%    - the percentage w.r.t total design area.
d    - dont_touch.
m(n) - multiple instantiated design (number of instantiation).
```

Designs	Attributes

RISC_CORE (*)	8.7%
STACK_TOP	3.9%
STACK_MEM (*)	7.6%, m(3), I_STACK_TOP/
I1_STACK_MEM	
STACK_FSM	3.2%
REG_FILE (*)	20.7%
DATA_PATH (*)	8.8%
PRGRM_CNT_TOP (*)	6.6%
PRGRM_CNT	2.2%
PRGRM_DECODE	2.2%
PRGRM_FSM	0.9%
CONTROL (*)	5.3%
INSTRN_LAT (*)	8.7%

ALU(*)

18.4%

The following example creates partitions on designs that are in hierarchical level one (subdesigns of the top-level design) and uses **-force** to enable the partition since there are multiple instantiated designs.

```
prompt> set_compile_partitions -level 1 -force
```

SEE ALSO

```
acs_compile_design(2)  
acs_recompile_design(2)  
acs_refine_design(2)  
report_partitions(2)  
acs_autopart_max_area(3)  
acs_autopart_max_percent(3)  
acs_use_autopartition(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

Congestion optimization and reporting is available with Design Compiler Graphical.

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 **write_context** 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 **dc_allocate_budgets** command, and then written out using the **write_context** 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)`
`dc_allocate_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

```
compile(2)  
current_design(2)  
remove_attribute(2)  
report_compile_options(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

```
compile(2)  
current_design(2)  
group_path(2)  
remove_attribute(2)  
report_constraint(2)  
report_path_group(2)  
reset_design(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

`get_cts_scenario(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_datapath_optimization_effort

Sets the `datapath_optimization_effort` attribute on specified designs, cells, or references, indicating the level of datapath optimization during `compile_ultra`.

SYNTAX

```
status set_datapath_optimization_effort  
object_list  
effort_level
```

Data Types

<code>object_list</code>	list
<code>effort_level</code>	string

ARGUMENTS

`object_list`
Specifies a list of cells, references, or designs for the attribute to be set.

`effort_level`
Specifies the effort level with which to set the `datapath_optimization_effort` attribute. Valid values are: **high**, **medium**, **low**. The default value is **high**.

DESCRIPTION

This command sets the `datapath_optimization_effort` attribute on the specified objects so that different levels of datapath optimization can be applied during `compile_ultra`. The `compile` command does not ungroup cells or designs with the `datapath_optimization_effort` attribute.

This attribute is removed with the `remove_attribute` or `reset_design` command.

EXAMPLES

In the following example, the `datapath_optimization_effort` attribute is set to **medium** on the `U1` cell:

```
prompt> set_datapath_optimization_effort U1 medium
```

If the `datapath_optimization_effort` attribute is specified on a reference object, cells using that reference are not ungrouped during `compile`. In this example, all instances of `ADDER` in the current design are not ungrouped:

```
prompt> set_datapath_optimization_effort [get_references ADDER] medium
```

A design with the attribute set to **high** is ungrouped whenever `compile` encounters the design:

```
prompt> set_datapath_optimization_effort [get_designs ND17] high
```

SEE ALSO

```
compile(2)  
current_design(2)  
remove_attribute(2)  
reset_design(2)
```

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.

set_default_driving_cell

-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 *value* is to apply only to minimum capacitance.

-max
Indicates that *value* 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 *value* is to apply only to pin capacitance.

-wire_load
Indicates that *value* is to apply only to wire capacitance.

-none
Indicates that previous information set by **set_default_load** is to be removed.

value
Specifies the capacitance 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 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. This command is supported only in topographical mode.

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.

-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.

`-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
1442
```

```
-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_dft_clock_controller

Specifies clock controller parameters for DFT insertion.

SYNTAX

```
int set_dft_clock_controller  
[-cell_name controller_cell_name]  
-design_name controller_design_name  
[-ateclocks port_list]  
[-pllclocks pin_list]  
[-chain_count number_chains]  
[-cycles_per_clock number_cycles]  
[-test_mode_port port_name]
```

ARGUMENTS

- cell_name** *controller_cell_name*
Specifies the instance name of the clock controller that will be inserted during insert_dft (for example, pll_controller1).
- design_name** *controller_design_name*
Only **snps_clk_mux** should be specified to insert a clock controller from the DFT library. Custom-designed clock controllers are not supported by this command. Note that the command is accepted with no -design_name but gives bad data to the DFT architect.
- ateclocks** *port_list*
One slow clock used for scan shifting that you want to connect to the DFT clock controller. If the clock signal is a reference clock for the on-chip clock generator, the signal must be previously defined by using the **set_dft_signal -type Oscillator** command. Note that you can only specify one clock because custom-designed clock controllers are not supported by this command.
- pllclocks** *pin_list*
An ordered list of fast clocks used for scan capture that you want to connect to the clock controller. The signals must be previously defined using the **set_dft_signal -type Oscillator** command.
- chain_count** *number_chains*
Specifies the number of clock chains to insert per clock controller. If not specified, defaults to 1.
- cycles_per_clock** *number_cycles*
Specifies the maximum number of capture cycles per clock. If not specified, defaults to 2. Capture cycles are cycles where capture clocks are pulsed.
- test_mode_port** *port_name*
Specifies which port of type TestMode is used to drive the test mode pin of the clock controller cell.

DESCRIPTION

The **set_dft_clock_controller** command specifies the clock controller characteristics for **insert_dft** to insert and connect clock controllers and clock chains into the design.

A clock controller is a design that controls which clock signals will propagate into scan chains during scan shifting and capture. It selects between fast clocks coming from an on-chip clock generator and slow clocks coming from external ports.

The clock signals of all the sequential elements of the design must be already connected, either directly or indirectly, to an on-chip clock generator (usually a phase-locked loop).

A clock chain is a scan chain segment that enables internal clocks during capture cycles.

Clock controller insertion supports the following scenario:

A. You want to insert a clock controller and clock chain from the Synopsys DesignWare library.

In this case, **insert_dft**

1. Instantiates a clock controller design and clock chain design from Synopsys DesignWare library
2. Inserts the clock controller in the clock path between the on-chip clock generator and the sequential cells that it drives
3. Connects the clock chain to the clock controller
4. Proceeds with the remainder of DFT insertion.

Note that the clock controller design from the DFT library should be viewed as an example implementation only; you should validate that the clock controller matches your design requirements.

EXAMPLES

Scenario A. Insertion of clock controller and clock chain from the DFT library

```
prompt> set_dft_configuration -clock_controller enable
prompt> set_dft_signal -view existing_dft -type ScanClock \
    -port slow_clock -timing {45 55}
prompt> set_dft_signal -view existing_dft -type Oscillator \
    -port slow_clock
prompt> set_dft_signal -view existing_dft -type Oscillator \
    -hookup_pin PLL/pll_fast_clock
prompt> set_dft_clock_controller -cell_name snps_pll_controller \
    -design_name snps_clk_mux -cycles_per_clock 2 -chain_count 1 \
    -pllclocks {PLL/pll_fast_clock} -ateclocks {slow_clock}
```

SEE ALSO

```
set_dft_signal(2)
set_dft_configuration(2)
insert_dft(2)
```

```
set_dft_clock_controller
1446
```

set_dft_configuration

Sets the DFT configuration for the current design.

SYNTAX

```
int set_dft_configuration
[-scan enable | disable]
[-fix_clock enable | disable]
[-fix_set enable | disable]
[-fix_reset enable | disable]
[-fix_xpropagation enable | disable]
[-fix_bus enable | disable]
[-fix_bidirectional enable | disable]
[-control_points enable | disable]
[-observe_points enable | disable]
[-logicbist enable | disable]
[-wrapper enable | disable]
[-boundary enable | disable]
[-bsd enable | disable]
[-clock_controller enable | disable]
[-mode_decoding_style binary | one_hot]
[-scan_compression enable | disable]
[-pipeline_scan_data enable | disable]
[-connect_clock_gating enable | disable]
[-integration enable | disable]
```

ARGUMENTS

-scan enable | disable

Enables or disables the SCAN utility.
The default value is **enable**.

-fix_clock enable | disable

Enables or disables the clock AutoFix utility. By default, clocks are not autofixed.

-fix_set enable | disable

Enables or disables the set AutoFix utility. By default, sets are not autofixed.

-fix_reset enable | disable

Enables or disables the reset AutoFix utility. By default, resets are not autofixed.

-fix_xpropagation enable | disable

Enables or disables the X-propagation AutoFix utility. By default, X-propagations are not autofixed.

-fix_bus enable | disable

Enables or disables the bus AutoFix utility. By default, busses are autofixed.

-fix_bidirectional enable | disable
Enables or disables the bidirectional disabling AutoFix utility. By default, bidirectionals are autofixed.

-control_points enable | disable
Enables or disables the control test point insertion utility. By default, the control point insertion utility is disabled. Control test points are not supported with LSSD scan style.

-observe_points enable | disable
Enables or disables the observe test point insertion utility. By default, the observe point insertion utility is disabled. Observe test points are not supported with LSSD scan style.

-logicbist enable | disable
Enables or disables the BIST utility, so that you can insert and connect the BIST controller.

-wrapper enable | disable
Enables or disables the Core wrapper and Shadow LogicDFT utilities. The specifications for the utilities are set with **set_wrapper_configuration** command.

-boundary enable | disable
Enables or disables the integration of BIST or scan cores using ANSI/IEEE Std 1149.1-1 compliant boundary-scan circuitry. The newly-synthesized circuitry is unmapped, and must be mapped using the **compile** command.

-bsd enable | disable
Enables or disables the insertion of IEEE Std 1149.1/1149.6 compliant boundary-scan circuitry.

-clock_controller enable | disable
Enables or disables the clock controller for on chip clocking.

-mode_decoding_style binary | one_hot
Specifies the type of port decoding to be used for the test modes controller. Allowed values are **binary** and **one_hot**. Specify **binary** to decode ports for the test controller or **one_hot**, to not decode ports for the test controller. If *n* test modes are specified for the design, specifying **binary** allows mode architecting to use $\text{ceiling}(\log_2(n))$ ports for the test controller. The **one_hot** mode decoding style allows mode architecting to use *n-1* ports for the test controller.
The default value is **binary**.

-scan_compression enable | disable
Enables or disables the insertion of scan compression structures.

-pipeline_scan_data enable | disable
Enables or disables the insertion of pipeline scan data registers to scan chains. By default, the pipeline scan data utility is disabled.

-connect_clock_gating enable | disable
Enables or disables checking for unconnected test pins of clock gating cells during **dft_drc** and their connection in **insert_dft**. The default value is

enable.

DESCRIPTION

The **set_dft_configuration** command specifies the DFT configuration for a design. The DFT configuration is specific to the current design. If you change the DFT configuration and then change the current design, your changes are no longer visible.

Use the **report_dft_configuration** command to display the current DFT configuration of the design.

Use the **reset_dft_configuration** command to reset the current DFT configuration on the design.

EXAMPLES

The following command enables the clock Autofix utility:

```
prompt> set_dft_configuration -fix_clock enable
```

The following command disables the check for unconnected test pins of clock gating cells during **dft_drc** and their connection in **insert_dft**:

```
prompt> set_dft_configuration -connect_clock_gating disable
```

SEE ALSO

```
insert_dft(2)  
report_dft_configuration(2)  
set_autofix_configuration(2)  
set_autofix_element(2)  
set_dft_signal(2)
```

set_dft_connect

Specifies a list of DFT connectivity associations. The specified source is connected to objects specified in the target list.

SYNTAX

```
status set_dft_connect  
string_label  
-type clock_gating_control | scan_enable  
-source port_or_pin  
[-target object_list]  
[-rise_target rise_clock_list]  
[-fall_target fall_clock_list]  
[-exclude cell_list]
```

Data Types

<i>string_label</i>	string
<i>port_or_pin</i>	string
<i>object_list</i>	string
<i>rise_clock_list</i>	string
<i>fall_clock_list</i>	string
<i>cell_list</i>	string

ARGUMENTS

string_label

Specifies an identifier for the connectivity association. Each association must be unique. If the same label is specified with two connectivity associations, the earlier specification is overwritten.
This argument is required.

-type clock_gating_control | scan_enable

Specifies the type of connectivity association described by the command. If **-type** is specified as **scan_enable**, you must also use either the **-target**, **-rise_target**, or **-fall_target** option. If **-type** is specified as **clock_gating_control**, the **-target** option is not required.
This argument is required.

-source *port_or_pin*

Specifies a DFT signal driver port or pin to be connected to the target cells. For a DFT signal with a hookup pin, its port must be specified as driver. For an internal pin DFT signal, the driver must be the hookup pin. For **-type clock_gating_control**, the driver must be defined as ScanEnable or TestMode with the **-usage clock_gating** switch of the **set_dft_signal** command. For **-type scan_enable**, the driver must be defined as ScanEnable with the **-usage scan** switch of the **set_dft_signal** command.
This argument is required.

-target *object_list*

Specifies a list of objects to be connected to the specified DFT signal driver port or pin. Specified objects can be a list of cells, designs, clocks, or

set_dft_connect

ScanEnable/TestMode pin/port on CTL modeled cells.

For **-type clock_gating_control**, the *cell_list* must be a list of either clock gating cells, designs containing clock gating cells, clock gating observation cells, or list of clocks *clock_list*.

For **-type scan_enable**, the *cell_list* must be a list of either valid scan cells or designs containing valid scan cells.

A Specified pin/port can be only a clock or pin (ScanEnable/TestMode) on a CTL modeled cell.

When a clock or list of clocks *clock_list* is specified, the tool selects appropriate elements in the respective clock domain, based on the connectivity type. Specified clocks must be valid DFT clocks defined with the **set_dft_signal** command. In addition, for **-type clock_gating_control**, the clocks must be defined with the **create_clock** command.

When doing clock-domain-based scan enable connections, the CTL model of the modeled cell/design must contain the relation between a Scan enable and the clock domain. DFT Compiler puts this relation in the model when doing scan insertion at the block level. For hierarchical scan synthesis, this relation is utilized to perform clock-domain-based connections at the top level. If the relation does not exist in the block level CTL model, then the user must specify appropriate pins on the block, leaving the CTL model to do the connections.

If the pins/ports on the CTL modeled cells are specified, then (1) the physical connectivity inside the CTL modeled cell is not validated, and (2) the user must determine the valid polarity/active states of the pins/ports on the CTL model cells and specify connectivity appropriately.

If **-target** is not specified, then the tool selects appropriate design elements based on **-type** in the current design.

Pure combinational cells and CTL modeled cells will be ignored and their specifications will be discarded.

-rise_target *rise_clock_list*

Specifies a list of clocks. All scan cells triggered by the rising edge of the clock will be connected to the specified DFT signal driver port or pin. This option is valid only with **-type scan_enable**. Clocks specified in the *rise_clock_list* must be defined as a valid DFT signal clock with **set_dft_signal**.

-fall_target *fall_clock_list*

Specifies a list of clocks. All scan cells triggered by the falling edge of the clock will be connected to the specified DFT signal driver port or pin. This option is valid only with **-type scan_enable**. Clocks specified in the *fall_clock_list* must be defined as a valid DFT signal clock with **set_dft_signal**.

-exclude *cell_list*

Specifies a list of cells to be excluded from the list of cells that will be connected to the specified driver port or pin.

DESCRIPTION

The **set_dft_connect** command specifies a DFT connectivity association. It is used to connect a DFT signal driver port or pin to a list of specified target cells when the **insert_dft** command is run.

Connections are done sequentially. First, connections are done for **-type clock_gating_control** and then for **-type scan_enable**. And for each type, connections are again done sequentially, so that subsequent **set_dft_connect** commands take precedence over earlier ones. Hence, the user must maintain order and precedence by specifying the **set_dft_connect** commands in right order.

Use the **report_dft_connect** command to display specified connectivity associations.

Use the **remove_dft_connect** command to remove specified connectivity associations.

EXAMPLES

The following example first sets the signals for ports SE1 through SE7, and then connects the sources to the targets:

```
prompt> set_dft_signal -view spec -type ScanEnable -port SE1 \
           -usage clock_gating

prompt> set_dft_signal -view spec -type ScanEnable -port SE2 \
           -usage clock_gating

prompt> set_dft_signal -view spec -type ScanEnable -port SE3 \
           fb-usage clock_gating

prompt> set_dft_signal -view spec -type ScanEnable -port SE4 -usage scan

prompt> set_dft_signal -view spec -type ScanEnable -port SE5 -usage scan

prompt> set_dft_signal -view spec -type ScanEnable -port SE6 -usage scan

prompt> set_dft_signal -view spec -type ScanEnable -port SE7 -usage scan

prompt> set_dft_connect label1 -source SE1 -type clock_gating_control \
           -target [list U1 U2]

prompt> set_dft_connect label2 -source SE2 -type clock_gating_control \
           -target [list U1/clk_gate*] -exclude U1/clk_gate_1

prompt> create_clock -p 5 clk1 -name CLK1

prompt> set_dft_signal -view exist -type ScanClock -port clk1 \
           -timing {45 55}

prompt> set_dft_connect label3 -source SE3 -type clock_gating_control \
           -target [list CLK1]

prompt> set_dft_connect label4 -source SE4 -type scan_enable \
           -target [list U1]

prompt> set_dft_connect label5 -source SE4 -type scan_enable \
           -target [list U2] -exclude U2/ff1

prompt> set_dft_signal -view exist -type ScanClock -port clk2 \
```



```
-timing {45 55}
```

```
prompt> set_dft_signal -view exist -type ScanClock -port clk3 \  
-timing {45 55}
```

```
prompt> set_dft_signal -view exist -type ScanClock -port clk4 \  
-timing {45 55}
```

```
prompt> set_dft_connect label6 -source SE5 -type scan_enable \  
-target [list clk2]
```

```
prompt> set_dft_connect label7 -source SE6 -type scan_enable \  
-rise_target [list clk2]
```

```
prompt> set_dft_connect label8 -source SE7 -type scan_enable \  
-fall_target [list clk3 clk4]
```

The following example connects a dedicated ScanEnable/TestMode on a CTL modeled cell to SE8, SE9, and TM at top level. Cell "core" has CTL model with core/SE_scan and core/SE_cg as dedicated ScanEnable signals, and core/TM_cg as dedicated TestMode signal.

```
prompt> set_dft_signal -view spec -type ScanEnable -port SE8 \  
-usage clock_gating
```

```
prompt> set_dft_signal -view spec -type ScanEnable -port SE9 \  
-usage scan
```

```
prompt> set_dft_signal -view spec -type TestMode -port TM \  
-usage clock_gating
```

```
prompt> set_dft_connect label9 -source SE8 -type clock_gating_control \  
-target core/SE_cg
```

```
prompt> set_dft_connect label10 -source SE9 -type scan_enable \  
-target core/SE_scan
```

```
prompt> set_dft_connect label11 -source TM -type clock_gating_control \  
-target core/TM_cg
```

SEE ALSO

```
remove_dft_connect(2)  
report_dft_connect(2)  
set_dft_signal(2)
```

set_dft_drc_configuration

Sets the DFT DRC configuration for the current design.

SYNTAX

```
int set_dft_drc_configuration  
[-internal_pins enable | disable]  
[-pll_bypass enable | disable]  
[-assume_pi_scan enable | disable]  
[-assume_po_scan enable | disable]  
[-static_x_analysis enable | disable]  
[-use_test_model true | false]  
[-use_test_mode core_mode_list]  
[-clock_gating_init_cycles integer]  
[-allow_se_set_reset_fix true | false]
```

ARGUMENTS

-internal_pins enable | disable

Enable the Internal Pins Flow where the internal pins can be specified as clock, reset, set, constant, scan-in, scan-out, scan-enable and test_mode signals in XG mode.

-pll_bypass enable | disable

Enable the pll bypass mode during the post insert_dft DRC.

-assume_pi_scan enable | disable

This applies only when BIST is enabled. When this option is enabled (default), Design Rule Checking will not check that all primary inputs are driven by scan elements, and just assumes that this is true. If you want Design Rule Checking to check that all primary inputs are driven by scan elements, set this option to "disable". Design Rule Checking will then report L-24 violations on the primary inputs that are not driven by scan elements.

-assume_po_scan enable | disable

This applies only when BIST is enabled. When this option is enabled (default), Design Rule Checking will not check that all primary outputs drive scan elements, and just assumes that this is true. If you want Design Rule Checking to check that all primary outputs drive scan elements, set this option to "disable".

-static_x_analysis enable | disable

When this option is enabled, Design Rule Checking will report static x scan cells violations. The default value is disable.

-use_test_model true | false

Instructs dft_drc to replace sub-blocks gate representations by the test model representations. The default value is true.

-use_test_mode *core_mode_list*

This applies only when design contains multi-mode core. When this option is enabled, Design Rule Checking will check only cores defined in the list under

a specified mode. The list contains an ordered list of instance cores and test modes (see example below).

-clock_gating_init_cycles *integer*

Automatically update the test_setup section of the test protocol with the integer number of clock pulses. This is useful for initialization of multiple cascaded clock-gating latches and flip-flops.

-allow_se_set_reset_fix true | false

Instructs dft_drc for cases where user has bypass logic for internally generated reset using scan_en if the value is set to true. The switch can be used to include such scan flops in the scan chain which would be otherwise marked violated and excluded from the scan chains. The default value is false.

DESCRIPTION

This command specifies the configuration for the DFT DRC. When **-internal_pins** is set to **enable**, the user can specify internal pins to be as clock, reset, set, constant, scan-in, scan-out, scan-enable and test_mode signals. The user can specify these internal pins using the **set_dft_signal** and **set_scan_path**. These signals are then used by dft_drc, preview_dft and insert_dft. However, the protocol generated at the end of post dft_drc is not complete and cannot be used by the user. When **-pll_bypass** is set to **enable**, Design Rule Checking after insert_dft is run in pll bypass mode. When **-assume_pi_scan** and **-assume_po_scan** are disabled, Design Rule Checking does not assume scanned primary inputs and outputs. When **-use_test_model** is set to FALSE, dft_drc uses the gates instead of a test model attached to a block. For blocks containing scan chains and complex test control structures, it is recommended to use the gates. The test model may not describe accurately the test control logic.

EXAMPLES

The following example enables the internal clocks support:

```
prompt> set_dft_drc_configuration -internal_pins enable
The following example enables the user test mode definition for core:
prompt> set_dft_drc_configuration -
use_test_mode {inst_1 mode_A inst_2 mode_B}
The following example sets the initialization clock pulses
prompt> set_dft_drc_configuration -
clock_gating_init_cycles N where, N = no of clock pulses
```

SEE ALSO

```
set_dft_configuration(2)
set_scan_configuration(2)
set_dft_signal(2)
set_scan_path(2)
write_test_protocol(2)
```

set_dft_drc_rules

Sets the severity on the allowed set of violations.

SYNTAX

```
int set_dft_drc_rules  
[-error drc_error_ID]  
[-warning drc_error_ID]  
[-ignore drc_error_ID]  
[-cell cell_list]
```

Data Types

<i>drc_error_ID</i>	string
<i>cell_list</i>	string

ARGUMENTS

- error *drc_error_ID***
Specifies explicitly to change the severity of the specified violations to ERROR. Changing the severity to ERROR causes the **dft_drc** to report the specified violations as error and the **insert_dft** to fail.
- warning *drc_error_ID***
Specifies explicitly to change the severity of the specified violations to WARNING. Changing the severity to WARNING causes the violation to be ignored by **insert_dft**. However, **dft_drc** still reports the violations for the user to keep track of the violation.
- ignore *drc_error_ID***
Specifies explicitly to change the severity of the specified violations to IGNORE. Changing the severity to IGNORE causes the violation to be ignored by both **dft_drc** and **insert_dft**.
- cell *cell_list***
Specifies the cells for which the violation severity is to be changed. However, if no cells are specified, the violation severity is changed for all the cells in the design.

DESCRIPTION

The **set_dft_drc_rules** command can be used to change the severity of the violations reported by **dft_drc**. However, user can change the severity only if the severity of that violation is allowed to be changed. The severity of the cells that are set using the **set_test_assume** command cannot be changed using the **set_dft_drc_rules** command. A list of valid violations is shown below.

```
DRC violation ID  
-----  
TEST-504  
TEST-505
```

Use the **report_dft_drc_rules** command to display the violations whose severity has been changed. To reset the severity of the violation to its default value, use **reset_dft_drc_rules** command.

EXAMPLES

The following example changes the severity of drc violation TEST-504 for all cells to warning, TEST-505 for cells reg3, reg4 to warning and TEST-505 for cell reg6 to ignore:

```
prompt> set_dft_drc_rules -warning {TEST-504}
prompt> set_dft_drc_rules -warning {TEST-505} -cell {reg3 reg4}
prompt> set_dft_drc_rules -ignore {TEST-505} -cell {reg6}
```

SEE ALSO

```
reset_dft_drc_rules(2)
report_dft_drc_rules(2)
dft_drc(2)
insert_dft(2)
```

set_dft_equivalent_signals

Sets a given list of DFT signals as equivalents.

SYNTAX

```
integer set_dft_equivalent_signals  
signal_list
```

Data Types

signal_list list

ARGUMENTS

signal_list
Lists the signals to check against the primary signal for equivalency requirements. The first signal in the list is the primary signal. The primary signal is used to check for valid equivalences with the remaining signals in the list for type and other (if any) equivalency requirements. All the signals must have the same signal type.

DESCRIPTION

This command sets a specified list of scan signals as equivalent for scan architect. This is supported only for clock signal types (ScanClock, MasterClock, and ScanMasterClock), as specified by the **set_dft_signal** command or inferred by the **create_test_protocol** command.

Use this command to specify a set of clocks as equivalent for scan architect. This enables the architect to consider all flops driven by these clocks as if driven by the same clock, and the architect will not add any synchronization elements when they are mixed in a scan chain.

EXAMPLES

The following example illustrates setting two clocks as equivalent to clock clk1:

```
prompt> set_dft_equivalent_signals [list clk1 clk2 clk3]
```

SEE ALSO

```
insert_dft(2)  
preview_dft(2)  
remove_dft_equivalent_signals(2)  
report_dft_equivalent_signals(2)  
set_dft_signal(2)
```

set_dft_insertion_configuration

Sets the DFT insertion configuration for the current design.

SYNTAX

```
int set_dft_insertion_configuration  
[-map_effort low | medium | high]  
[-synthesis_optimization none | all]  
[-route_scan_enable true | false]  
[-route_scan_clock true | false]  
[-route_scan_serial true | false]  
[-preserve_design_name true | false]  
[-unscan true | false]
```

ARGUMENTS

- map_effort** low | medium | high
Specifies the optimization effort level during DFT insertion. Valid values are **low**, **medium**, and **high**.
The default value is **medium**.
- synthesis_optimization** none | all
Specifies whether to perform optimizations during DFT insertion. Valid values are **none** to turn off all optimizations, and **all** to turn on all optimizations.
The default value is **all**.
- route_scan_enable** true | false
Specifies whether to route scan enables during DFT insertion. Valid values are **true** to perform scan enable routing, and **false** to turn off scan enable routing.
The default value is **true**.
- route_scan_clock** true | false
Specifies whether to route scan clocks during DFT insertion. Valid values are **true** to route the scan clocks, and **false** to turn off scan clock routing.
The default value is **true**.
- route_scan_serial** true | false
Specifies whether to route scan serial signals during DFT insertion. Valid values are **true** to route the scan serial signals, and **false** to turn off scan serial routing.
The default value is **true**.
- preserve_design_name** true | false
Specifies whether to preserve the design name when writing from the tool to the database during DFT insertion. Valid values are **true** to preserve the design name, and **false** to permit renaming the design.
The default value is **false**.
- unscan** true | false
Specifies whether to unscan cells during DFT insertion. Valid values are **true** to unscan the cells, and **false** to turn off cell unscanning.

The default value is **false**.

DESCRIPTION

The **set_dft_insertion_configuration** command specifies the configuration to use for the current design during the DFT insertion process. The settings apply only to the current design. If you change the DFT insertion configuration and then change the current design, your changes are no longer visible.

Use the **report_dft_insertion_configuration** command to display the current DFT insertion configuration of the design.

Use the **reset_dft_insertion_configuration** command to remove the current DFT insertion configuration from the design.

EXAMPLES

The following command sets the mapping effort level to **high**:

```
prompt> set_dft_insertion_configuration -map_effort high
```

The following command turns off all synthesis optimizations:

```
prompt> set_dft_insertion_configuration \  
-synthesis_optimization none
```

SEE ALSO

```
insert_dft(2)  
report_dft_insertion_configuration(2)  
reset_dft_insertion_configuration(2)
```


set_dft_location

Specifies the DFT hierarchy location for DFT insertion.

SYNTAX

```
status set_dft_location
dft_hier_name
[-
exclude DFTMAX | TCM | LOCKUP_LATCH | PIPELINE_SI_LOGIC | PIPELINE_SE_LOGIC | WRAPP
ER]
[-
include DFTMAX | TCM | LOCKUP_LATCH | PIPELINE_SI_LOGIC | PIPELINE_SE_LOGIC | WRAPP
ER]
```

Data Types

dft_hier_name string

ARGUMENTS

dft_hier_name
Specifies the hierarchical path name of an instance in the current design into which all DFT logic will be added during DFT insertion. This argument is required.

-exclude DFTMAX | TCM | LOCKUP_LATCH | PIPELINE_SI_LOGIC | PIPELINE_SE_LOGIC | WRAPPER
Specifies the types of DFT logic to be excluded from the specified DFT location. The excluded DFT logic is added to its default location. More than one type of DFT logic can be excluded with this option. There is no default value for this option. When this option is not specified, all types of DFT logic are added to the user-specified location.

-include DFTMAX | TCM | LOCKUP_LATCH | PIPELINE_SI_LOGIC | PIPELINE_SE_LOGIC | WRAPPER
Specifies the types of DFT logic to be included in the specified DFT location. More than one type of DFT logic can be included with this option. There is no default value for this option. When this option is not specified, all types of DFT logic are added to the user-specified location.

DESCRIPTION

The **set_dft_location** command can be used to add all DFT logic into a user-specified hierarchy during DFT insertion.

EXAMPLES

The following example specifies a DFT hierarchy location for the design top.

```
prompt> set_dft_location core_inst/dft_inst
Accepted dft location specification.
1
```

The following example specifies a DFT hierarchy location for the design top. All Wrapper and Lockup latch logic should not be part of this location.

```
prompt> set_dft_location core_inst/dft_inst -exclude {WRAPPER LOCKUP_LATCH}  
Accepted dft location specification.  
1
```

The following example specifies a DFT hierarchy location for Wrapper logic, another dft location for DFTMAX logic. Lockup latch logic should not be part of these locations.

```
prompt> set_dft_location core_inst/dft_inst1 -include {WRAPPER} -  
exclude {LOCKUP_LATCH}  
Accepted dft location specification.  
1  
prompt> set_dft_location core_inst/dft_inst2 -include {DFTMAX} -  
exclude {LOCKUP_LATCH}  
Accepted dft location specification.  
1
```

SEE ALSO

```
insert_dft(2)  
remove_dft_location(2)  
report_dft_location(2)
```

set_dft_signal

Specifies the DFT signal types for DRC and DFT insertion.

SYNTAX

```
status set_dft_signal
[-view existing_dft | spec]
[-test_mode mode_name]
-type signal_type
[-port port_list]
[-active_state active_state]
[-timing timing]
[-period period]
[-hookup_pin hookup_pin]
[-hookup_sense hookup_sense]
[-internal_clocks none | single | multi]
[-ctrl_bits ctrl_bits_list]
[-pll_clock pll_clock]
[-ate_clock ate_clock]
[-differential_clock clock_port]
[-connect_to pin_list]
[-usage use_type]
[-associated_internal_clocks clock_pins_list]
[-associated_clock associated_clock]
```

Data Types

<i>mode_name</i>	string
<i>signal_type</i>	string
<i>port_list</i>	list
<i>active_state</i>	integer
<i>timing</i>	list
<i>period</i>	float
<i>hookup_pin</i>	list
<i>hookup_sense</i>	string
<i>ctrl_bits_list</i>	list
<i>pll_clock</i>	string
<i>ate_clock</i>	string
<i>clock_port</i>	string
<i>pin_list</i>	list
<i>use_type</i>	list
<i>associated_clock</i>	string

ARGUMENTS

-view existing_dft | spec

Indicates the view to which the specification applies. The following views are valid:

- **existing_dft** implies that the specification refers to the existing usage of a port. For example, when working with a DFT-inserted design, the command to indicate that port A is used as a scan enable follows this syntax:

set_dft_signal -view existing_dft -port A -type ScanEnable

- **spec** (the default value) implies that the specification refers to ports that the tool must use during DFT insertion. For example, when preparing a design for DFT insertion, the command to specify that port A is used as a test enable follows this syntax:

set_dft_signal -view spec -port A -type ScanEnable

-test_mode mode_name

Specifies the mode to which the specification applies. The default mode is **all_dft**. Specifying **all** as the *mode_name* implies that the specification applies to all modes. Any mode other than the default mode or **all** must be created before it can be used.

-type signal_type

Specifies the signal type. The valid signal types are as follows:

- **Reset** is the asynchronous set or reset of the design.
 - **Constant** is a continuously applied value to a port.
 - **TestMode** is a continuously applied value to a test mode port that may have different values between test modes.
 - **TestData** is the port used to apply scan data to a portion of the design that has had pre-DRC violations fixed by Autofix.
 - **ScanDataIn** is one of the scan inputs of the design.
 - **ScanDataOutt** is one of the scan outputs of the design.
 - **ScanMasterClock** is the clock responsible for the shift and capture in a MUX-D design. In an LSSD scan style the ScanMasterClock is the A clock, and in a clocked scan style it is the shift clock.
 - **ScanSlaveClock** is the B clock in an LSSD scan style.
 - **ScanEnable** is the shift enable of a scan MUX-D design.
 - **MasterClock** is the clock responsible for shift and capture in a MUX-D design. In an LSSD design the MasterClock is the system clock, and in a clocked scan style it is the capture clock.
 - **Oscillator** is a constantly running clock that never turns off. For example, the PLL source clock has no specific timing.
 - **RefClock** is a constantly running external clock. The clock can have timing other than the test default period.
- The ScanClock signal type can be used as a shortcut to simultaneously specify both MasterClock and ScanMasterClock types. This is useful for the multiplexed flip-flop scan style. In general, the MasterClock is used for referring to the capture clock and ScanMasterClock for referring to the shift clock. Both of these signal types refer to the same signal in the multiplexed flip-flop scan style (MUX-D). In the LSSD scan style, the two shift clocks are specified as ScanMasterClock and ScanSlaveClock, and only the capture

clock is specified as MasterClock.

The Oscillator signal type is used to specify the output pins of the on-chip clock generator or the output pins of the user instantiated on-chip clock controller. In both cases, **-hookup_pins** must be used.

The RefClock signal type is used to specify On-Chip Clocking reference clocks.

To specify the sense for an internal hookup only pin (no port is associated with the hookup pin), use the **-active_state** option. The following signal types are used in BSD synthesis and integration:

- **tdi** specifies the TDI port or bsd reg tdi access pin
- **tdo** specifies the TDO port or bsd reg tdo access pin
- **tck** specifies the TCK port
- **tms** specifies the TMS port
- **trst** specifies the TRST port
- **bsd_shift_en** specifies the register access pin to be hooked up to TAP shift_dr pin when the instruction that selects the register is active.
- **bsd_capture_en** specifies the register access pin to be hooked up to TAP sync_capture_en pin when the instruction that selects the register is active. This signal is also used in core integration.
- **bsd_capture_dr** specifies the register access pin to be hooked up to TAP Capture-DR state on the negative edge of TCK when the instruction that selects the register is active.
- **bsd_update_en** specifies the register access pin to be hooked up to TAP sync_update_dr pin when the instruction that selects the register is active. This signal is also used in core integration.
- **bsd_update_dr** specifies the register access pin to be hooked up to TAP Update-DR state on the negative edge of TCK when the instruction that selects the register is active.
- **capture_clk** specifies the register access pin to be hooked up to TCK/clock_dr pin when the instruction that selects the register is active.
- **update_clk** specifies the register access pin to be hooked up to TCK/update_dr pin when the instruction that selects the register is active.
- **inst_enable** specifies the register access pin to be held active when the instruction that selects the register is active.
- **bist_enable** specifies the register access pin to be held active when the instruction that selects the register is active and TAP is in Run-Test-Idle state.
- **bist_clk** specifies the register access pin to be hooked up to TCK when the instruction that selects the register is active and TAP is in Shift-DR state. This signal is also used in core integration.

- **bsd_reset** specifies the register access pin to be hooked up to TAP Test-Logic-Reset state when the instruction that selects the register is active. When the bsd_reset signal is not part of any instruction's test data register, it is hooked up to TAP Test-Logic-Reset state.

-port *port_list*

Indicates the list of ports on which to apply the specifications.

-active_state *active_state*

Specifies the active states for the following signal types:

ScanEnable

Reset

Constant

TestMode

The active state can be 0 or 1 and it specifies the active sense of the port (high or low), or an internal hookup only pin (no port is associated with the hookup pin).

-timing *timing*

Specifies the rise time and fall time for clocks. For example:

```
set_dft_signal -view existing_dft -type MasterClock \  
-port CLK -timing [list 45 55]
```

-period *period*

Specifies the period of the On-Chip Clocking reference clock. The period value is a floating point number. The time unit is nanosecond. When **-period** is used, the values specified after **-timing** indicate the leading edge and the trailing edge of the reference clock. The **-period** option is valid only on signals of the RefClock type.

-hookup_pin *hookup_pin*

Identifies a specific pin to which wires are to be connected. The **insert_dft** command connects wires to the core side of identified signal ports, jumping pads, and buffers as needed. The **-hookup_pin** argument overrides this behavior and instructs **insert_dft** to connect wires to the specified pin. Validation ensures that the pin direction is consistent with the signal type. Verify that a specific access pin is associated with only one design port. If hookup pins are specified, only one port can be specified in the *port_list*. However, if the Internal Pins Flow is enabled using the **set_dft_drc_configuration** command, a hookup pin can be specified without specifying a port. The hookup pin specified should be either on a black box or should have a proper global driver. If a proper global driver is not found, **insert_dft** will not stitch to the specified hookup pin. When the **-type** is a scan clock and **-view** is set to existing, the **-hookup_pin** argument allows you to specify internal pins for the scan clocks along with their associated top-level clock port specified with **-port**. The resulting protocol is accurate and complete with real top-level clocks.

-hookup_sense *hookup_sense*

Specifies the hookup sense for the hookup pin. This option is valid only when a single hookup pin or a port is specified. The valid values are **inverted** and **non-inverted**. The default value is **non-inverted**.

set_dft_signal

`-internal_clocks none | single | multi`

Specifies the setting for an internal clock. An internal clock is defined as an internal signal driven by a multiplexer (or multiple input gate) output pin. This option applies only to the multiplexed flip-flop scan style and it is ignored for other scan styles.

Note that the output of clock gating cells that are introduced by the Synopsys Power Compiler will not be treated as internal clocks, even if the **-internal_clocks** option is enabled.

The **-internal_clocks** option can be set to the following values:

- **single** specifies that **insert_dft** treats any internal clocks in the design as separate clocks for the purpose of scan chain architecting. The **single** value instructs the tool to stop at the first buffer or inverter driving the flip-flops clock.
- **none** (the default value) specifies that **insert_dft** does not treat internal clocks as separate clocks.
- **multi** specifies that **insert_dft** treats any internal clocks in the design as separate clocks for the purpose of scan chain architecting. The **multi** value instructs the tool to jump over buffers and inverters, stopping at the first multi-input gate driving the flip-flops clock.

`-ctrl_bits ctrl_bits_list`

Lists triplets that specify the sequence of control bits needed to enable the propagation of the clock generator outputs. This option is used in the On-Chip Clocking flow. The first element of each triplet is the cycle number (integer) indicating when the clock signal will be propagated. The second element is the pin name of the control bit (a valid design hierarchical pin name). The third element is the active value (0 or 1) of the control bit. For example:

```
set_dft_signal -type Oscillator -hookup_pin pll_controller/clk \  
-pll_clock pll_i/pll_clk -ate_clock ate_clk \  
-ctrl_bits [list 0 clk_chain_i/clk_ctrl_data[0] \  
1 1 clk_chain_i/clk_ctrl_data[1] 1] \  
-view existing_dft
```

`-pll_clock pll_clock`

Specifies the port name of the pll clock. This option is used in the On-Chip Clocking flow.

`-ate_clock ate_clock`

Specifies the port name of the ATE clock. This option is used in the On-Chip Clocking flow.

`-differential_clock clock_port`

Specifies the pin name of a differential clock. For example, if `clk_m` is the minus side of `clk_p`:

```
set_dft_signal -view existing_dft -type ScanClock \  
-port clk_m -differential {clk_p}
```

Note that the minus side will track all of the properties of the plus side with an inverted polarity. A complete example for `clk_m` and `clk_p` is as follows:

```

set_dft_signal -view existing_dft -type Oscillator \
  -port clk_p
set_dft_signal -view existing_dft -type ScanClock \
  -port clk_p -timing [list 45 55]
set_dft_signal -view existing_dft -type ScanClock \
  -port clk_m -differential {clk_p}

```

-connect_to *pin_list*

Specifies the correspondence between the top level ATE clock port and the OCC block ATE clock pins. This option is valid only in a Hierarchical On-Chip Clocking flow. To specify an existing connection, use the **-view existing** option. When **-view spec** is used, DFT Compiler adds the connection between the top-level ATE clock port and the OCC block ATE clock pins. For example:

```

set_dft_signal -view spec -port TOP_CLK -type MasterClock \
  -connect_to [get_pins {s1/CLK2 s0/CLK2}] -timing {45 55}

```

-usage *use_type*

Specifies the usage of the signal. You must also specify **-view spec** with this option. This option specifies that **insert_dft** is to use the specified signal for that purpose only.

The signal must be defined with suitable valid values when using it to connect design elements using the **set_dft_connect** command. The valid values for *use_type* are as follows:

- **clock_gating** specifies that the signal is to be connected to the test pin of clock gating cells during **insert_dft**. The feature to connect the unconnected test pins of clock gating cells during **insert_dft** must be enabled by using the **-connect_clock_gating** option of the **set_dft_configuration** command.

For example,

```

set_dft_signal -view spec -port SE \
  -type ScanEnable -usage clock_gating

set_dft_signal -view spec -port TM \
  -type TestMode -usage clock_gating f{

```

- **scan** specifies that the signal is to be connected to the enable of scan elements during **insert_dft**. This usage is valid only for **-type ScanEnable**. For example,

```

set_dft_signal -view spec -port SE \
  -type ScanEnable -usage scan

```

Multiple usages can be specified for a signal by specifying them as a list. For example, a ScanEnable can be used to connect to test pins of clock gating cells as well as enabling scan elements.

For example,

```

fbset_dft_signal -view spec -port SE \
  -type ScanEnable -usage {scan clock_gating}

```


`-associated_internal_clocks clock_pins_list`

Specifies the relation between a list of hierarchical pin names and a design clock port. These pin names are considered as internal clock sources by `preview_dft` and `insert_dft` to determine clock domains for the design. The top level clock, the `set_dft_signal` command applies to, is used in the test model to describe the scan clock of the chain containing scan elements triggered by these internal clock drivers. For example,

```
set_dft_signal -type ScanClock -port CLK \  
-associated_internal_clocks [list U1/int_clk U2/int_clk1 U1/int_clk2] \  
-view existing
```

`-associated_clock associated_clock`

Specifies the pin or port name of the clock that you want to associate with a specified scan enable signal. This option is used in a domain-based scan enable flow. For example,

```
set_dft_signal -type ScanEnable -port SE \  
-associated_clock U1/int_clk \  
-view spec
```

DESCRIPTION

The **set_dft_signal** command can be used to specify one or more primary input or output ports as DFT signals. It can be used either to describe the existing usage or to prescribe the usage during implementation.

A DFT signal has a port and a signal type. You can specify multiple DFT signals, but you can specify only one signal type. Port directions must be consistent with the signal type.

The **set_dft_signal** commands are not incremental. A **set_dft_signal** command that identifies a port overwrites any previous **set_dft_signal** command that identifies the same port.

To review your DFT signal specifications, use the **report_dft_signal** command. To remove a specification, use the **remove_dft_signal** command. To implement the DFT signal, use the **insert_dft** command.

Descriptive specification (using **-view existing_dft**) will invalidate test protocol and details about DFT structures.

SEE ALSO

```
current_design(2)  
dft_drc(2)  
insert_dft(2)  
preview_dft(2)  
remove_dft_signal(2)  
report_dft_signal(2)  
set_dft_configuration(2)  
set_dft_drc_configuration(2)  
write(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_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_retime

Sets the **dont_retime** attribute on cells and designs in the current design to prevent sequential cells from being moved by retiming optimizations.

SYNTAX

```
int set_dont_retime
object_list
[true | false]
```

Data Types

object_list list

ARGUMENTS

object_list
Specifies list of objects (cells or designs) on which the **dont_retime** attribute is to be set, so that the cells themselves (if they are sequential cells) or the sequential child cells of designs or hierarchical cells will not be moved by retiming optimizations, if the effective attribute value is true.

true | false
Specifies the value with which to set the **dont_retime** attribute. The default is **true**.

DESCRIPTION

This command sets the **dont_retime** attribute on cells, and designs in the current design to prevent moving of sequential cells during retiming optimizations. Setting the **dont_retime** attribute on a hierarchical cell implies "dont_retime" on all sequential cells below it that do not have **dont_retime** set to a **false** value. Setting the **dont_retime** attribute on a leaf level cell implies "dont_retime" on the cell, if it is a sequential cell. Setting the **dont_retime** attribute on a design implies "dont_retime" on all sequential cells inside that do not have **dont_retime** set to a **false** value. **dont_retime** attributes set on child cells or designs have preference over **dont_retime** attributes set on any ancestor cell or design. **dont_retime** attributes set on hierarchical cells that are instances of a design have preference over **dont_retime** attributes set on the design itself.

You can see the **dont_retime** value explicitly on cells or designs by using the **report_cell**, **report_design** or **report_attribute** command.

To remove the **dont_retime** attribute, use **remove_attribute**.

EXAMPLES

The following example specifies that the cells named *z1_reg* and *z2_reg* are not to be modified during compile.

```
prompt> set_dont_retime [get_cells {z1_reg z2_reg}] true
```

The following example specifies that the cell named *H1/z_reg* can be moved during retiming, but any other sequential cell inside cell *H1* cannot be moved.

```
prompt> set_dont_retime [get_cells "H1"] true  
prompt> set_dont_retime [get_cells "H1/z_reg"] false
```

The following example specifies that no sequential cells inside any instance of design *Controller* may be moved by retiming optimizations.

```
prompt> set_dont_retime find [get_designs "Controller"] true
```

SEE ALSO

```
compile(2)  
compile_ultra(2)  
report_cell(2)  
report_design(2)  
report_attribute(2)  
optimize_registers(2)  
balance_registers(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 is ignored on nets that have unmapped cells on them. Warnings are issued by the **compile** command for **dont_touch** nets connected to unmapped cells (generic logic).

- 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.
- Setting the **dont_touch** attribute on unmapped cells in Design Compiler topographical mode causes the **compile_ultra** command to fail during execution.
- Setting the **dont_touch** attribute on selectops and their control logic is not supported if this logic resides at the top level of the design.
- A **dont_touch** attribute on a flip-flop prevents scan replacement only on a Verilog netlist. However, it still allows scan routing of the flip-flop if it is already scan replaced. If a .ddc (Synopsys logical database format) file is used, the command **set_scan_element false** must be used to prevent scan replacement because the **dont_touch** attribute will not prevent scan replacement.

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

`compile(2)`
`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

```
compile(2)  
remove_attribute(2)  
report_cell(2)  
report_clock(2)  
report_design(2)  
report_net(2)  
report_port(2)  
set_dont_touch(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

compile(2)
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

`report_resources(2)`
`remove_dp_int_round(2)`

set_dp_smartgen_options

Controls the strategies that are used when datapath smart generation is active.

SYNTAX

```
status set_dp_smartgen_options
[-all_options auto | true | false | default]
[-booth_encoding auto | true | false]
[-booth_radix8 auto | true | false]
[-booth_mux_based auto | true | false]
[-booth_cell auto | true | false]
[-mult_nand_based auto | true | false]
[-inv_out_adder_cell auto | true | false]
[-4to2_compressor_cell auto | true | false]
[-adder_radix auto | 2 | 3 | 4]
[-ling_adder auto | true | false]
[-hybrid_adder auto | true | false]
[-carry_select_adder_cell auto | true | false]
[-cond_sum_adder auto | true | false]
[-sklansky_adder auto | true | false]
[-brent_kung_adder auto | true | false]
[-bounded_fanout_adder auto | true | false]
[-mux_based auto | true | false]
[-inv_adder_cell auto | true | false]
[-sop2pos_transformation auto | true | false]
[-tp_opt_tree auto | true | false]
[-tp_oper_sel auto | true | false]
[-smart_compare auto | true | false]
[-optimize_for default | area | speed | area,speed]
[design or cell list]
```

ARGUMENTS

- all_options auto | true | false | default**
Specifies the default value for all smart generation options. It can be used with other switches to give a specific value to one option and default values to other options.
The **-all_options** argument also accepts the value of **default**. With this value all smart generation options are set to their default state.
The value of **-all_options** is always processed before any other options.
- booth_encoding auto | true | false**
Controls whether or not Booth encoding architectures are used to implement multipliers.
When set to **auto**, the default, the option is only used if the tool determines there is a QoR benefit. When set to **true**, the option is always used, and when set to **false**, the option is never used.
- booth_radix8 auto | true | false**
Controls whether or not radix-8 Booth encoding architectures are used to implement multipliers. This option is only active if the **-booth_encoding** option is active (set to **auto** or **true**).

When set to **auto**, the default, the option is only used if the tool determines there is a QoR benefit. When set to **true**, the option is always used, and when set to **false**, the option is never used.

`-booth_mux_based auto | true | false`

Controls whether mux-based Booth encoding is used. Mux-based encoding can give better QoR for technologies with fast multiplexer cells. Otherwise xor-based Booth encoding is used.

When set to **auto**, the default, the option is only used if the tool determines there is a QoR benefit. When set to **true**, the option is always used, and when set to **false**, the option is never used.

`-booth_cell auto | true | false`

Controls the usage of special Booth multiplier cells in the target library for this compilation.

When set to **auto**, the default, the option is only used if the tool determines there is a QoR benefit. When set to **true**, the option is always used, and when set to **false**, the option is never used.

`-mult_nand_based auto | true | false`

Controls the usage of NAND-based multiplier structure. Non-Booth multipliers use AND gates or an INVERT-NOR structure to generate the partial-product bits. This option allows the use of NAND gates to produce inverted bits, which are then added directly. This can result in lower dynamic power and less glitching.

When set to **auto**, the default, the option is only used if the tool determines there is a QoR benefit and if dynamic power is optimized. When set to **true**, the option is always used, and when set to **false**, the option is never used. The option does not take effect when Booth encoding is used in multipliers.

`-inv_out_adder_cell auto | true | false`

Controls the usage of full adder cells with inverted sum and carry outputs in the target library for this compilation.

When set to **auto**, the default, the option is only used if the tool determines there is a QoR benefit. When set to **true**, the option is always used, and when set to **false**, the option is never used.

`-4to2_compressor_cell auto | true | false`

Controls the usage of 4-2 compressor cells in the target library for this compilation.

When set to **auto**, the default, the option is only used if the tool determines there is a QoR benefit. When set to **true**, the option is always used, and when set to **false**, the option is never used.

`-adder_radix auto | 2 | 3 | 4`

Controls the radix of the prefix structure in adders. Most adder architectures use a parallel-prefix structure to propagate carries. The default prefix structure has radix 2 (combines 2 bits per prefix node). Radix 3 and 4 (combine 3 or 4 bits per prefix node) result in fewer but more complex prefix nodes. They can deliver better QoR if special cells are available in the technology to implement them.

When set to **auto**, the optimal radix is automatically selected for best QoR. When set to **2, 3 4** the radix is set accordingly. The default is **2**.

-ling_adder auto | true | false
 Controls whether or not to use Ling adder architectures.
 When set to **auto**, the default, the option is only used if the tool determines there is a QoR benefit. When set to **true**, the option is always used, and when set to **false**, the option is never used.

-hybrid_adder auto | true | false
 Controls whether or not to use hybrid parallel-prefix/carry-select adder architectures (spanning-tree adder).
 When set to **auto**, the default, the option is only used if the tool determines there is a QoR benefit. When set to **true**, the option is always used, and when set to **false**, the option is never used.

-carry_select_adder_cell auto | true | false
 Controls the usage of carry select adder cells in the target library for this compilation.
 When set to **auto**, the default, the option is only used if the tool determines there is a QoR benefit. When set to **true**, the option is always used, and when set to **false**, the option is never used.

-cond_sum_adder auto | true | false
 Controls whether or not to use conditional-sum adder architectures.
 When set to **auto**, the default, the option is only used if the tool determines there is a QoR benefit. When set to **true**, the option is always used, and when set to **false**, the option is never used.

-sklansky_adder auto | true | false
 Controls whether or not to use Sklansky parallel-prefix adder architectures. This is a static architecture that is not optimized for context. It usually gives worse QoR but can be more predictable.
 When set to **auto**, the option is only used if the tool determines there is a QoR benefit. When set to **true**, the option is always used, and when set to **false**, the default, the option is never used.

-brent_kung_adder auto | true | false
 Controls whether or not to use Brent-Kung parallel-prefix adder architectures. This is a static architecture that is not optimized for context. It usually gives worse QoR but can be more predictable.
 When set to **auto**, the option is only used if the tool determines there is a QoR benefit. When set to **true**, the option is always used, and when set to **false**, the default, the option is never used.

-bounded_fanout_adder auto | true | false
 Controls whether or not to use bounded fanout adder architectures (Kogge-Stone parallel-prefix).
 When set to **auto**, the option is only used if the tool determines there is a QoR benefit. When set to **true**, the option is always used, and when set to **false**, the default, the option is never used.

-mux_based auto | true | false
 Controls whether or not mux-based architectures should be considered by datapath generators.
 When set to **auto**, the default, the option is only used if the tool determines there is a QoR benefit. When set to **true**, the option is always used, and when set to **false**, the option is never used.

-inv_adder_cell auto | true | false

Controls the usage of full adder cells with inverted carry-in or carry-out in the target library for this compilation.

When set to **auto**, the default, the option is only used if the tool determines there is a QoR benefit. When set to **true**, the option is always used, and when set to **false**, the option is never used.

-sopos_transformation auto | true | false

Controls whether "sop to pos" optimization strategies can be used in datapath blocks. An example of a "sop 2 pos" transformation is as follows:

$a*b + a*c \rightarrow (b + c) * a$

This optimization can potentially improve design area, at a possible cost in delay.

When set to **auto**, the default, the option is only used if the tool determines there is a QoR benefit. When set to **true**, the option is always used, and when set to **false**, the option is never used.

-tp_opt_tree auto | true | false

Controls whether the carry-save adder tree should be optimized for low power based on transition probabilities (TP). This optimization can reduce internal switching activity and therefore dynamic power in SOPs when the inputs have a non-uniform transition probability distribution.

When set to **auto**, the default, the option is only used if the tool determines there is a QoR benefit and if dynamic power is optimized. When set to **true**, the option is always used if dynamic power is optimized, and when set to **false**, the option is never used.

-tp_oper_sel auto | true | false

Controls whether multiplier input operands are selected (ordered) for low power based on transition probability (TP). This optimization can reduce internal switching activity and therefore dynamic power in Booth multipliers when the inputs have significantly different transition probabilities.

When set to **auto**, the default, the option is only used if the tool determines there is a QoR benefit and if dynamic power is optimized. When set to **true**, the option is always used if dynamic power is optimized, and when set to **false**, the option is never used. The option only affects multipliers with Booth encoding.

-smart_compare auto | true | false

Controls whether comparator inputs are ordered for better timing. Depending on the logic depth of the comparator logic, an inverter is required at the output. Flipping the inputs eliminates the need for this inverter and shortens the critical path.

When set to **auto**, the default, or **true** the option is used if the tool determines there is a QoR benefit. When set to **false**, the option is not used.

-optimize_for default | area | speed | area,speed

Supplies a specific optimization goal for the specified cells. This command overrides the default optimization goals for the specified cell or cells.

design or cell list

An optional list of cells and designs. When present, the switch settings in this command will affect only the specified objects.

Option values set on specific objects will be listed in the implementation report by **report_resources**.

set_dp_smartgen_options

DESCRIPTION

The **set_dp_smartgen_options** command controls smart generation strategies used in datapath synthesis. Datapath generators select from these strategies to improve the final QoR.

The default value for all smart generation strategies is **auto**.

This command is only valid when datapath smart generation is enabled by setting the the **synlib_dwgen_smart_generation** variable to **true**.

EXAMPLES

The following example uses **set_dp_smartgen_options** to control the usage of booth encoded architectures when generating multipliers.

```
prompt> set_dp_smartgen_options -booth_encoding true
```

The following example sets all of the options to **auto**, except for **-booth_encoding**, which is set to **true** and overrides the value of **-all_options**:

```
set_dp_smartgen_options -all_options auto -booth_encoding true
```

SEE ALSO

```
report_dp_smartgen_options(2)  
synlib_dwgen_smart_generation(3)
```

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_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_extraction_options

Sets the parameters that influence extraction.

SYNTAX

```
status set_extraction_options  
[-max_process_scale max_process_scaling]  
[-min_process_scale min_process_scaling]  
[-default]
```

Data Types

<i>max_process_scaling</i>	float
<i>min_process_scaling</i>	float

ARGUMENTS

-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.

-default
Resets all options to their default values.

DESCRIPTION

The **set_extraction_options** command specifies the parameters that influence the extraction engine.

You can specify process scaling factors with the **-max_process_scale** and **-min_process_scale** options. The lengths and widths of the interconnect wires are scaled by the specified factors, without affecting interconnect wire depths or metal geometries inside standard cells and macros. You can use process scaling factors to adjust RC estimation for a process shrink, such as shrinking from 90 nm to 65 nm.

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 0.8 -
max_process_scale factor and a 0.7 -**min_process_scale** factor:

```
prompt> set_extraction_options -max_process_scale 0.8 -min_process_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)
```

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

set_false_path

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.

`set_false_path`

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

<i>value</i>	float
<i>port_list</i>	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:

```
set_fanout_load  
1510
```

```
prompt> set_fanout_load 2 all_outputs()
```

SEE ALSO

```
all_outputs(2)  
compile(2)  
remove_attribute(2)  
report_port(2)  
reset_design(2)  
set_max_fanout(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

`compile(2)`
`create_clock(2)`
`current_design(2)`
`remove_attribute(2)`
`report_clock(2)`
`reset_design(2)`
`set_prefer(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

compile(2)
current_design(2)
remove_attribute(2)
reset_design(2)

set_flatten

Sets or removes the **flatten** attribute on specified designs or on the current design, to enable or disable the flattening optimization step during **compile**.

SYNTAX

```
int set_flatten  
[true | false]  
[-effort low | medium | high]  
[-minimize single_output | multiple_output | none]  
[-phase true | false] [-design design_list] [-quiet]
```

Data Types

design_list list

ARGUMENTS

true | false
When *true*, sets the **flatten** attribute on the designs in *design_list* and enables flattening for those designs. When *false* (the default value), the **flatten** attribute is removed and flattening is turned off for those designs.

-effort low | medium | high
Indicates the level of CPU effort that **compile** uses to flatten a design. The default is *low*.

-minimize single_output | multiple_output | none
Indicates the minimization strategy to be used after a design is flattened. If *single_output* (the default value), Design Compiler independently minimizes the equations for each output in a design. If *multiple_output*, Design Compiler minimizes all logic for a design. If *none*, Design Compiler does not perform minimization.

-phase true | false
When *true*, allows logic flattening to invert the phase of outputs during **compile**. When *false* (the default value), logic flattening does not invert the phase of outputs. This option is used only if the **flatten** attribute is set.

-design *design_list*
Specifies a list of designs to be flattened. The default is the current design.

-quiet
Indicates that warning messages are not to be displayed.

DESCRIPTION

Sets or removes the **flatten** attribute on specified designs or on the current design, enabling or disabling the flattening optimization step during **compile**. If no options are specified, the **flatten** attribute is set on the current design.

To enable flattening during optimization across the entire design hierarchy, issue the **set_flatten** command on the top design and each subdesign. To display the values of all **flatten** attributes, use **report_compile_options**.

This optimization step reduces a logic network to a two-level sum-of-products (AND/OR) representation by removing all intermediate variables. Full flattening eliminates all existing logic structure. By removing sub-optimal intermediate variables, **compile** is able to use other intermediate variables that otherwise might not have been found.

If phase assignment has been enabled by invoking the **set_flatten** command with **-phase true**, then the flattening optimization step in **compile** will compare the implementation of each logic equation in both its original and complemented form and choose the form with the smallest estimated area.

When the minimization strategy is set to *single_output*, Design Compiler minimizes the equations for each output individually. This results in the smallest implementation for each output, but the design as a whole might not be the most efficient because product (AND terms) are not well shared between outputs. When the strategy is *multiple_output*, Design Compiler minimizes all the logic for a design, and shares as many product terms between outputs as possible.

NOTE: Setting the **flatten** attribute can potentially cause **compile** to have long run times or to run out of memory, because logic flattening and equation minimization is a potentially exponential process. If you encounter an "out of memory" error or excessive run time, you can modify the options one at a time and re-run after each modification. Here is the recommended order for modifying the options:

1. Use *single_output* instead of *multiple_output* minimization strategy.
2. Use the **-phase false** option instead of the default **-phase true** option.
3. Use a lower effort setting.

To remove the **flatten** attribute, use **remove_attribute** or execute **set_flatten false**. **reset_design** removes all attributes, including **flatten**.

For more details on flattening during optimization, refer to the *Design Compiler Reference Manual: Optimization and Timing Analysis*.

EXAMPLES

In this example, **flatten** is enabled on the design "TEST".

```
prompt> set_flatten -design TEST
```

In this example, **flatten** is enabled on the current design with the **effort** and **minimize** options specified, and **flatten** is disabled on the design "OLD."

```
prompt> read TEST
prompt> set_flatten -effort high -minimize multiple_output
prompt> set_flatten -design OLD false
```

SEE ALSO

`compile(2)`
`current_design(2)`
`remove_attribute(2)`
`reset_design(2)`

set_fsm_encoding

Specifies the bit encodings for states in the current design.

SYNTAX

```
int set_fsm_encoding  
encoding_list
```

Data Types

encoding_list list

ARGUMENTS

encoding_list

A list of all states in the current design with the assigned bit encoding. Each state name and the encoding must be enclosed in double quotes (" "). Encodings are specified using one of two formats. The first format consists of a base specifier followed by the # character, followed by a string of digits in the given base numbering system. The string of digits may be separated by an optional underscore character to make the encodings easier to read. If the base specifier and # character are omitted, the string of digits are interpreted as a decimal (base 10) value. For example, 15 may be entered as:

2#1111

8#17

10#15

15

16#f

The second format consists of the "^" character, followed by a character base specification, followed by a string of digits in the given base. For example, 15 may be entered in this format as:

^B1111(Binary)

^O17(Octal)

^D15(Decimal)

^Hf(Hexadecimal)

If more than one state is to be contained in the list, the list of states must be enclosed in braces ({}).

DESCRIPTION

The **set_fsm_encoding** command specifies the set of all legal states and encodings in the current design. Any state omitted from the encoding list has unspecified encodings set (unless **extract -reachable** is run after this command). This information is used for extraction of a state machine from a netlist.

Only one *encoding_list* may be active at any given time on the current design.

An empty *encoding_list* removes the encodings for all states in the design.

EXAMPLES

The following example assigns codes to two states in the machine. Encodings for the states not in the list have their encodings removed.

```
prompt> set_fsm_encoding {"IDLE=2#000" "FIVE=2#001"}
```

To remove all encodings from the states, enter an empty *encoding_list* .

```
prompt> set_fsm_encoding {}
```

SEE ALSO

```
compile(2)  
set_fsm_encoding_style(2)  
current_design(3)
```

set_fsm_encoding_style

Defines the encoding style for assigning unencoded states.

SYNTAX

```
int set_fsm_encoding_style  
{one_hot | zero_one_hot | binary | gray | auto | neutral}
```

ARGUMENTS

```
{one_hot | zero_one_hot | binary | gray | auto | neutral}
```

Specifies the encoding style that is to be used during state assignment. The encoding styles are described in the DESCRIPTION section. The default is **auto**.

DESCRIPTION

Defines the encoding style to be used for determining unencoded states during the generation of a state machine design. (The process of determining encodings for states is referred to as state assignment.) If you have manually specified encodings for any of the states in the design, state assignment does not reassign any of these encodings. The code length is based on 1) the length of any manually-assigned encodings, or 2) the number of instance names specified in the **set_fsm_state_vector** command. If the code length cannot be determined by any of these criteria, it is determined by the encoding style selected.

The encoding styles are described below.

The **one_hot** encoding style generates codes with a bit length equal to the number of states in the state machine, each state being represented by one bit position. The code is a string of zeros, interrupted by a one in the state's particular bit position. A warning is generated if the number of elements in the state vector is not equal to the number of states, in which case the length of the code is reset to the number of states in the machine. If the **one_hot** encoding style is specified, all manually-assigned codes must also be **one_hot** codes.

The **binary** and **gray** encoding styles sequentially assign codes to unassigned states using a binary numbering or gray numbering sequence, respectively. The binary numbering sequence consists of using binary values to encode the sequence of counting integers. For example, four states encoded using two bits would be numbered 0, 1, 2, and 3, represented by the binary encodings of 00, 01, 10, and 11. The **gray** numbering sequence assigns codes to states such that successive codes never differ by more than one bit. In this case, the same four states encoded using the gray encoding style would be numbered 0, 1, 3, and 2, represented by the encodings of 00, 01, 11, and 10. Note that in this encoding sequence only one bit position changes value in going from one to the next (for example, 01 to 11.) The sequence in which the codes are assigned to the states is based on the ordering of the states. States are ordered via their entry in the state table, or may be manually assigned using the **set_fsm_order** command. If any states have manually assigned encodings, then these states and their respective codes are omitted from the ordering of binary or gray codes. Additionally, if any state is unordered, an arbitrary ordering is used

that does not affect any existing order. The default code length is base log2 of the number of states in the machine.

The **auto** encoding style generates codes chosen in a manner to best minimize the logic of the state machine. If the length of the bit encodings cannot be determined using the criteria specified below, the Design Compiler's **auto** encoding style selects the code length that best minimizes the state machine.

The **neutral** encoding style will keep state assign as user defined in the HDL code, i.e., keep the encoding style unchanged.

EXAMPLES

The following example shows four states, "IDLE", "FIVE", "TEN", and "OWE_DIME", with a **one_hot** encoding style:

```
prompt> set_fsm_encoding_style one_hot
```

During **compile**, the state assignment encodes the states as 1000, 0100, 0010, and 0001, respectively.

The following example shows the same machine with the states ordered as "IDLE", "FIVE", "TEN", and "OWE_DIME", with a **binary** encoding style:

```
prompt> set_fsm_encoding_style binary
```

During **compile**, the state assignment encodes the states as 00, 01, 10, and 11, respectively.

SEE ALSO

```
compile(2)  
current_design(2)  
set_fsm_order(2)  
set_fsm_state_vector(2)
```

set_fsm_minimize

Determines whether or not state minimization is to be performed on the state machine design during **compile**.

SYNTAX

```
int set_fsm_minimize  
true | false
```

ARGUMENTS

```
true | false  
    Indicates whether or not state minimization is to be performed. The default  
    is false.
```

DESCRIPTION

Determines whether or not state minimization is to be performed on the state machine design during **compile**. State minimization is not performed again on a design that has already had state minimization performed (for example, with the **fsm_minimize** command). You can additionally specify states that are to be preserved by using the **set_fsm_preserve_state** command.

EXAMPLES

The following examples show a state table design being read in, set up for state minimization, and **compiled**.

```
prompt> read -f st example.st  
prompt> set_fsm_minimize true  
prompt> compile
```

SEE ALSO

```
compile(2)  
set_fsm_preserve_state(2)
```

set_fsm_order

Sets the ordering of states in a state machine design.

SYNTAX

```
int set_fsm_order  
state_list
```

Data Types

```
state_list      list
```

ARGUMENTS

```
state_list
```

List of states in the state machine to which an order is assigned. All states in the *state_list* must be states in the state machine. If more than one state is specified, they must be enclosed in braces ({}).

DESCRIPTION

Defines an ascending ordering of states in the current design. Any previously-assigned order is removed and replaced by the new order specified in *state_list*. Any states that are not included in *state_list* are given an arbitrary order that does not violate the specified ordering.

compile uses state ordering when determining encodings for unencoded states using the **binary** or **gray** encoding style. (See **set_fsm_encoding_style** for a description of those encoding styles.) The first state is assigned the lowest unused state code, and each successive state is assigned the next unused code in in the **binary** or **gray** numbering sequence.

An empty *state_list* removes the ordering of all states.

EXAMPLES

The following example defines the ordering of states such that "IDLE" is the first state, "FIVE" is the second state, and so on:

```
prompt> set_fsm_order {IDLE FIVE TEN OWE_DIME}
```

To remove the ordering from all states, enter an empty *state_list*:

```
prompt> set_fsm_order {}
```

SEE ALSO

```
compile(2)  
set_fsm_encoding_style(2)
```

```
set_fsm_order  
1524
```


set_fsm_preserve_state

Specifies states to be preserved during state minimization.

SYNTAX

```
int set_fsm_preserve_state  
state_list
```

Data Types

state_list list

ARGUMENTS

state_list
List of state names to be preserved. If more than one state is specified, they must be enclosed in braces ({}).

DESCRIPTION

Specifies those states which are to be preserved during state minimization.

This command overrides any previously specified state preservation. To undo this command, enter **set_fsm_preserve_state** with an empty *state_list*.

EXAMPLES

The following command specifies that state "IDLE" is to be preserved during state minimization:

```
prompt> set_fsm_preserve_state {IDLE}
```

SEE ALSO

set_fsm_state_vector

Specifies the instance names for flip-flops used to implement the state vector.

SYNTAX

```
int set_fsm_state_vector  
vector_list
```

Data Types

vector_list list

ARGUMENTS

vector_list
Specifies a list of instance names for the state vector in the state machine. If more than one name is specified, they must be enclosed in braces ({}). The first instance name in the list is considered to be the most significant bit of the vector, while the last instance name is considered the least significant bit of the vector.

DESCRIPTION

Explicitly names the instances of the flip-flops used to store the current state of the state machine. Additionally, the specification of a state vector defines the length of the codes used for the encoded states in the machine.

EXAMPLES

The following example specifies the state vector's instance names:

```
prompt> set_fsm_state_vector {ff0 ff1 ff2 ff3}
```

SEE ALSO

```
compile(2)  
set_fsm_encoding(2)
```

set_fuzzy_query_options

Defines query options for fuzzy matching.

SYNTAX

Boolean **set_fuzzy_query_options**
[-hierarchical_separators *separator_list*]
[-bus_name_notations *bus_name_list*]
[-class *class_list*]
[-regsub *regsub*]
[-regsub_cumulative]
[-reset]
[-show]
[-verbose]

Data Types

<i>separator_list</i>	list
<i>bus_name_list</i>	list
<i>class_list</i>	list
<i>regsub</i>	list

ARGUMENTS

-hierarchical_separators *separator_list*

Specifies a list of equivalent hierarchy separators in object names. There must be at least 2 elements in the list, and the list element must be a single character. The default list is {/ _ .}. The following are not supported as hierarchical separators: 0-9, a-z, A-Z, '*', '?', '\', '+', '^', '[', ']', '(', ')', '<', '>', and '{', '}'.

-bus_name_notations *bus_name_list*

Specifies a list of equivalent bus name notations in object names. There must be at least 2 elements in the list, and the list element must be 2 characters. The first character of the element is the opening bus name character, and the second character is the closing bus name character. The default list is {[_ ()}]. The following are not supported as opening or closing bus name characters: 0-9, a-z, A-Z, '*', '?', '\', '+', '^', and '/'. Bracketed bus notations must be paired. For example, "[" are not properly paired brackets, so they are not supported. For non-bracket bus name notations, the opening and closing bus name character must be the same. For example, "-_" is not a supported bus name notation.

-class *class_list*

Specifies a list of object classes for which the query options will be applied. Only cell, port, pin and net object classes are supported. The default list is {cell pin port net}.

-regsub *regsub*

Specifies a list of options for the **regsub** Tcl command. Each list should include 3 string elements {{switches}, {match_regexp}, {substitution_exp}}. The options can be specified multiple times in one **set_fuzzy_query_options**

command. When multiple **-regsub** options are used without **-regsub_cumulative**, the **-regsub** options are applied to the leaf object name and pattern in the order you specify. If **-regsub_cumulative** is used, multiple **-regsub** options are still applied to the object name and pattern in the order you specify. However, it has the cumulative effect in the sense that the subsequent **-regsub** option is applied to the outputs of the object name and pattern string from the previous **-regsub** option.

This option is only applicable to a pin, port, or net. If the **-regsub** is used without other fuzzy matching options, the default values for other options are still used. In order to save run time, use this option only if the existing fuzzy matching scheme fails to match an object.

-regsub_cumulative

Specifies that multiple **-regsub** options are to be applied to the outputs of the object name and pattern string from the previous **-regsub** option.

-reset

Resets query options to the program default.

-show

Shows the current definition of the query options. If used with other options, the new query options will be set first and then displayed.

-verbose

Prints detailed information regarding the matched object and the fuzzy rules.

DESCRIPTION

The **set_fuzzy_query_options** allows you to define query options for some applications such as **extract_physical_constraints** to guide the fuzzy matching. The purpose of the fuzzy matching is to resolve the naming differences that are often due to ungrouping and **change_names** command activity. The hierarchy separators and bus names are typical sources of the naming differences.

The **set_fuzzy_query_options** command defines two types of query options: hierarchy separators and bus name notations.

When a list of characters is defined as equivalent in terms of hierarchy separator, the object query uses the equivalence when matching the objects in the in-memory netlist. For example, if the list `{/ _ .}` is defined as equivalent hierarchy separators, the cell named "a.b_c/d_e" is matched with name string "a/b_c.d/e" provided in the Tcl scripts or in the DEF files. The default hierarchical separator list for fuzzy matching has a value of `{/ _ .}`.

When a list of bus notations is defined as equivalent in terms of bus names, the object query uses the equivalence when matching the objects in the in-memory netlist. For example, if the list `{[] __ }` is defined as equivalent bus notations, the cell named "a[4][5]" is matched with the name string "a_4__5_" provided in the Tcl scripts or DEF files. The default bus name notation list for fuzzy matching has value of `{[] __ ()}`.

When a list or multiple lists of options for the **regsub** Tcl command are specified, the **regsub** command is applied on both the object name and the pattern string for fuzzy name matching if the existing fuzzy matching scheme fails to match an object.

set_fuzzy_query_options

The **set_fuzzy_query_options** command can be issued multiple times. Unless the **-show** option is used alone, the new options always overwrite the previous option settings.

Use the **-reset** option to reset the fuzzy query options to the program default. Use **-show** to report the current fuzzy matching options. When **-show** is used with other options, the new options are set first and then reported.

Be cautious when defining fuzzy matching options. To save runtime, do not add unnecessary elements in the hierarchical separator list or the bus name notation list.

Fuzzy matching is enabled by variable **fuzzy_matching_enabled**. The **set_fuzzy_query_options** command is effective only when the **fuzzy_matching_enabled** variable is set to true. Runtime for fuzzy matching could be much slower. Fuzzy matching should be disabled once it is no longer needed. By default, fuzzy matching is off.

Fuzzy matching does not support wildcard.

Set variable **find_ignore_case** to true to enable case-insensitive fuzzy matching. Avoid using **-nocase** option in commands such as **get_cells** or **get_pins** option to enable case-insensitive fuzzy matching. This is because during fuzzy matching, switching case sensitivity mode using command line option **'-nocase'** will force fuzzy matching to reconfigure each time the mode is changed, and incur high runtime. It is recommended to either set variable **find_ignore_case** to "true" so that fuzzy matching always works in case-insensitive mode, or set variable **find_ignore_case** to "false" and at the same time do not use **'-nocase'** for any command so that fuzzy matching always works in case sensitive mode.

EXAMPLES

The following examples use **set_fuzzy_query_options** to set the fuzzy matching options:

```
# Show program default.
prompt> set_fuzzy_query_options -show
*****
```

```
Query Options
*****
```

Hierarchy-Separator	Bus-Notation	Object-Class
{/ _ .}	{[] __ ()}	{cell pin port net}

1

```
# Non-default
prompt> set_fuzzy_query_options -hierarchical_separators {/ _} \
    -bus_name_notations {[] ||} -class {cell port} -show
*****
```

```
Query Options
*****
```

Hierarchy-Separator	Bus-Notation	Object-Class
{/ _}	{[] }	{cell port}

1

Default hierarchy separators with customized bus names and class list.

```
prompt> set_fuzzy_query_options -bus_name_notations {[] $$}\
      -class {cell port} -show
```

Query Options

Hierarchy-Separator	Bus-Notation	Object-Class
{/ _ .}	{[] \$\$}	{cell port}

1

Reset it

```
prompt> set_fuzzy_query_options -reset -show
```

Query Options

Hierarchy-Separator	Bus-Notation	Object-Class
{/ _ .}	{[] __ ()}	{cell pin port net}

1

The following example matches the object named "a_BaR" and user pattern "a":

```
prompt> set_fuzzy_query_options -regsub {{-all -nocase} {_BAR} {}}
```

The following example matches the object named "a_3" and the user pattern "a[3]":

```
prompt> set_fuzzy_query_options -regsub {{}} {(.*)([d+])} {1_2}}
```

The following example matches the object named "a[3]_BAR" and the user pattern "a[RED]":

```
prompt> set_fuzzy_query_options -regsub {{-nocase} {_BAR$} {}} \
      -regsub {{}} {RED} {3}} -regsub_cumulative
```

The following example matches the object named "a[3]" and the user pattern "a_3":

```
prompt> set_fuzzy_query_options -regsub {{-nocase} {_BAR$} {}} \
      regsub {{}} {(.*)([d+])} {1_2}} -regsub {{}} {RED} {3}}
```

The following example matches the object named "a_3_BAR" and user pattern "a[RED]":

set_fuzzy_query_options

```
prompt> set_fuzzy_query_options -regsub {} {_BAR$} {} \  
       -regsub {} {RED} {3}} -regsub {} {(.*)(d+)} \  
       {1_2}} -regsub_cumulative
```

SEE ALSO

```
fuzzy_matching_enabled(3)  
find_ignore_case(3)
```

set_host_options

Controls the maximum number of CPU cores that can be used for parallel execution.

SYNTAX

```
status set_host_options  
[-max_cores number_of_cores]
```

Data Types

number_of_cores integer

ARGUMENTS

```
-max_cores number_of_cores  
           Specifies the maximum number of CPU cores that are allowed for parallel  
           execution.
```

DESCRIPTION

The **set_host_options** command is used to specify the maximum number of CPU cores that the tool can use during parallel execution. The default is 1, which indicates that parallel execution is not enabled. The maximum number of cores that can be specified is 16.

To disable parallel execution, set the **-max_cores** option to 1 or run the **remove_host_options** command.

EXAMPLES

The following example tells the tool that up to four cores can be used for parallel execution:

```
prompt> set_host_options -max_cores 4
```

SEE ALSO

```
compile_ultra(2)  
remove_host_options(2)  
report_host_options(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*.

```
set_ideal_network  
1538
```

```
prompt> current_design CLOCK_GEN  
prompt> set_ideal_network {port1 port2}
```

SEE ALSO

```
compile(2)  
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_ignored_layers

Sets ignored routing layers for RC estimation and congestion analysis. This command is supported only in topographical mode.

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 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 tool marks all routing layers above the specified layer as ignored layers for RC and congestion estimation.

DESCRIPTION

In Design Compiler topographical mode, this command sets the routing layers that are ignored during congestion analysis and RC estimation.

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}
```

```
set_ignored_layers  
1542
```

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_impl_priority

Sets the formula attribute of the priority parameter and/or the set_id attribute for implementations in synthetic libraries.

SYNTAX

```
status set_impl_priority  
[-priority formula]  
[-set_id id]  
implementation_list
```

Data Types

<i>formula</i>	string
<i>id</i>	int
<i>implementation_list</i>	list

ARGUMENTS

-priority *formula*
Specifies the string to which the **formula** attribute of the **priority** parameter should be set. An empty string causes the **priority** parameter to be removed from the specified implementations. The formula should evaluate to an integer between 0 and 10.

-set_id *id*
Specifies the value to which the **set_id** attribute should be set.

implementation_list
Specifies a list of implementations for which the **priority** parameter and/or the **set_id** attribute are to be set. Implementation names must contain a library prefix. For more information, refer to the EXAMPLES section of this man page. If more than one object is included, they must be enclosed in quotes or braces ({}).

DESCRIPTION

The **set_impl_priority** command sets the **formula** attribute of the **priority** parameter and/or the **set_id** attribute on the specified implementations. Either **-priority** or **-set_id** must be specified. The synthetic libraries containing the implementations must be loaded into **dc_shell**.

NOTE: This command cannot be used in **dc_shell** scripts embedded in HDL descriptions. In addition, the 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**, all **priority** parameters or **set_id** attributes will be written out, causing library objects to be permanently changed.

EXAMPLES

The following command sets the formula attribute in the **priority** parameter to "n < 4 ? 6 : 2" and the **set_id** attribute to 4 for 'my_lib.sldb/m1/rpl':

```
prompt> set_impl_priority -priority "n < 4 ? 6 : 2" -set_id 4 my_lib.sldb/m1/rpl
```

The following command sets the **set_id** attribute to 3 for all the implementations of module 'm2' in the synthetic library 'my_lib.sldb':

```
prompt> set_impl_priority -set_id 3 my_lib.sldb/m2/*
```

The following command removes the **priority** parameter for implementations 'imp1' and 'imp2' of the module 'syn_lib.sldb/mod':

```
prompt> set_impl_priority -priority "" {syn_lib.sldb/mod/imp1, syn_lib/mod/imp2}
```

SEE ALSO

compile(2)
report_synlib(2)

set_implementation

Specifies the implementation to use for synthetic library cell instances in a design.

SYNTAX

```
status set_implementation  
implementation_name  
cell_list  
[-check_impl]
```

Data Types

<i>implementation_name</i>	string
<i>cell_list</i>	list

ARGUMENTS

implementation_name
Specifies the name of the implementation used to implement the function of the *cell_list* synthetic library cell instances. A single period (.) in the implementation name locks in the current implementation of the instance. The implementation name must also contain the module name, such as *DW01_add/cia*, when setting implementation on synthetic operators.

cell_list
Specifies the synthetic library cell instances to be implemented by *implementation_name*. If more than one instance name is specified, enclose the names in quotes (") or in braces ({}). Specify either instantiated cell instances or inferred cell instances, but not both, in *cell_list*.

-check_impl
Checks for the existence of the specified implementation before setting implementation on the cell instances.

DESCRIPTION

The **set_implementation** command tags specified synthetic library cell instances with the name of an implementation to use to implement them. When the **compile** command is run, the tool chooses the user-specified implementation to implement a cell instance. Otherwise, **compile** chooses the implementation it considers most appropriate for a cell instance. For a design that has already been compiled, **set_implementation** with a period (.) locks in the current implementation. Subsequent compiles do not change the implementation.

The **set_implementation** command does not function on cell instances that are not defined in a synthetic library.

The **report_resources** command lists the current implementation of all synthetic library instances, and indicates whether the implementation has been locked by **set_implementation**.

Use the **report_synlib** command for the possible implementations of a synthetic library cell.

EXAMPLES

If *A1* is an instance of the *DW01_ADD* cell in the current design, a *cla* carry-lookahead implementation can be specified to implement *A1*. The following example assumes that *DW01_ADD* is a module defined in a synthetic library listed in **synthetic_library**, and that *DW01_ADD* has a *cla* carry-lookahead implementation associated with it:

```
prompt> set_implementation cla A1
```

The following example directs **compile** to use *rpl* for both *Adder1* in the design *FIFO1* and *Adder2* in the design *FIFO2*:

```
prompt> set_implementation rpl {FIFO1/Adder1 FIFO2/Adder2}
```

The following example removes any effects of **set_implementation** from all synthetic cells of the current design:

```
prompt> remove_attribute [get_cells *] implementation
```

The following example locks the implementation of all synthetic cells in the current design. On subsequent compiles, the implementation does not change.

```
prompt> set_implementation . ""
```

The following example sets a *cla* implementation on an ADD operator cell named *A1*:

```
prompt> set_implementation DW01_add/cla A1
```

SEE ALSO

```
compile(2)
remove_attribute(2)
report_resources(2)
report_synlib(2)
synthetic_library(3)
```

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_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

`compile(2)`
`report_compile_options(2)`
`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_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_leakage_power_model

Specifies the model that will be optimized by leakage optimizations.

SYNTAX

```
status set_leakage_power_model
[-type model_name]
[-mvth_weights weights]
[-reset]
```

Data Types

<i>model_name</i>	string
<i>weights</i>	string

ARGUMENTS

-type *model_name*
Specifies the name of the leakage power model. Valid model names are **leakage** and **channel_width**. The default model is **leakage**.

-mvth_weights *weights*
Specifies the design-level weights for the voltage groups. The library level values specified in the technology library are ignored and the values specified with the **-mvth_weights** option are used to calculate the weighted sum of channel widths.

-reset
Resets the model to **leakage** and clears the weights that were previously set.

DESCRIPTION

Specifies the model that will be optimized by leakage. The leakage optimizations during compile optimize a model. One of the following models may be chosen as the leakage power model:

leakage
channel_width

When the model is **leakage**, the leakage power is computed and optimized. When the model is **channel_width**, a weighted sum of channel width is computed and optimized.

The default model is **leakage**.

For this command to work, the library must contain:

- threshold-voltage groups

- the channel width for cells

Furthermore, if the `threshold_voltage_channel_width_factor` is not present in the library, you must assign `-mvth_weights` at the design level.

EXAMPLES

In this example, the weights specified at the library level are used to calculate the weighted sum:

```
prompt> set_max_leakage_power 0
prompt> set_leakage_power_model -type channel_width
prompt> compile_ultra
```

The weights specified in the library are used to calculate the weighted sum. The following example illustrates the specification of design-level weights. This specification overrides the library-level weights specified in the technology library.

```
prompt> set_max_leakage_power
prompt> set_leakage_power_model -type channel_width \
    -mvth_weights "lvt = 1000 nvt = 300 hvt = 1"
prompt> compile_ultra
```

SEE ALSO

```
report_power(2)
set_max_leakage_power(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_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:

```
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 <i>attribute_name</i>

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_libcell_dimensions

Sets the width and height of a library cell.

SYNTAX

```
int set_libcell_dimensions  
-cell cell_name  
-width width  
-height height
```

Data Types

<i>cell_name</i>	string
<i>width</i>	float
<i>height</i>	float

ARGUMENTS

-cell *cell_name*
Specifies the name of the cell in the target library.

-width *width*
Specifies the width (dimension in x direction) of the library cell.

-height *height*
Specifies the height (dimension in y direction) of the library cell.

DESCRIPTION

The **set_libcell_dimensions** command sets the width and height of the specified library cell. All the arguments to this command are required. The units of width and height are in microns.

During **reoptimize_design**, LBO will use the pin location in conjunction with the cell location, dimension, and orientation to determine the absolute location of the pin. This will improve the accuracy of delay prediction during LBO, especially when large cells such as RAMs and large macros are used.

EXAMPLES

In the following example, the width of library cell INVERT_2 is set to 10 microns and the height is set to 15 microns.

```
prompt> set_libcell_dimensions -cell INVERT_2 -width 10 -height 15
```

SEE ALSO

set_libpin_location(2)

set_libpin_location

Sets the location of a pin of a library cell relative to the origin of the library cell.

SYNTAX

```
int set_libpin_location  
-cell library_cell_name  
-pin pin_name_of_the_library_cell  
-coordinate {x_coordinate y_coordinate}
```

Data Types

<i>library_cell_name</i>	string
<i>pin_name_of_the_library_cell</i>	string

ARGUMENTS

```
-cell library_cell_name  
    Specifies the name of the library cell.  
  
-pin pin_name_of_the_library_cell  
    Specifies the name of the pin of the library cell.  
  
-coordinate {x_coordinate y_coordinate}  
    Specifies the x and y coordinates of the pin. These coordinates are relative  
    to the origin of the cell specified. The units of the coordinates are in  
    microns.
```

DESCRIPTION

The **set_libpin_location** command sets the location of the specified pin of the specified library cell. All the arguments to this command are required.

This command should be issued before **reoptimize_design**. During **reoptimize_design**, LBO will use the pin location inconjunction with the cell location and orientation to determine the absolute location of the pin. This will improve the accuracy of delay prediction during LBO, especially when large cells such as RAMs and large macros are used.

Specify **set_libpin_location** only once for all the pins on the library cell, irrespective of the number of times the cell is instantiated or where it will be instantiated.

EXAMPLES

In the following example, location of the pin Z of library cell INVERT_2 is set to {10.1 5.2}.

```
prompt> read_clusters design.pdef prompt> set_libpin_location -cell INVERT_2
```



```
-pin A -coordinate {10.1 5.2}
```

SEE ALSO

```
set_port_location(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)
set_wire_load_min_block_size(2)

```
set_wire_load_mode(2)
set_wire_load_model(2)
set_wire_load_selection_group(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

`compile(2)`
`current_design(2)`
`insert_dft(2)`
`link(2)`
`remove_attribute(2)`
`reset_design(2)`
`translate(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)
simplify_constants(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)  
simplify_constants(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)  
simplify_constants(2)
```

set_logicbist_configuration

Controls the parameters for insertion (X)DBIST logic.

SYNTAX

```
status set_logicbist_configuration
[-bist_ready true_or_false]
[-integration true_or_false]
[-codec_insertion true_or_false]
[-auto true_or_false]
[-type dbist_or_xdbist_or_sbist]
[-max_chain_length length]
[-codec_count n]
[-balance_bist_segments true_or_false]
[-invert_prpg_clock true_or_false]
[-prpg_shadow_si count]
[-prpg_length 257_or_479]
[-test_mode test_mode_names]
[-max_pattern_count max_pattern_count]
```

Data Types

<i>length</i>	integer
<i>n</i>	integer
<i>count</i>	integer

ARGUMENTS

-bist_ready *true_or_false*
Allows the design to be analyzed for testability and for inserting test points to make the design BIST Ready. This switch is turned on by default unless you specify *-integration*.

-integration *true_or_false*
Performs codec insertion and integration of the codec inserted design with the Bist Controller when set to true.

-codec_insertion *true_or_false*
Performs codec insertion alone on the BIST cores, if you need High Capacity Bist Insertion Flow. This switch can be combined with *-integration* to perform BIST integration.

-type *dbist_or_xdbist_or_sbist*
Specifies the type of BIST codec and controller. Allowed values are *dbist*, *sbist* and *xdbist*.

-max_chain_length *length*
Specifies the counter length to be used in the BIST Controller. If the specified count is less than the length of the longest Bist Scan Chains used in the design, the count specified is ignored.

`-codec_count n`
Specifies the number of bist codecs to be inserted. If none specified, only one codec is inserted for bist insertion.

`-balance_bist_segments true_or_false`
Balances the bist channels in BIST mode. By default, the bist channels are balanced. To turn it off, set the switch value to false.

`-invert_prpg_clock true_or_false`
Inverts the bist clock that feeds the PRPG Registers. By default, the bist clock is not inverted.

`-prpg_shadow_si count`
Specifies the number of PRPG shadow chains. Allowed values are between 1 and 12.

`-prpg_length 257_or_479`
This switch is used to specify the length of the PRPG to be used. You can either use a 257 bit PRPG or a 479 bit PRPG using this switch. If none is specified a 479 bit PRPG will be used.

DESCRIPTION

The **set_logicbist_configuration** command specifies the BIST configuration for a design. Using different switches, you can fully customize the way the BIST codec and Controller are instantiated into the design.

EXAMPLES

The following example specifies that logic BIST codec should have 8 PRPG shadow inputs:

```
prompt> set_logicbist_configuration -prpg_shadow_si 8
```

The following example specifies that BIST be integrated into the design:

```
prompt> set_logicbist_configuration -integration TRUE
```

SEE ALSO

```
report_dft_configuration(2)  
report_logicbist_configuration(2)  
set_autofix_configuration(2)  
set_dft_configuration(2)
```

set_map_only

Sets the **map_only** attribute on specified objects so that they can be excluded from logic-level optimization during **compile**.

SYNTAX

```
status set_map_only  
object_list  
[flag]
```

Data Types

<i>object_list</i>	list
<i>flag</i>	string

ARGUMENTS

object_list
Specifies a list of cells, references, or designs on which to set the **map_only** attribute.

flag
Determines the value to which the **map_only** attribute is to be set. Valid values are true (the default) or false.

DESCRIPTION

The **set_map_only** command places the **map_only** attribute on the specified objects, and sets the attribute value to either true or false. The default value of the **map_only** attribute is false. Setting the value to true excludes specified objects from logic-level optimization (flattening and structuring) during **compile**. Setting the **map_only** attribute to false resets the attribute to its default value, allowing the objects to be included in logic-level optimization.

To remove the **map_only** attribute, use the **remove_attribute** command. Alternatively, you can set **map_only** to false, which has the same effect as removing it. The **reset_design** command removes all attributes from a design, including **map_only**.

When the **map_only** attribute is set to true on a cell, reference, or design, the optimization of that part is treated specially. In general, **compile** attempts to map the cell exactly in the target library. For example, if you flag a GTECH_ADD_ABC cell as **map_only**, **compile** will try to find a full adder in your target library to use in the implementation. This allows you to suggest to **compile** (in a technology-independent manner) to use a gate that it otherwise might not have used.

While **map_only** suggests a gate for **compile** to use, it does not guarantee the gate's use. If **compile** cannot find a gate in your target library that has the correct function, **map_only** will be ignored. If the **map_only** gate is fed by a constant, **compile** will ignore the **map_only** attribute in order to simplify the gate.

The **map_only** attribute may be overridden in order to meet timing constraints. When

compile -map_effort is set to high, an extra synthesis is done on the design. This synthesis selectively replaces **map_only** gates in order to meet design constraints. Gates that are **map_only** are kept if they are not on the critical path or have proved to be the best (fastest) choice for optimization. The **map_only** gates are remapped only if they are preventing constraints from being met.

EXAMPLES

In the following example, the **map_only** attribute is set to true on the U1 cell:

```
prompt> set_map_only U1
```

If the **map_only** attribute is set on a reference object, cells using that reference have the **map_only** attribute set during **compile**. In the following example, the **map_only** attribute is set to false on the U1 cell:

```
prompt> set_map_only U1 false
```

In the following example, all instances of MUX21 in the current design are considered **map_only**:

```
prompt> set_map_only [get_references MUX21]
```

If the **map_only** attribute is set on a library design object, cell instances of that design have the **map_only** attribute during **compile**. In this example, all the instances of AND2 from **my_library** are considered **map_only**:

```
prompt> set_map_only my_library/AND2
```

SEE ALSO

```
compile(2)
current_design(2)
get_references(2)
remove_attribute(2)
reset_design(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

```
compile(2)  
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

set_max_delay

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

```
compile(2)  
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
```

```
set_max_dynamic_power  
1600
```

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.

```
set_max_fanout  
1602
```

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)  
compile_ultra(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.

```
set_max_time_borrow  
1610
```

```
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_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
```

```
set_message_info
1616
```

```
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_message_severity

Sets the severity level of the DFT message.

SYNTAX

```
int set_message_severity  
-names message_tags  
severity
```

Data Types

<i>message_tags</i>	list
<i>severity</i>	string

ARGUMENTS

-names message_tags
Specifies the list of messages **message_tags** whose severity needs to be changed (down/up graded).

severity
Specifies the severity for the messages *message_tags*. Valid values are error, warning and ignore.

DESCRIPTION

The **set_message_severity** command identifies the set of messages whose severity needs to be changed. Both the options **-names** and *severity* are required.

set_message_severity can be used to reset the severity of a message back to its original type.

EXAMPLES

The following example specifies the severity of messages "TEST-813" and "TEST-893" to be warning.

```
prompt> set_message_severity -names {TEST-813 TEST-893} w
```

The following example resets the severity of message "TEST-893" to error.

```
prompt> set_message_severity -names {TEST-893} error
```

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)
compile(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

`compile(2)`
`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_minimize_tree_delay

Sets the **minimize_tree_delay** attribute on a design or designs.

SYNTAX

```
integer set_minimize_tree_delay  
[true | false]  
[-design design_list]  
[design_list]
```

Data Types

design_list list

ARGUMENTS

true | false
 Determines the value to which the **minimize_tree_delay** attribute is to be set.
 The default is **true**.

-design *design_list*
 Specifies a list of designs on which to set the **minimize_tree_delay**
 attribute. The default is the current design.

DESCRIPTION

This command sets the **minimize_tree_delay** attribute on a design or designs, thus determining whether an arithmetic expression tree (for example, an adder chain) is to be restructured to minimize delay during **compile**. If **minimize_tree_delay** is not set, then the value of the **hlo_minimize_tree_delay** environment variable is used. The default value of this variable is **true**. Thus, by default, all expression trees are candidates for tree height minimization if timing constraints are specified. Setting the **minimize_tree_delay** attribute overrides the value of **hlo_minimize_tree_delay** for the design or designs on which the attribute is set.

The restructuring of arithmetic expression trees is based on the arrival times of inputs to the tree. In the case of equal arrival times, a balanced tree is constructed.

Note that even if **minimize_tree_delay** is set to **true**, tree height minimization is not performed if the design has no timing constraints, or if the **hlo_resource_allocation** variable is set to *area_only*, *area_no_tree_balancing*, or *none*.

For more details on tree height minimization, see the *VHDL Compiler Reference Manual* or the *HDL Compiler for Verilog Reference Manual*.

To remove the **minimize_tree_delay** attribute, use the **remove_attribute** command. The **reset_design** command removes all attributes from a design, including **minimize_tree_delay**.

EXAMPLES

In the following example, **minimize_tree_delay** is enabled on the design named TEST:

```
prompt> set_minimize_tree_delay -design TEST
```

In the following example, **minimize_tree_delay** is enabled on the current design and disabled on the design named OLD:

```
prompt> read TEST
```

```
prompt> set_minimize_tree_delay
```

```
prompt> set_minimize_tree_delay false -design OLD
```

SEE ALSO

```
compile(2)  
current_design(2)  
remove_attribute(2)  
reset_design(2)  
hlo_resource_allocation(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_model_drive

Sets the **model_drive** attribute to a specified value on specified input or inout ports to set their drive values during synthetic library modeling.

SYNTAX

```
int set_model_drive  
  drive_value  
  port_list
```

Data Types

```
drive_value      float  
port_list        list
```

ARGUMENTS

drive_value
Specifies the value to which the **model_drive** attribute is to be set, thus specifying the estimated drive value for the *port_list* in terms of standard drives of the current technology library. The value must be greater than or equal to zero.

port_list
Specifies a list of input or inout port names on which the **model_drive** attribute is to be set. If more than one port is specified, they must be enclosed in quotes (") or in braces ({}).

DESCRIPTION

Sets the **model_drive** attribute to a specified value on specified input or inout ports when performing synthetic library modeling. Setting the attribute sets the estimated drive of ports in terms of standard drives of the current technology library. A standard drive is equal to the output drive of a typical gate in the current target library.

To view **model_drive** values on ports, use **report_port**.

To remove **model_drive** attributes from ports, use **remove_attribute**. **reset_design** removes all attributes, including **model_drive**.

CAVEATS

This command is valid only during synthetic library modeling, and has **no effect** when run directly from dc_shell. Therefore the command also has no effect when directly elaborating DesignWare parts that contain these directives in their embedded dc_shell scripts. To set the drive in dc_shell use the set_drive command.

EXAMPLES

The following example sets the **model_drive** attribute to 2.0 on ports "A", "B" and "C".

```
prompt> set_model_drive 2.0 {A B C}
```

SEE ALSO

```
remove_attribute(2)  
report_port(2)  
reset_design(2)  
set_drive(2)  
set_model_load(2)
```

set_model_load

Sets the **model_load** attribute to a specified value on specified ports to set their load values during synthetic library modeling.

SYNTAX

```
int set_model_load  
load_value port_list
```

Data Types

```
load_value      float  
port_list       list
```

ARGUMENTS

load_value
Specifies the value to which the **model_load** attribute is to be set, thus specifying the estimated load value for the *port_list* in terms of standard loads of the current technology library. Must be greater than or equal to zero.

port_list
Specifies a list of output or inout port names on which the **model_load** attribute is to be set. If more than one port is specified, they must be enclosed in quotes (") or in braces ({}).

DESCRIPTION

Sets the **model_load** attribute to a specified value on specified output or inout ports during synthetic library modeling. Setting the attribute sets the estimated load of ports in terms of standard loads of the current technology library. A standard load is equal to the input pin capacitance of a typical gate in the current target library. Load values are stored on ports as the **model_load** attribute.

To view **model_load** values on ports, use **report_port**.

To remove **model_load** attributes from ports, use **remove_attribute**. **reset_design** removes all attributes, including **model_load**.

CAVEATS

This command is valid only during synthetic library modeling, and has **no effect** when run directly from dc_shell. Therefore the command also has no effect when directly elaborating DesignWare parts that contain these directives in their embedded dc_shell scripts. To set the load in dc_shell use the set_load command.

EXAMPLES

The following example sets the **model_load** attribute to 2.0 on ports "A", "B" and "C".

```
prompt> set_model_load 2.0 {A, B, C}
```

SEE ALSO

```
find(2)  
remove_attribute(2)  
report_port(2)  
reset_design(2)  
set_load(2)  
set_model_drive(2)
```

set_model_map_effort

Sets the **model_map_effort** attribute to a specified value on the current design, to specify the relative amount of CPU time to use during synthetic library modeling.

SYNTAX

```
int set_model_map_effort  
low | medium | high
```

ARGUMENTS

```
low | medium | high
```

Specifies the value to which the **model_map_effort** attribute is to be set, which specifies the relative amount of CPU time spent during the mapping phase of modeling. Values of 1, 2, or 3 can also be used.

DESCRIPTION

Sets the **model_map_effort** attribute to a specified value on the current design during synthetic library modeling. A value of *low*, *medium*, or *high* indicates the relative amount of CPU time to be used during the mapping phase of modeling. This command would typically be used in an embedded dc_shell script in a synthetic library part. The value of **model_map_effort** attribute (*low*, *medium*, or *high*) indicates the relative amount of CPU time to be used during the mapping phase of modeling, and overrides the default value stored in the environment variable **synlib_model_map_effort**. If the **model_map_effort** attribute is not set, the value of **synlib_model_map_effort** is used.

CAVEATS

This command is valid only during synthetic library modeling, and has **no effect** when run directly from dc_shell. Therefore the command also has no effect when directly elaborating DesignWare parts that contain these directives in their embedded dc_shell scripts. To set the model_map_effort in dc_shell use the -map_effort option of the compile command.

SEE ALSO

```
compile(2)  
set_model_load(2)  
set_model_drive(2)
```

set_multibit_options

Sets the **multibit_mode** and **minimum_multibit_width** attributes to specified values on the current design.

SYNTAX

```
int set_multibit_options  
[-default]  
[-mode multibit_mode] [-minimum_width width]
```

Data Types

<i>multibit_mode</i>	string
<i>width</i>	int

ARGUMENTS

-default
Sets the **multibit_mode** and **minimum_multibit_width** attributes so that **compile** uses their default values.

-mode *multibit_mode*
Specifies the value to give to the **multibit_mode** attribute.

-minimum_width *width*
Specifies the value to give to the **minimum_multibit_width** attribute.

DESCRIPTION

The value given to the **multibit_mode** attribute must be either *user_driven*, *structured*, *start_multibit*, or *start_singlebit*. The default for the **multibit_mode** attribute is *user_driven*.

In the *user_driven* mode, multibit components are mapped to multibit cells in the library, or to identical single bit cells if no multibit cells are available. The best implementation is chosen. This mode can potentially hurt the timing of the design, because the multibit cells are not converted to an array of single bit cells, even if that can improve timing. Use this mode if the cells to be mapped to multibit cells are known exactly.

The *structured* mode is identical to the *user_driven* mode, except that the multibit components are only implemented using identical single-bit cells in the library. Thus, multibit cells in the library are ignored. Use this mode to get a uniform structure for all cells in a multibit component.

In the *start_multibit* mode, multibit components are inferred early in the compile process. But a multibit implementation is compared to an implementation using single bit cells, and chosen only if it is better. Also, if an array of single bit cells is better for timing, such a replacement is done.

In the *start_singlebit* mode, an attempt is made to get the best quality of results,

and multibit cells are used only if the quality is not worsened. Thus, the compile process operates in exactly the same way as it would had there been no multibit components in the design. The decision to use multibit cells is made at the very end, and the replacement is done only if the quality of the design is not worsened.

Use the `start_multibit` and `start_singlebit` modes if the quality of results are more important, and the cells to be mapped to multibit cells are not known exactly.

The **`minimum_multibit_width`** attribute controls the size of the the multibit components for which the new multibit optimizations are done. Thus, multibit optimizations are done only on multibit components that have a width greater than the value specified for this attribute. Note that the components are still preserved and reported. This attribute only affects optimization. The default value for this attribute is 2.

If **`set_multibit_options`** is specified more than once on a design, the most recent settings for the attributes are used and the earlier values are removed.

EXAMPLES

The following example sets the **`multibit_mode`** attribute.

```
prompt> set_multibit_options -mode "structured"
```

This example sets the **`multibit_mode`** and **`minimum_multibit_width`** attributes to default values.

```
prompt> set_multibit_options -default
```

SEE ALSO

```
compile(2)
create_multibit(2)
current_design(2)
remove_attribute(2)
remove_multibit(2)
report_compile_options(2)
report_constraint(2)
report_multibit(2)
reset_design(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

set_multicycle_path

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.

set_multicycle_path

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:

```
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_operand_isolation_cell

Specifies a list of operators or hierarchical combinational cells to be included or excluded as operand isolation candidates.

SYNTAX

```
status set_operand_isolation_cell  
object_list  
[true | false]
```

Data Types

object_list list

ARGUMENTS

object_list
 Specifies a list of operators or hierarchical combinational cells that should be included or excluded as operand isolation candidates.

true | false
 Specifies whether to include (true) or exclude (false) the objects in the *object_list* as operand isolation candidates. The default value is true (include as operand isolation candidates).

DESCRIPTION

The **set_operand_isolation_cell** command sets attributes on the list of objects to control whether or not the objects will be considered as candidates for operand isolation. The cells in the *object_list* argument must be operators or hierarchical combinational cells.

Using this command with the optional argument true is an alternative to specifying the candidates for isolation using the HDL pragma *isolate_operands*.

Directives to include an object as an operand isolation candidate are only honored for user-driven operand isolation (when the **-user_directives** option is specified with the **set_operand_isolation_style** command). Directives to exclude an object as a potential operand isolation candidate are always honored.

EXAMPLES

The following example specifies that the combinational hierarchical cell U3/U5 should never be isolated:

```
prompt> set_operand_isolation_cell [get_cells U3/U5]
```

The next example indicates that a DesignWare adder U1/plus/plus should be isolated during the compilation phase:

```
prompt> set_operand_isolation_style -user_driven  
prompt> set_operand_isolation_cell [get_cells U1/plus/plus]
```

SEE ALSO

```
set_operand_isolation_style(2)  
do_operand_isolation(3)
```

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

```
set_operand_isolation_cell(2)  
set_operand_isolation_style(2)  
do_operand_isolation(3)
```


set_operand_isolation_slack

Sets the timing threshold to a value below which the automatic isolation roll back operation is not triggered.

SYNTAX

```
status set_operand_isolation_slack  
[slack_value]  
[-weight weight_value]
```

Data Types

<i>slack_value</i>	float
<i>weight_value</i>	float

ARGUMENTS

slack_value
Specifies the threshold value. This is a target for the worst negative slack (WNS).

-weight *weight_value*
Specifies the weight used to relax the automatic isolation roll back objective during mapping.

DESCRIPTION

Use of **set_operand_isolation_slack** is not recommended. This feature will be obsolete in a future release. The **set_operand_isolation_slack** command sets a target for the worst negative slack (WNS) to control the automatic roll back operation. The automatic roll back operation is a mechanism that removes the isolation logic after the delay optimization if the timing violation is not acceptable (that is, the negative slack exceeds the threshold).

If the threshold value is violated, the automatic roll back operation first removes all isolation logic on the critical path. It then performs timing analysis of the design and determines if the new critical path violates the threshold and contains isolation logic. If so, the automatic roll back operation repeats the above step until either the critical path meets the threshold, or there is no more isolation logic in the design. By default, Power Compiler uses 0 as the threshold value.

In a two-pass flow, no automatic roll back is performed during the initial **compile**. The automatic isolation roll back mechanism takes place during subsequent optimization with **compile -incremental**. In a one-pass flow, it is often required to perform automatic isolation roll back after insertion of operand isolation gates during mapping. The timing slack values during the initial roll back can be quite pessimistic compared to the final values after compile. To avoid unnecessary removal of operand isolation logic, you can relax the roll back timing constraint by using the **-weight** option.

The objective of automatic isolation roll back is to iteratively remove isolation

gates from the critical path until either no more isolation gates exist on the critical path, or the WNS is reduced from its starting value to the target WNS set by the **set_operand_isolation_slack** command. The weight relaxes the target reduction in WNS set forth at the beginning of automatic isolation roll back.

The weight is a value between 0 and 1. A weight equal to 0 (the default value) indicates that no rollback should be performed during the initial mapping. This corresponds to the most aggressive setting for dynamic power optimization. A weight equal to 1 corresponds to the most conservative approach, where all operand isolation on critical paths with WNS exceeding the threshold value will be removed. The weight value is ignored during incremental mapping: the timing constraint is not relaxed during automatic isolation roll back with **compile -incremental**.

Note that the threshold value and weight do not affect the behavior of manual roll back, which is invoked by the **remove_operand_isolation** command.

EXAMPLES

The following example sets the threshold of the library timing unit to 5. The automatic roll back operation does not start unless the negative timing slack of the critical path is greater than 5. During mapping, the timing constraint for automatic roll back will be relaxed by a weight of 0.5.

```
prompt> set_operand_isolation_slack 5 -weight 0.5
```

SEE ALSO

```
remove_operand_isolation(2)  
set_operand_isolation_style(2)  
do_operand_isolation(3)
```

set_operand_isolation_style

Sets the operand isolation style used by Power Compiler.

SYNTAX

```
int set_operand_isolation_style  
[-logic logic_style]  
[-user_directives]  
[-verbose]
```

Data Types

logic_style string

ARGUMENTS

-logic *logic_style*

Specifies the logic to use to isolate the inputs of modules. The valid values are **AND**, **OR**, or **adaptive** (the default). The **adaptive** value allows the tool to determine whether AND- (isolate to 0) or OR-logic (isolation to 1) is the optimal implementation.

-user_directives

Performs operand isolation in user-driven mode. In the user-driven mode, the tool isolates only the operators flagged with the **set_operand_isolation_cell** command. If this option is not specified, the tool performs automatic operand isolation by default.

-verbose

Reports the progress and the details of the operand isolation optimization step in the mapping process.

DESCRIPTION

Use of **set_operand_isolation_style** is not recommended. This feature will be obsolete in a future release. The **set_operand_isolation_style** command sets the operand isolation style that is used by Power Compiler when the operand isolation feature is enabled by setting the **do_operand_isolation** variable to **true**.

In a traditional implementation, datapath modules are always operational and they dissipate power even when their output is not used. In the operand isolation method, additional logic is inserted to stabilize the inputs of the modules when the output is not used. During scheduling, combinational module inputs are identified for isolation, and isolation banks are implemented for these module inputs.

When the logic style is set to **adaptive**, the optimal implementation (either AND or OR logic) is chosen, based on the static probability on the data net that is being isolated. The operators and hierarchical combinational cells to be isolated are also selected based on the activity of the nets in the design, unless the **-user_directives** option is specified. If **-user_directives** is specified, only the operators and hierarchical combinational cells that were included as potential

candidates with HDL pragma **isolate_operands** or with the **set_operand_isolation_cell** command are considered.

EXAMPLES

The following example sets the operand isolation style to perform isolation by AND or OR gates, selected adaptively based on the switching activity on the data net:

```
prompt> set_operand_isolation_style -logic adaptive
```

This example sets the operand isolation style to isolate only user-directed operators and hierarchical combinational cells. The isolation is performed with AND gates, and verbose reporting during mapping is enabled.

```
prompt> set_operand_isolation_style -logic AND -verbose
```

SEE ALSO

```
remove_operand_isolation(2)  
set_operand_isolation_cell(2)  
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

set_operating_conditions

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

```
compile(2)  
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_registers

Sets the **optimize_registers** attribute on the specified design or on the current design, so that **compile** automatically invokes the DC-Ultra **optimize_registers** command to retime the design during optimization.

SYNTAX

```
status set_optimize_registers
[true | false]
[-design design_list]
[-minimum_period_only]
[-sync_transform multiclass | decompose | dont_retime]
[-async_transform multiclass | decompose | dont_retime]
[-check_design [-verbose]]
[-print_critical_loop]
[-clock clock_name [-edge rise | fall]]
[-latch]
[-justification_effort low | medium | high]
[design_list]
```

Data Types

<i>design_list</i>	list
<i>clock_name</i>	string

ARGUMENTS

true | false
Specifies the value with which to set the **optimize_registers** attribute. The default value is true.

-design *design_list*
Specifies a list of designs to retime. The default is the current design.

-minimum_period_only
Indicates that only the minimum period step of the retiming algorithm (minimum clock period retiming) and not the minimum area (sequential cell count optimization) step is to be executed during compile. By default, both the minimum period and minimum area optimization steps are executed. This option is useful if you want a fast turnaround when trying to optimize the design's timing. The runtime is reduced, but your area results will not be optimal. After you are satisfied with the timing, you can attempt to reduce area by running the retiming command without this option.

-sync_transform multiclass | decompose | dont_retime
Specifies which transformation method is used for synchronous sequential cells in the design. An edge-triggered register is synchronous if none of its input pins change the outputs asynchronously. A level-sensitive latch is considered synchronous if none of its input pins can change the outputs during the clock phase where the latch is not transparent.
Selecting *multiclass* transformation specifies that the identifiable synchronous clear, set, and enable functionality is moved with the

synchronous sequential cells if they are moved during retiming. Sequential cells are classified according to their set, clear, and enable connections. The class of the sequential cells at the fanin or fanout of a combinational cell determines whether retiming across this cell can be performed.

Selecting the *decompose* transformation specifies that synchronous sequential in the design are transformed into an instance of a D-flip-flop respectively D-latch and additional combinational logic to create the necessary synchronous functionality. Only the D-flip-flop/D-latch instance can be moved during retiming.

Specifying *dont_retime* specifies that synchronous sequential cells will not be moved during retiming. Their mapping might still be changed to a different flip-flop from the technology library. The **dont_touch** attributes and retiming transformation attributes set on individual sequential cells override the value set in this option. To set retiming transform attributes, use the **set_transform_for_retiming** command. The default argument for this option is *multiclass*.

-async_transform multiclass | decompose | dont_retime

Specifies which transformation method is used for asynchronous sequential cells in the design. An edge-triggered register is asynchronous if at least one of its input pins changes the outputs asynchronously. A level-sensitive latch is asynchronous if at least one of its inputs can change the outputs during the clock phase where the latch is not transparent.

Selecting the *multiclass* transformation specifies that the identifiable asynchronous clear and set functionality, as well as any synchronous set, clear, and enable functionality, is moved with the asynchronous sequential cells, if they are moved during retiming. Sequential cells are classified according to their set, clear, and enable connections. The class of the sequential cells at the fanin or fanout of a combinational cell determines whether retiming can be performed across this cell.

Selecting the *decompose* transformation specifies that asynchronous sequential cells in the design are transformed into an instance of a flip-flop respectively latch with asynchronous set and clear inputs, as necessary, and additional combinational logic to create the necessary synchronous functionality. Only the flip-flop/latch instances can be moved during retiming. They will be classified according to their synchronous set, clear, and enable functionality.

Selecting the *dont_retime* value specifies that asynchronous sequential cells will not be moved during retiming. Their mapping might still be changed to a different flip-flop from the technology library. The **dont_touch** attributes and retiming transformation attributes set on individual sequential cells override the value set in this option. To set retiming transform attributes, use the **set_transform_for_retiming** command. The default value for this option is *multiclass*.

-check_design

Indicates that additional information about the design is to be displayed before and after retiming. This information includes the number of cells in different categories (for example, hierarchy cells with **dont_touch** attributes or non-movable sequential cells) and more detailed information about the selection of the preferred flip-flop respectively latch. Use this information to help in troubleshooting if retiming does not show the expected results.

-verbose

Indicates that the explicit names of the cells are to be displayed along with

the number of cells in each category for most categories of the **-check_design** option. The explicit naming of cells can help to locate a problem; however, the lists of output names might be long. This option can be used only with the **-check_design** option.

-print_critical_loop

Indicates that the critical loop of the design, as seen during retiming, is to be displayed. The critical loop is defined as the sequence of directly-connected combinational and sequential cells whose total combinational delay divided by the number of registers in the loop has a higher value than any other loop in the design. The critical loop limits the minimum clock period that can be achieved by retiming. Use this option to help troubleshoot problem areas of the design if the intended clock period cannot be achieved with the given number of sequential cells in the design. If you are pipelining a data path, you might need to add pipeline stages in the HDL code.

-clock *clock_name*

Specifies the name of the clock whose sequential cells are to be retimed. The clock must not be a virtual clock; it must have a associated clock port. The name of the clock is either the name specified in the **create_clock** command or the name of the clock port, if the **create_clock** command had no name specified. The registers of the clock are all sequential cells that are triggered by this clock. The connection from the clock port to the sequential cell can be through clock gating cells, buffer cells and inverter cells. If the **-clock** option is specified and edge-triggered registers are retimed, registers of other clocks are not retimed. If level-sensitive latches are retimed and the **-clock** option is specified, the latches driven by this clock as well as those driven by other clocks needed to complete a two-phase clock system are retimed. If edge-triggered registers are retimed, only registers triggered by one specific edge of the clock are retimed. By default the registers triggered by the rising edge are retimed. A different edge can be specified using the **-edge** option.

-edge rise | fall

Specifies whether the registers triggered by the rising or the falling edge of the clock are to be retimed. This option can only be used together with the **-clock** option. When level-sensitive latches are retimed this option does not matter.

-latch

Specifies that level-sensitive latches are to be retimed instead of edge-triggered sequential cells (flip-flops). If this option is used the edge-triggered sequential cells in the design will not be moved. In order to be able to retime latches they must be driven by a symmetrical two-phase clocks system. Latches that are used to prevent glitches in gated clocks will not be moved, even if the **-latch** option is used. These latches are in the fanin of clock-gating cells.

-justification_effort low | medium | high

Specifies the justification effort level that is to be used during backward justification of registers. You can specify a value of low, medium, or high. Specifying low ensures that justification terminates very quickly, but the QoR may be bad. Specifying medium provides good QoR with considerable runtime. Specifying high provides the best QoR without any regard for runtime. The default value for this option is medium.

DESCRIPTION

This command sets the **optimize_registers** attribute on the specified design or on the current design, so that the **compile** command automatically invokes **optimize_registers** to retime the design during optimization. If you specify any of the new options, then **compile** invokes **optimize_registers** with the new options. The DC-Ultra **optimize_registers** command moves registers in a design across combinational logic to achieve a target clock period, then minimizes the registers while maintaining that clock period. The **optimize_registers** attribute is particularly useful for creating embedded dc_shell compilation scripts for HDL descriptions that are to be transformed to DesignWare synthetic library components.

When **optimize_registers** is invoked during **compile** as a result of setting the **optimize_registers** attribute, the target clock period target is extracted from the timing requirements of the design. Subdesigns in the hierarchy are ungrouped into the design, unless the **dont_touch** attribute is set; for example, setting the attribute on a design is equivalent to calling **optimize_registers -flatten'** after compilation. For more information, refer to the **optimize_registers** command man page.

For more details on retiming during optimization, see the Design Compiler documentation.

To remove **optimize_registers** use the **remove_attribute** command. You can achieve the same effect by setting the **optimize_registers** attribute to false; execute **set_optimize_registers false**. The **reset_design** command removes all attributes, including **optimize_registers**.

Licensing during compile when using the **optimize_registers** attribute can be controlled by using the **compile_retime_license_behavior** dc_shell variable.

EXAMPLES

In the following example, **optimize_registers** is enabled on the TEST design:

```
prompt> set_optimize_registers -design TEST
```

In the following example, **optimize_registers** is enabled on the current design and disabled on the OLD design:

```
prompt> read TEST
prompt> set_optimize_registers
prompt> set_optimize_registers false -design OLD
```

In the following example, **optimize_registers** is enabled on the TEST design with the **-minimum_period_only** option. This means that when compile automatically calls **optimize_registers** under the hood, it invokes **optimize_registers** with **-minimum_period_only**.

```
prompt> set_optimize_registers -design TEST -minimum_period_only
```

SEE ALSO

`compile(2)`
`current_design(2)`
`optimize_registers(2)`
`remove_attribute(2)`
`reset_design(2)`

set_output_clock_port_type

Specifies the output clock port as a data port or a clock port.

SYNTAX

```
status set_output_clock_port_type  
-data | -clock  
port_list
```

Data Types

```
port_list      list
```

ARGUMENTS

-data
Specifies the port type as data. You must specify either the **-data** or the **-clock** option; they are mutually exclusive. The default type is data.

-clock
Specifies the port type as clock. You must specify either the **-data** or **-clock** option; they are mutually exclusive.

port_list
Specifies the ports to which the port type is assigned.

DESCRIPTION

This command specifies the type of output port.

An output clock port is the port that the clock can reach. An output clock port can drive either clock pins or data pins. The tool cannot determine the type of pin to which the output clock port will be connected. You must specify either **-data** or **-clock** as the port type. By default, all output clock ports will be treated as data ports.

Note that even if the port type is set to clock, the output delay will still be honored, which may impact the timing result. For this reason, it is considered best practice not to set output delay on an output clock port, if the port has already been set as clock port.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example specifies the port named OUT1 as a clock port:

```
prompt> set_output_clock_port_type -clock OUT1
```


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_physical_hierarchy

Sets a list of cells to be physical hierarchies. This command is supported only in Design Compiler topographical mode.

SYNTAX

```
status set_physical_hierarchy  
object_list
```

Data Types

object_list list

ARGUMENTS

object_list
Specifies the list of hierarchical cells to be set as a physical hierarchy.

DESCRIPTION

The **set_physical_hierarchy** command sets cells in the object list as physical hierarchies. By default, a physical hierarchy has the **dont_touch** attribute.

EXAMPLES

The following example sets the HIER_1 hierarchical cell to be a physical hierarchy:

```
prompt> set_physical_hierarchy HIER_1
```

SEE ALSO

`get_physical_hierarchy(2)`
`set_cell_location(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_pipeline_scan_data_configuration

Specifies the pipeline scan data configuration for the design.

SYNTAX

```
int set_pipeline_scan_data_configuration  
[-head_pipeline_clock clock_name]  
[-tail_pipeline_clock clock_name]  
[-head_pipeline_stages integer]  
[-head_pipeline_stages integer]  
[-tail_pipeline_stages Desired_number_of_tail_pipeline_stages]
```

ARGUMENTS

- head_pipeline_clock *clock_name***
Specifies the shift clock for the head pipeline scan data registers. All head pipeline registers will be triggered by the trailing edge of the specified clock. If no clock is specified the tool will create a new clock port. The specified clock must be previously declared as a shift clock using **set_dft_signal**.
- tail_pipeline_clock *clock_name***
Specifies the shift clock for the tail pipeline scan data registers. All tail pipeline registers will be triggered by the leading edge of the specified clock. If no clock is specified the tool will create a new clock port. The specified clock must be previously declared as a shift clock using **set_dft_signal**.
- head_pipeline_stages *integer***
Specify the number of head pipeline stages. The default value is 1. The user can choose not to insert head pipeline stages by specifying a depth of 0.
- tail_pipeline_stages *Desired_number_of_tail_pipeline_stages***
Specify the number of tail pipeline stages. The default value is 1. The user can choose not to insert tail pipeline stages by specifying a depth of 0.

SEE ALSO

reset_pipeline_scan_data_configuration(2)
set_dft_configuration(2)

set_placement_area

Creates the core placement area. This command is supported only in topographical mode.

SYNTAX

```
int set_placement_area  
-coordinate {X1 Y1 X2 Y2}  
[-fixed | -unfixed]
```

ARGUMENTS

-coordinate {X1 Y1 X2 Y2}
 {X1 Y1 X2 Y2} specify the lower left and upper right coordinates of the core area. The coordinates are in microns relative to the chip origin. This options will imply -fixed flag unless -unfixed is used explicitly.

-fixed
 Marks the core area to be fixed. The tool should not change the core area.

-unfixed
 Reset the fixed flag.

DESCRIPTION

The **set_placement_area** command defines the core placement area for rough placement. This will overwrite the previous core area.

This command can also be used to mark or unmark "fixed_core" attribute which indicate the core area may or may not be changed by the tool.

EXAMPLES

The following example shows how to set the placement area to the rectangle with lower left corner (100,100) and upper right corner (2000,2000).

```
prompt> set_placement_area -coordinate {100 100 2000 2000}
```

SEE ALSO

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_port_side

Specifies the port side constraints for core generation. This command is supported only in topographical mode.

SYNTAX

```
int set_port_side  
port_list  
-side {l | r | b | t | number}
```

ARGUMENTS

port_list

Specifies the list of ports that this command applies to.

-side {l | r | b | t | number}

Specifies the side to which the ports should be snapped. For a rectangular core outline, specify one of the following values: **l** (left), **t** (top), **b** (bottom), and **r** (right). For a rectilinear core outline, specify the edge number as specified by points on the rectilinear outline.

By default, there is no side constraint on the ports, and they are snapped to the nearest side of the block.

DESCRIPTION

The command **set_port_side** specifies the port side constraints for ports. The constraints will be used to generate coarse floorplan during synthesis.

If the port side constraints are provided, the ports will be snapped to the specified side. Otherwise, in default, the ports will be snapped to the side nearest to the port location assigned by the coarse placer.

Note that if ports with side constraints also have locations constraints set by **set_port_location**, the location constraints have higher priority during synthesis and side information will be ignored. In this case, the side information will be neither reported by **report_physical_constraints** nor written out by **write_physical_constraints**.

Use this command after loading the physical library and design, but before running synthesis commands.

If the core outline is provided then you can use the edge number as constraints in this command. Please see example below.

EXAMPLES

The following example shows how to constrain the ports clk and reset to be snapped to the left side of the block.

```
prompt> set_port_side {clk reset} -side left
```

The following example shows how to constraint the port on the right lower edge of the L - shaped rectilinear outline

```
prompt>set_rectilinear_outline -cooor {0 0 100 0 100 50 50 50 50 100 0 100}
prompt> set_port_side {ABC} -side 2
```

SEE ALSO

```
set_port_location(2)
report_physical_constraints(2)
write_physical_constraints(2)
```


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  
1688
```

```
power_enable_power_gating(3)  
target_library(3)
```

set_power_prediction

Sets the power prediction mode for **compile_ultra** or **compile_ultra -incremental**. This command is supported only in topographical mode.

SYNTAX

```
int set_power_prediction  
[true | false]  
[-ct_references lib_cell_list]
```

ARGUMENTS

```
[true | false]  
    When true, sets the power prediction mode. When false, turns off the power  
    prediction mode. When no argument is given, true is assumed.
```

```
[-ct_references lib_cell_list]  
    Restricts the list of library cells that can be used for clock tree  
    estimation.
```

DESCRIPTION

In power prediction mode, **compile_ultra** or **compile_ultra -incremental** tries to correlate the post-synthesis power numbers to that at the end of place and route. Along with improved correlation for power consumption of design components, it predicts the power consumption of missing elements in the design such as clock tree. By using **-ct_references** switch, the clock tree estimation step can be customized by specifying specific buffers and/or inverters that would be used in real clock tree synthesis.

In power prediction mode, **report_power** after **compile_ultra** reports the correlated power. Currently the different switches of **report_power** are not supported.

The power prediction mode setting is saved into the design.

The following commands automatically invoke power prediction:

```
set_max_dynamic_power  
set_max_leakage_power  
set_max_total_power  
set_power_gating_style
```

To invoke power optimization without power prediction, turn off power prediction *after* turning on power optimization.

The **report_power** command issues an error and exits if power prediction is off. To report the power of the design, turn on power prediction and run **compile_ultra -incremental**.

EXAMPLES

The following example sets the power prediction optimization mode and checks out the DC Ultra license.

```
prompt> set_power_prediction
Information: Power preduction mode successfully set. (UIO-67)
1
prompt> compile_ultra
prompt> report_power
```

The following example resets the power prediction mode.

```
prompt> set_power_prediction false
Information: Power prediction mode successfully reset. (UIO-71)
```

The following example shows how to account for power of DFT logic.

```
prompt> insert_dft
prompt> set_power_prediction
prompt> compile_ultra -incremental
prompt> report_power
```

The following example shows how power optimization turns on power prediction.

```
prompt> set_max_leakage_power 0 mW
prompt> compile_ultra -incremental
prompt> report_power
```

The following example shows how to use power optimization with out power prediction.

```
prompt> set_max_leakage_power 0 mW
prompt> set_power_prediction false
prompt> compile_ultra -incremental
prompt> # report_power is not possible here
```

SEE ALSO

`compile_ultra(2)`
`report_power(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)  
translate(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_preferred_scenario

Sets the preferred scenario.

SYNTAX

```
status set_preferred_scenario  
[scenario_name]
```

Data Types

scenario_name string

ARGUMENTS

scenario_name
 Specifies the preferred scenario name. When specified, the scenario must be an active scenario. If not specified, the preferred scenario is not set.

DESCRIPTION

The **set_preferred_scenario** command sets the preferred scenario used for multimode or multicorner optimization. When a scenario is specified as preferred, the scenario is treated as the most constraining scenario. The preferred scenario must be an active scenario, or the tool generates an error. When no scenario name is specified, the preferred scenario is reset.

Multicorner-Multimode Support

This command is not supported in a multi-scenario design flow.

EXAMPLES

The following command specifies the preferred scenario:

```
prompt> set_preferred_scenario s1  
Preferred scenario is set to s1.  
1
```

SEE ALSO

```
create_scenario(2)  
compile_ultra(2)  
set_active_scenarios(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_propagated_clock`
1696

```
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}
```

```
set_pulse_clock_cell  
1698
```

SEE ALSO

`check_timing(2)`

set_register_merging

Sets the **register_merging** attribute on the specified cells or designs, allowing register merging optimization on the objects.

SYNTAX

```
status set_register_merging  
obj_list  
[true | false]
```

Data Types

obj_list list

ARGUMENTS

obj_list
Specifies a list of cell or design names for which to enable register merging optimization. Cell names in *obj_list* must be from the current design. If multiple objects are specified, they must be enclosed in quotation marks (") or braces ({}).

true | false
Specifies the value with which to set the **register_merging** attribute. When this option is set to true (the default), register merging is enabled.

DESCRIPTION

The **set_register_merging** command sets the **register_merging** attribute on the specified *obj_list*. This attribute is used to specify the cells or designs to be optimized using register merging.

If a cell with a specified name is found in the current design, the **register_merging** attribute is set to the specified value in the cell. If no cell with a specified name is found, the tool searches for a design and the attribute set for the design.

EXAMPLES

The following example shows how to enable register merging optimization during optimization for the sequential cells named *U0* and *U1* and how to disable register merging for the cell named *U2*:

```
prompt> set_register_merging {U0 U1} TRUE  
prompt> set_register_merging U2 FALSE
```

SEE ALSO

```
compile(2)  
get_attribute(2)
```

set_register_merging
1700

`remove_attribute(2)`

set_register_replication

Sets the **register_replication** attribute on the specified sequential cells, thus allowing register replication on the objects.

SYNTAX

```
int set_register_replication  
[-max_fanout max_fanout_value]  
[-num_copies copy_value]  
object_list
```

Data Types

object_list list

ARGUMENTS

-max_fanout *max_fanout_value*
 Specifies the maximum fanout of the register after the replication.

-num_copies *copy_value*
 Specifies the number of copies the specified register is replicated.

object_list
 Specifies the set of registers to replicate.

DESCRIPTION

Sets the **register_replication** attribute on *object_list*. This attribute is used to specify how many copies the sequential cells are replicated. When the option -*num_copies* is present, the **register_replication** attribute value is the value of the option -*num_copies*. Otherwise, it is the value of the option -*max_fanout*.

If a sequential cell with a specified name is found in the *current design*, the **register_replication** attribute is set to the specified value in the cell.

EXAMPLES

The following example shows how to replicate sequential cell *U0* 3 times and how to replicate sequential cell *U1* such that its fanout is less or equal to 4.

```
prompt> set_register_replication -num_copies 3 U0  
prompt> set_register_replication -max_fanout 4 U1
```

SEE ALSO

```
compile(2)  
find(2)  
get_attribute(2)  
remove_attribute(2)
```

set_register_replication
1702

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)
translate(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
```

```
set_relative_always_on  
1708
```


SEE ALSO

`create_power_domain(2)`
`remove_power_domain(2)`
`report_power_domain(2)`

set_replace_clock_gates

Set directives for clock gate replacement. Forces the enabling or disabling of clock gate replacement for specified combinational cells in the current design. Also sets the edge type for modules or black-box cells that otherwise could not be replaced. The cells are replaced by executing the **replace_clock_gates** command.

SYNTAX

```
int set_replace_clock_gates  
[-include_cells cell_list]  
[-exclude_cells cell_list]  
[-rising_edge_clock pin_list]  
[-falling_edge_clock pin_list]  
[-undo object_list]
```

Data Types

<i>cell_list</i>	list
<i>pin_list</i>	list
<i>object_list</i>	list

ARGUMENTS

- include_cells *cell_list***
Specifies a list of cells in the current design to be included for clock gate replacement. A cell cannot be on more than one of the **-include_cells**, **-exclude_cells**, and **-undo** lists.
- exclude_cells *cell_list***
Specifies a list of cells in the current design to be excluded from clock gate replacement. A cell cannot be on more than one of the **-include_cells**, **-exclude_cells**, and **-undo** lists.
- rising_edge_clock *pin_list***
Specifies a list of pins or ports in the current design to be recognized as a positive edge triggered clock pin during clock gate replacement. A (corresponding) port cannot be on more than one of the **-rising_edge_clock**, **-falling_edge_clock**, and **-undo** lists.
- falling_edge_clock *pin_list***
Specifies a list of pins or ports in the current design to be recognized as a negative edge triggered clock pin during clock gate replacement. A (corresponding) port cannot be on more than one of the **-rising_edge_clock**, **-falling_edge_clock**, and **-undo** lists.
- undo *object_list***
Specifies a list of pins, ports or cells in the current design for which the previous specification to **set_replace_clock_gates** is to be undone. A cell cannot be on more than one of the **-include_cells**, **-exclude_cells**, and **-undo** lists, while a (corresponding) port cannot be on more than one of the **-rising_edge_clock**, **-falling_edge_clock**, and **-undo** lists.

DESCRIPTION

This command specifies combinational cells for which clock gate replacement is to be either enabled or disabled. By default, all combinational cells on the clock paths that are not buffers or inverters are considered for clock gate replacement. Cells on the **-exclude** lists will not be replaced with Power Compiler clock-gating cells, regardless of any previous specifications. If cells are specified on the **-include** lists, only those cells are considered for replacement.

This command is also used to specify the edge type of clock ports for modules or black-box cells that otherwise could not be clock gated by applying clock gate replacement to manually inserted clock gating cells. clock gated otherwise. One of the conditions for clock gate replacement is that the fanout of the manually inserted clock gate should drive only registers that are triggered by the same clock edge. If the fanout contains a black-box cell or sequential cell for which the edge type cannot be identified, or if the edge type is not compatible with the registers in the fanout of the gating cell, it is possible to override the edge detection mechanism used by **replace_clock_gates**.

The same port cannot be specified as **rising_edge** and **falling_edge** at the same time. If an instance pin is specified, the corresponding design port is used instead, since the edge type property is design specific.

You must run **set_replace_clock_edges** before issuing **replace_clock_gates**; the specifications persist during all subsequent executions of **replace_clock_gates**. Use the **-undo** option to remove the effect of an earlier invocation of **set_replace_clock_gates**.

EXAMPLES

The following example excludes the manually inserted clock-gating cells named **C12** and **C13** in the current design. All other cells are still considered for clock gate replacement.

```
prompt> set_replace_clock_gates -exclude_cells {C12 C13}
```

The following example includes the cell named **C71** in the subdesign *MID*, so that only this cell is considered for clock gate replacement at that level of the hierarchy.

```
prompt> set_replace_clock_gates -include_cells MID/C71
```

The following example sets the edge type of clock input **clk** of the black-box instance **RAM_03** in the current design to positive edge triggered.

```
prompt> set_replace_clock_gates -rising_edge_clock RAM_03/clk
```

The following example achieves the same result by specifying the edge type directly on the port of the black-box design.

```
prompt> current_design RAM
prompt> set_replace_clock_gates -rising_edge_clock clk
prompt> current_design top_level
```

SEE ALSO

`replace_clock_gates(2)`
`report_clock_gating(2)`
`set_clock_gating_style(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_resource_allocation

Sets the `resource_allocation` attribute on the current design, specifying the type of resource allocation to be used by `compile`.

SYNTAX

```
integer set_resource_allocation  
none | area_only | area_no_tree_balancing | constraint_driven
```

ARGUMENTS

none | area_only | area_no_tree_balancing | constraint_driven
Specifies the allocation type with which the **resource_allocation** attribute is to be set. The type values are as follows:

- **none** directs **compile** to use no resource sharing, so that each operation is implemented with separate circuitry.
- **area_only** directs **compile** to share operators without considering timing constraints while simultaneously balancing all arithmetic expression trees.
- **area_no_tree_balancing** directs **compile** to share operators without considering timing constraints and without balancing expression trees.
- **constraint_driven** (the default) directs **compile** to share resources so that timing constraints are met or not worsened by sharing.

DESCRIPTION

This command sets the **resource_allocation** attribute on the current design, specifying the type of resource allocation to be used by **compile**. The **synthetic_library** variable must be set for resource sharing to take affect. This command has no effect in **compile_ultra**.

If **resource_allocation** is not set, then the value of the **hlo_resource_allocation** environment variable is used. The default value for the variable is **constraint_driven**. By default, **compile** shares resources so that timing constraints are met or not worsened. Setting the **resource_allocation** attribute overrides the value of **hlo_resource_allocation** for the current design.

Use the **remove_attribute** command to undo this command.

EXAMPLES

The following example sets the **resource_allocation** attribute to **area_only** on the design named TEST:

```
prompt> current_design TEST  
prompt> set_resource_allocation area_only
```

The following example removes the **resource_allocation** attribute from the current design:

```
prompt> remove_attribute current_design resource_allocation
```

SEE ALSO

```
compile(2)  
get_attribute(2)  
remove_attribute(2)  
hlo_resource_allocation(3)  
synthetic_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_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_retention_control_pins

Converts the retention register library cell attributes in the old library format to the ones that can be used in \$retain flow. The \$retain flow requires retention register library cells to have new retention cell attributes, which is different from the original power gating flow.

SYNTAX

```
status set_retention_control_pins
[-type style]
[-power_pin_index power_pin
 | -library_pin library_pin_name]
[-is_save_pin | -is_restore_pin | -is_save_restore_pin]
lib_or_lib_cell_list
```

Data Types

<i>style</i>	string
<i>power_pin</i>	string
<i>library_pin_name</i>	string
<i>lib_or_lib_cell_list</i>	string

ARGUMENTS

-type *style*
Specifies the retention register style.

-power_pin_index *power_pin*
Specifies the power pin index number of the old retention register library pin.

-library_pin *library_pin_name*
Specifies the name of the library pin of the old retention register cell.

-is_save_pin
Assigns the specified power pin index or library pin as a SAVE pin.

-is_restore_pin
Assigns the specified power pin index or library pin as a RESTORE pin.

-is_save_restore_pin
Assigns the specified power pin index or library pin as a SAVE_RESTORE pin.

lib_or_lib_cell_list
Specifies the retention register cell library or a list of library cells to which the new attributes are applied.

DESCRIPTION

The \$retain flow requires retention register library cells to have new retention cell attributes, which is different from the original power gating flow. This

command converts the retention register library cell attributes in the old library format to the ones that can be used in \$retain flow.

EXAMPLES

The command should be specified before running the **compile** command. The following example sets the new attributes to the list of specified library cells:

```
prompt>set_retain_cell_CLK_LOW_lat \  
[get_lib_cell GS60_W_125_1.08_CORE_RET_SNPB.db/RTCLLAH*]  
  
prompt>set_retain_cell_CLK_LOW_reg \  
[get_lib_cell GS60_W_125_1.08_CORE_NSFLOP_RET_SNPB.db/RTCLDFF*]  
  
prompt>set_retention_control_pins -type CLK_LOW \  
-power_pin_index 1 -is_save_pin $retain_cell_CLK_LOW_lat  
  
prompt>set_retention_control_pins -type CLK_LOW -power_pin_index 2 \  
-is_restore_pin $retain_cell_CLK_LOW_lat  
  
prompt>set_retention_control_pins -type CLK_LOW -power_pin_index 1 \  
-is_save_pin $retain_cell_CLK_LOW_reg  
  
prompt>set_retention_control_pins -type CLK_LOW -power_pin_index 2 \  
-is_restore_pin $retain_cell_CLK_LOW_reg
```

SEE ALSO

set_retention_control_pins(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]
```

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 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.

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.

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 **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_rtl_load

Sets an RTL load value for capacitance and resistance on pins, ports, and nets.

SYNTAX

```
status set_rtl_load
[-min]
[-max]
pin_net_list
[-capacitance cvalue]
[-resistance rvalue]
```

Data Types

<i>pin_net_list</i>	list
<i>cvalue</i>	float
<i>rvalue</i>	float

ARGUMENTS

-min

Specifies that the load value is to be used for minimum delay analysis. If no minimum load value is specified, the maximum value is used. If neither **-min** nor **-max** are specified, the load values are used for both minimum and maximum delay calculations.

-max

Specifies that the load value is to be used for maximum delay analysis. If no value is specified for maximum analysis on a net, the minimum value is used. If neither **-min** nor **-max** are specified, the load values are used for both minimum and maximum delay calculations.

pin_net_list

Specifies a list of ports, pins, and nets in the current design on which the RTL loads are set.

-capacitance *cvalue*

Specifies the capacitance value with which to set the RTL load value on the pins, ports, and nets contained in *pin_net_list*. The *cvalue* must be expressed in units consistent with the technology library used during optimization. For example, if the technology library specifies capacitance values in picofarads, *cvalue* must also be expressed in picofarads. If *cvalue* is not specified, it remains unchanged from any previous setting. If *cvalue* is never set, the value defaults to zero. You must specify either **-capacitance** or **-resistance**, and you can specify both.

When *cvalue* is set on a net, the capacitance is split equally among the sticky pins of that net. For more information about sticky pins, see the "Sticky Pins and RTL Load on Nets" subsection below.

-resistance *rvalue*

Specifies the resistance value with which to set the RTL load value on the ports, nets, or pins contained in *pin_net_list*. The *rvalue* must be expressed

in units consistent with the technology library used during optimization. For example, if the technology library specifies resistance values in Kohms, *rvalue* must also be expressed in Kohms. If *rvalue* is not specified, it remains unchanged from any previous setting. If *rvalue* is never set, the value is calculated based on the specified capacitance value. For more information, see the "Calculation of Wire Delays and Capacitances" subsection below. You must specify either **-capacitance** or **-resistance**, and you can specify both. When *rvalue* is set on a net, the resistance value is assigned to all sticky pins of that net. This is not the same method used to distribute *cvalue* on a net. For more information about sticky pins, see the "Sticky Pins and RTL Load on Nets" subsection below.

DESCRIPTION

The **set_rtl_load** command the override of the calculation of wire characteristics during RTL level synthesis and other early phases of synthesis. By default, the **compile** command uses wire load models (WLMS) to calculate both the total capacitance and the time-of-flight delay of a wire. When **set_rtl_load** values are specified for particular pins, ports, or nets, the default WLM calculation is replaced by a detailed RC calculation.

Apply **set_rtl_load** only to sticky pins, or nets attached to sticky pins. For information about sticky pins, see the "Sticky Pins and RTL Load on Nets" subsection below.

The **set_rtl_load** command values are ignored during the later stages of synthesis that include the **reoptimize_design** command, because the placement-based wire capacitance and delay calculations of these commands are more accurate than the **set_rtl_load** values.

Sticky Pins and RTL Load on Nets

Annotating capacitance and resistance values on a design at the RTL level can be difficult. The objects to which you want to annotate might not exist before the first compile. Moreover, compile optimization could change or delete an object you want to annotate. The **set_rtl_load** command circumvents this problem by defining the concept of "sticky pins". Sticky pins are pins (or ports) of your design that are guaranteed to be persistent throughout the early stages of optimization. Therefore, annotated capacitance and resistance can stick to these pins and not be lost. You can apply **set_rtl_load** only to sticky pins.

A sticky pin is defined as one of the following:

- A top-level port of your design
- A pin of a hierarchical block
- A pin of a cell marked with the **dont_touch** attribute

- A pin of a cell marked with the **size_only** attribute
- A pin of a black-box cell (has a 'b' attribute in **report_cell**)
- A pin connected to a net marked with the **dont_touch** attribute.

Applying **set_rtl_load** to a net is shorthand for applying **set_rtl_load** to the sticky pins of the net. For example, consider a net that is connected to a top-level port, the pin of a RAM that is a black box, two hierarchical pins, and two leaf cells that are not marked with the **dont_touch** attribute. This net is connected to four sticky pins and two non-sticky pins. If the specified RTL load capacitance for the net is 4, the capacitance is distributed equally, with 1 to each of the four sticky pins.

If an RTL load resistance is specified for the net, that resistance is assigned to each of the four sticky pins. For example, if the specified resistance is 3, each of the sticky pins is assigned an RTL load resistance of 3.

If you want more detailed control of the assignment of RTL load capacitance and resistance on a net, assign values to each pin individually.

Calculation of Wire Delays and Capacitances

When **set_rtl_load** is in use, wire load models are switched off for the annotated wire. Instead, the values are calculated with a hybrid of annotated RTL loads and wire load models.

The total capacitance of a wire is needed in the delay calculation of the wire's driving cell. The wire capacitance is calculated as the sum of all annotated RTL load capacitances of the wire, plus an extra capacitance to account for the wiring required to connect to any nonannotated pin. For the example net with four annotated sticky pins, two hierarchy blocks, and two other leaf pins, the wire's total capacitance is the sum of the following:

- The total annotated RTL load capacitance on the four sticky pins
- The WLM calculation of the wire capacitance needed to connect the two nonsticky pins
- The WLM calculation of the wire capacitance needed to connect any nonannotated pins within the subhierarchy of the top-level block.

One implication of the method by which capacitances are calculated is that setting an RTL load capacitance is not strictly an additive process. If a net's capacitance as calculated by wire load models is 2.0, and you set an RTL load capacitance of 1.0 on the net, the resulting capacitance could be more or less than 3.0. If you intend to use **set_rtl_load** to obtain a particular fixed capacitive value, see the "Layout-based Annotation" subsection below.

The time-of-flight delay on a wire that has RTL loads is calculated in two phases: first, an RC network that represents the wire is built; second, Elmore delay calculation is applied to the RC network. The RC network is built as follows:

- Any leaf-level pin or top-level port that has an annotated RTL load is given an RC segment that is connected to that pin or port. The capacitance and resistance values of the RC segment are taken from the **set_rtl_load** values.
- Any hierarchy pin that has an annotated RTL load is given an RC segment. The capacitance and resistance values of the RC segment are taken from the **set_rtl_load** values. Wire segments within the hierarchy block are placed on one side of the pin's RC segment. Wire segments outside the hierarchy block are placed on the other side of the pin's RC segment. In this way, annotated hierarchy pins segment the wire into two parts.
- Any leaf-level pin that does not have an annotated RTL load is connected into the RC network at the appropriate hierarchy level. The capacitance and resistance of the connection is calculated using the wire load model for that hierarchy level. For example, if 3 nonannotated pins are connected to the network at the top level, the connection capacitance and resistance are calculated using the top-level wire load model for 3 connections.

Default Resistance Values

It is not necessary to specify resistance values directly to the **set_rtl_load** command. Instead, **set_rtl_load** can calculate the resistance value as a constant factor times capacitance. You determine the constant factor by setting the **rtl_load_resistance_factor** variable to the required value. The default is 0.0. For example, if you set the **rtl_load_resistance_factor** variable to 0.5, specify the RTL load capacitance of a pin as 4, and do not specify the RTL load resistance, **set_rtl_load** calculates the RTL load resistance to be 2. The factor is applied to each annotated pin of a net individually, not to the net's capacitance as a whole. The default library units are used throughout.

Reporting and Resetting RTL Load Values

Use one of the following to display a report of the current value of RTL load values on cells or nets:

```
prompt> report_cell -connections cells
```

```
prompt> report_net -connections -verbose nets
```

Use the **write_rtl_load** command to display all RTL load information in the current design.

Remove RTL load annotations using the **remove_rtl_load** command. The **reset_design** command removes all attributes from a design, including load annotations placed by **set_rtl_load**.

Layout-based Annotation

RTL loads are intended to be used for annotation of physical interconnect information, such as RC values derived from top-level floorplans. Unlike the traditional layout-based annotation commands (**set_load** for nets, **read_sdf** and

set_annotated_delay), RTL loads can be used at the RTL level and for preplaced designs. However, RTL loads are not a replacement for the traditional layout-based commands. The **reoptimize_design** command has sophisticated algorithms to incrementally update annotated capacitances and delays from the traditional layout-based commands as layout-based optimization proceeds. RTL load values are fixed. Thus, in the **reoptimize_design** and **report_timing** commands, annotations from **set_load** for nets, **read_sdf** and **set_annotated_delay** override any calculation based on RTL loads.

It is possible that your design methodology can generate early place and route results that are available before your RTL synthesis is finalized. Use current information from the layout to obtain RTL load annotation for the next pass of RTL synthesis using the **calculate_rtl_load** command. The **calculate_rtl_load** command converts back-annotated layout information from **set_load** for nets, **read_sdf**, and **set_annotated_delay** to RTL load values suitable for RTL level optimization. A typical sequence of commands is shown in the following script example:

```
/* Load your currently placed netlist */
read my_netlist.db
/* Annotate capacitance and delay information from placement */
read_sdf my_delays.sdf
include my_set_load_info.scr

/* Generate RTL load information for selected nets
 * and save the result
 */
calculate_rtl_load -cap -delay [get_nets *]
calculate_rtl_load -cap -delay [get_pins my_ram/*]
write_rtl_load -output my_rtl_load.scr

/* Apply the RTL loads to the RTL design and compile */
remove_design -designs
read my_rtl.db
include my_rtl_load.scr
compile
```

For more information, see the man page for the **calculate_rtl_load** command.

There are some caveats associated with the use of **calculate_rtl_load**. Use **set_rtl_load** and **calculate_rtl_load** only on nets that are predictably long. This includes nets that cross between floorplan blocks or nets that connect to large macro blocks. Do not use **set_rtl_load** and **calculate_rtl_load** for nets that occur deep in a physical hierarchy. Unlike the traditional layout-based annotation commands, which are kept up-to-date by **reoptimize_design**, RTL loads do not change if the layout changes. Deep in the physical hierarchy, a net that is long in one iteration of placement is not guaranteed to be long in the next iteration. Synthesis results could degrade if unreliable RTL loads are annotated onto the design.

EXAMPLES

The following example sets an RTL load capacitance of 4 on each of the specified pins and ports. The RTL load resistance is left as its default value.

```
prompt> set_rtl_load -cap 4 {my_port my_ram/ADDR[*]}
```

The following example sets maximum and minimum RTL load values on a pin:

```
prompt> set_rtl_load -max -cap 4 -res 0.5 my_hier/SIG1
```

```
prompt> set_rtl_load -min -cap 3 -res 0.0 my_hier/SIG1
```

The following example splits an RTL load capacitance of 4 across the sticky pins of a net, then assigns a resistance of 0.5 to each sticky pin.

```
prompt> set_rtl_load -cap 4 my_hier/net1
```

```
prompt> set_rtl_load -res 0.5 my_hier/net1
```

SEE ALSO

```
remove_rtl_load(2)  
report_cell(2)  
report_net(2)  
reset_design(2)  
set_dont_touch(2)  
set_size_only(2)  
set_wire_load_model(2)  
write_rtl_load(2)  
rtl_load_resistance_factor(3)
```

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_compression_configuration

Specifies the scan compression configuration for the design.

SYNTAX

```
status set_scan_compression_configuration
[-minimum_compression compression_factor]
[-xtolerance high | default]
[-synchronize_chains none | all | tail | head]
[-integration_only true | false]
[-hybrid true | false]
[-force_diagnosis true | false]
[-static_x_chain_isolation true | false]
[-chain_count scan_mode_chain_count]
[-max_length max_length_of_chains]
[-inputs number_of_inputs]
[-outputs number_of_outputs]
[-base_mode base_mode_name]
[-test_mode scan_compression_mode_name]
[-min_power true | false]
[-location compressor_decompressor_location]
[-core_port Compressor/Decompressor_interface_for_integration]
```

Data Types

<i>compression_factor</i>	integer
<i>scan_mode_chain_count</i>	integer
<i>max_length_of_chains</i>	integer
<i>number_of_inputs</i>	integer
<i>number_of_outputs</i>	integer
<i>base_mode_name</i>	string
<i>scan_compression_mode_name</i>	string
<i>compressor_decompressor_location</i>	string

ARGUMENTS

-minimum_compression *compression_factor*
Specifies the compression factor. This option impacts the number of scan chains built in the scan compression mode and the architecture of the input decompressor. The minimum value is 2. The default value is 10. The number of chains built in the scan compression mode is a function of the number of reconfigurable mode scan chains (specified with the **set_scan_configuration -chain_count M** command) and the minimum compression (specified with the **set_scan_compression_configuration -minimum_compression N** command). The number of chains in scan compression mode is $M * N * 1.2$ (pattern inflation factor).

-xtolerance high | default
Enables the insertion of fully X-tolerant scan compression structures when the value is set to high. By default (the default value), the tool does not insert fully X-tolerant scan compression structures.

set_scan_compression_configuration

`-synchronize_chains none | all | tail | head`
 Specifies the shift state of all compression mode chains. The values are as follows:

- **head** specifies that the first shift state of all compression mode chains are synchronized to the same clock edge.
- **tail** specifies that the last shift state of all compression mode chains are synchronized to the same clock edge.
- **all** specifies that both the first and the last shift states of all the compression mode chains are synchronized.
- **none** (the default behavior) specifies that no synchronization is performed.

`-integration_only true | false`
 Turns on scan compression integration. The design must have cores that contain scan compression structures inserted. This feature integrates the cores at the chip level by connecting the test signals to chip-level ports and creating a chip-level test protocol. By default, the value is false. This option is mutually exclusive with the **-hybrid** option.

`-hybrid true | false`
 Turns on the hybrid flow when set to true. This enables insertion of scan compression logic at the chip level along with integration of cores with scan compression structures in one pass. By default, the value is false. This option is mutually exclusive with the **-integration_only** option.

`-force_diagnosis true | false`
 Issues an error if a compressor cannot be built with full diagnosis capabilities when set to true. By default, the value is false.

`-static_x_chain_isolation true | false`
 Specifies that all chains containing static X scan cells will only be observed in direct observe ability modes when set to true. By default, the value is false.

`-chain_count scan_mode_chain_count`
 Specifies the number of scan compression mode chains. The tool attempts to meet this constraint. If unable to do so, the tool issues a warning and adjusts the chains accordingly. By default, the value is 1.

`-max_length max_length_of_chains`
 Specifies the maximum allowed length of scan compression mode chains. The tool attempts to meet this constraint. If unable to do so, the tool issues a warning and adjusts the chains accordingly.

`-inputs number_of_inputs`
 Specifies the number of inputs to load scan data for scan compression. The inputs are a subset of the scan inputs used for base mode (reconfigurable scan mode). By default, the value is equal to the minimum number of chains that DFT will build as if scan compression were not enabled.

`-outputs number_of_outputs`
 Specifies the number of outputs to observe scan data for scan compression.

The outputs are a subset of the scan outputs used for base mode (reconfigurable scan mode). By default, the value is equal to the minimum number of chains that DFT will build as if scan compression were not enabled.

`-base_mode base_mode_name`

Specifies the base mode (or Internal_scan) mode that is to be associated with the ScanCompression_mode indicated by the **-test_mode** switch. By default, the value is Internal_scan.

`-test_mode scan_compression_mode_name`

Specifies the scan compression test mode to which the specification applies. The default value is *all_dft*, except when **define_test_mode** is used. In this case, the default is the value of the last test_mode defined.

`-min_power true | false`

Specifies that compressor inputs are to be gated for power saving when set to true. By default, the value is false.

`-location compressor_decompressor_location`

Specifies the instance name in which the compressor and decompressor will be instantiated.

DESCRIPTION

The **set_scan_compression_configuration** command allows you to specify certain DFT MAX insertion parameters prior to using the **insert_dft** command. The parameters include scan chain specifications such as **-max_length**, **-chain_count**, and **-minimum_compression**, as well as integration strategies such as **-xtolerance**, **-integration_only**, and **-hybrid**.

The command also enables configuration of the number of scan inputs and scan outputs that are to be used for a particular **scan_compression -test_mode** command, as well as certain clock and diagnostic features that are supported.

The **-static_x_chain_isolation** option set to true takes effect only when the **-xtolerance** option is set to high. In this case, X chains are excluded from the full observation modes.

The default value for the **-min_power** option is false. When you set the value to true, you instruct **insert_dft** to create the compressor logic such that when scan enable or ScanCompression mode is inactive, the compressor logic does not toggle. This is to eliminate power consumption due to switching activity in the compressor during functional operation.

The instance name specified for the **-location** option cannot be a library cell, a black-box CTL model, or a black box.

SEE ALSO

`define_test_mode(2)`
`insert_dft(2)`
`preview_dft(2)`
`report_scan_compression_configuration(2)`
`reset_scan_compression_configuration(2)`

set_scan_compression_configuration

set_scan_configuration

Specifies the scan chain design.

SYNTAX

```
int set_scan_configuration
[-add_lockup true | false]
[-chain_count integer
  | -max_length integer_or_default
  | -exact_length integer_or_default
  | -count_per_domain integer_or_default]
[-clock_mixing no_mix | mix_edges | mix_clocks | mix_clocks_not_edges]
[-create_dedicated_scan_out_ports true | false]
[-internal_clocks single | none | multi]
[-insert_terminal_lockup true | false]
[-lockup_type latch | flip_flop]
[-mix_internal_clock_driver true | false]
[-partition vertical | horizontal]
[-
style multiplexed_flip_flop | clocked_scan | lssd | aux_clock_lssd | combinational
| none]
[-exclude_elements exclude_list]
[-voltage_mixing true | false]
[-power_domain_mixing true | false]
[-test_mode mode_name]
[-domain_based_scan_enable true | false]
[-pipeline_scan_enable true | false]
[-pipeline_fanout_limit integer_or_default]
[-reuse_mv_cells control_reuse_of_existing_MV_isolation_and_level-shifter_cells]
[-create_test_clocks_by_system_clock_domain create_test_clock_from_system_clock]
[-replace replace]
[-hierarchical_isolation hierarchical_isolation ]
[-shared_scan_in number_of_shared_pin_count]
[-preserve_multibit_segment multibit_segment]
[-begin_and_end_with_rising_edge chain_terminals_optimization]
```

Data Types

<i>integer_or_default</i>	string
<i>exclude_list</i>	cell names, instance names, or core segment names

ARGUMENTS

-add_lockup true | false
Inserts lockup latches (synchronization element) between clock domain boundaries on scan chains, when set to **true** (the default). To disable synchronization element insertion, set this option to **false**. If the scan specification does not mix clocks on chains, **insert_dft** ignores this option.

-chain_count *integer*
Specifies a positive integer for the number of chains that **insert_dft** is to build. If not specified, **insert_dft** builds the minimum number of scan chains

consistent with clock mixing constraints. The **-chain_count** and **-max_length** and **-exact_length** options are mutually exclusive.

-max_length *integer_or_default*

Specifies a positive integer that indicates the maximum chain length, or the string *default*. If this option is set with a positive integer, **insert_dft** builds scan chains without exceeding the specified length. If the option is set to *default*, **insert_dft** builds the minimum number of scan chains consistent with clock mixing constraints. The **-chain_count** and **-max_length** and **-exact_length** options are mutually exclusive.

-exact_length *integer_or_default*

Specifies a positive integer that indicates the exact chain length, or the string *default*. If this option is set with a positive integer, **insert_dft** builds, as much as possible, scan chains with the specified length. If the option is set to *default*, **insert_dft** builds the minimum number of scan chains consistent with clock mixing constraints. The **-chain_count** and **-max_length** and **-exact_length** options are mutually exclusive.

-count_per_domain *integer_or_default*

Specifies a positive integer for the number of chains that **insert_dft** is to build per clock domain. If not specified, **insert_dft** builds the minimum number of scan chains consistent with clock mixing constraints. The **-chain_count** and **-max_length** and **-exact_length** options are mutually exclusive.

-clock_mixing *no_mix | mix_edges | mix_clocks | mix_clocks_not_edges | mix_clocks_not_edges*

Specifies whether **insert_dft** can include cells from different clock domains in the same scan chain. The following allowed values control the clocking of cells in scan chains to be inserted by **insert_dft**.

no_mix The default; cells must be clocked by the same edge of the same clock.

mix_edges Cells must be clocked by the same clock, but the clock edges can be different.

mix_clocks_not_edges Cells must be clocked by the same clock edge, but the clocks can be different.

mix_clocks Cells can be clocked by different clocks and different clock edges.

-create_dedicated_scan_out_ports *true | false*

Instructs **insert_dft** to implement dedicated scan-out signal ports on the current design, when set to **true**. When set to **false** (the default), **insert_dft** uses mission-mode ports as scan-out ports whenever possible.

-internal_clocks *single | none | multi*

Applies only to the multiplexed flip-flop scan style, and is ignored for other scan styles.

An internal clock is defined as an internal signal driven by a multiplexer (or multiple input gate) output pin.

Note that the output of clock gating cells that are introduced by the Synopsys

Power Compiler tool are not be treated as internal clocks even if this **internal_clocks** option is enabled.

The **-internal_clocks** option is common to all multi mode scenarios. Hence this option can be specified only if the **-test_mode** option value is 'all'.

Otherwise, the tool will show a warning message and ignore the **internal_clocks** option and accept the **set_scan_configuration** command specification with the remaining options. In such cases, the user needs to respecify the **-internal_clocks** option with **-test_mode** 'all'.

The internal clocks option can be set to the following values:

single - **insert_dft** treats any internal clocks in the design as separate clocks for the purpose of scan chain architecting. The **single** value stops at the first buffer or inverter driving the flip-flops clock.

none (the default) - **insert_dft** does not treat internal clocks as separate clocks. This is the default value for **internal_clocks** option.

multi - **insert_dft** treats any internal clocks in the design as separate clocks for the purpose of scan chain architecting. The **multi** value jumps over any buffers and inverters, stopping at the first multi-input gate driving the flip-flops clock.

-insert_terminal_lockup true | false

Inserts synchronization element at the end of scan chains when set to **true**. The default value is **false**. If the scan style is not **multiplexed_flip_flop**, **insert_dft** ignores this option.

-lockup_type latch | flip_flop

Selects the type of synchronization element used in the scan chain. The default lock-up type is a level-sensitive latch. If you choose **flip_flop** as the lock-up type, an edge-triggered flip-flop is used as the synchronization element. In either case, the cell is not constrained from the target library used for mapping the synchronization element. You can use this option in conjunction with **-add_lockup** option or **-insert_terminal_lockup** option.

-mix_internal_clock_driver true | false

When **true**, **insert_dft** allows mixing in a scan chain of any internal clocks that are derived from a single external clock pin. This capability can only be activated when the clock mixing scan configuration is either **no_mix** or **mix_edges**.

-partition vertical | horizontal

Allows you to make the physical scan cells partitioning in a vertical or horizontal fashion. The default is *horizontal*.

-style multiplexed_flip_flop | clocked_scan | lssd | aux_clock_lssd | combinational | none

aux_clock_lssd | combinational | none

Identifies the scan style. Select *none* if you have not selected a scan style for the design. By default, **insert_dft** uses the scan style value specified by environment variable **test_default_scan_style** in your **.synopsys_dc.setup** file.

-exclude_elements *exclude_list*

Specifies the list of cell names to exclude during scan chain insertion. In other words, any cell name provided as input to this option will not be as part of the scan chain after scan insertion. This option also accepts design instance names. If a design instance name is specified, all the elements in

that instance will be excluded from the scan chain. If the specification is a design instance name which is scan inserted already, all the core segments with in this instance will be ignored. If the input is a segment name, then that segment alone will be excluded during scan insertion. This option also accepts wild cards and collection inputs. Provide all these entries as a list. Refer to the example at the end of this man page. This option is only available in incremental mode and is cumulative.

`-voltage_mixing true | false`

Allows **insert_dft** to insert scan elements from different voltage domains into the same scan chain, when set to **true**. This requires you to insert level shifters after scan insertion is finished. The default value is **false**.

`-power_domain_mixing true | false`

Specifies that **insert_dft** can mix scan elements from different power domains into the same scan chain, when set to **true**. This require you to insert isolation cells after scan insertion is finished. The default value is **false**.

`-test_mode mode_name`

Indicates which test mode the scan configuration applies to.

`-domain_based_scan_enable true | false`

When *true*, **fpreview_dft/Binsert_dft** creates one scan enable signal per clock domain. This process is independent of the value of the "-clock_mixing" option. The default value is **false**.

`-pipeline_scan_enable true | false`

When *true*, **preview_dft/insert_dft** creates a scan enable pipelining stage for each scan enable signal. This option only takes effect if the "-domain_based_scan_enable" option is set to **true**. The default value is **false**.

`-pipeline_fanout_limit integer_or_default`

When *true*, **preview_dft/insert_dft** creates one scan enable signal so that the number of flip-flops driven by the same scan enable signal does not exceed the value. This option only takes effect if the "-domain_based_scan_enable" option is set to **true**. The default value is **false**.

EXAMPLES

The following example allows DFT Compiler to establish the maximum sequential length of scan chains (30 in this example):

```
prompt> set_scan_configuration -max_length 30
```

The following example allows DFT Compiler to exclude the cell names, segment names, wildcard entries and collections using the **-exclude_elements** option. Note that the input is provided as a list for this option. In this example, *foo* is the collection defined. The **exclude_elements** option takes *s2/ff1* which is a cell name, *s1/f**, which is a wild card representation, *s0/1* which is a segment name, *\$foo* for accepting the collection input, and *U5* for the design instance name.

```
prompt> set foo [get_cell s4/*]
```

```
prompt> set_scan_config -exclude_elements \
```



```
[list s2/ff1 s1/f* s0/1 $foo U5]
```

SEE ALSO

```
current_design(2)  
insert_dft(2)  
preview_dft(2)
```

set_scan_element

Sets the **scan_element** attribute on specified design objects, to determine whether **insert_dft** replaces them with scan cells.

SYNTAX

```
int set_scan_element  
true | false  
cell_design_ref_list
```

Data Types

cell_design_ref_list list

ARGUMENTS

true | false

The Boolean value with which to set the **scan_element** attribute on the *cell_design_ref_list*. When **true**, (the default), the specified objects are replaced by sequential cells in the design. When **false**, scan replacement is disabled.

cell_design_ref_list

A list of design objects for scan replacement. Objects must be of the following *sequential* types:

- cells (such as flip-flops and latches)
- hierarchical cells (containing flip-flops or latches)
- references
- library cells
- designs

DESCRIPTION

Sets the **scan_element** attribute to **true** on objects in *cell_design_ref_list*, indicating that **insert_dft** is to replace them with scan cells and make them part of the scan path.

Unless you want to change a previous setting of **false**, it is unnecessary to set the **scan_element** attribute to **true** because, by default, the **insert_dft** command replaces all nonviolated sequential cells with equivalent scan cells for full scan designs. Sequential cells with the **scan_element** attribute set to **false** are not to be replaced by equivalent scan cells.

Sequential cells violated by *fdft_drc* are not replaced by equivalent scan cells, regardless of their **scan_element** attribute values.

For hierarchical cells, the **scan_element** attribute value applies to all sequential leaf cells within the hierarchical cell. For references, library cells, or designs, the **scan_element** attribute value applies to the instances of these objects, or to the sequential leaf cells within the hierarchy of the instances.

The effects of the **scan_element** attribute are applied hierarchically in a design. The value of the **scan_element** attribute on a lower-level object in the design takes precedence over the value of the **scan_element** attribute on a higher level object in the design. For example, if the value of the **scan_element** attribute on a higher-level object in the hierarchy is **false** and the value of the **scan_element** attribute on a sequential element inside the higher-level object is **true**, the sequential element is scan replaced.

The **scan_element** attribute value on a cell instance takes precedence over the **scan_element** value on the reference for that cell. The **scan_element** attribute value on a library cell is treated as a technology limitation and cannot be overwritten in any way.

The **set_scan_element** command supports full instance-based specification. However, the **scan_element** attributes on instances are specific to the current design. For example, if you start with a lower-level design, use the **set_scan_element** command to put **scan_element** attributes on instances in this context, and change the current design to a higher-level design, the **scan_element** attributes are no longer visible at the higher level.

To remove individual **scan_element** attributes, use the **remove_attribute** command. To remove all attributes, use the **reset_design** command.

EXAMPLES

The following example specifies that the **insert_dft** command is not to scan replace three sequential cells in the current design.

```
prompt> set_scan_element false {U3_1 U3_2 U3_3}
```

The following example specifies that the **insert_dft** command is to scan replace every instance of cell DFF in the current design.

```
prompt> set_scan_element true [get_references DFF]
```

The following example specifies that the **insert_dft** command is not to scan replace instances of library cell DLATCH.

```
prompt> set_scan_element false tech_lib/DLATCH
```

The following example specifies that the **insert_scan** command is not to scan replace instances of design REGFILE (sequential cells within REGFILE are not to be replaced by equivalent scan cells).

```
prompt> set_scan_element false [get_designs REGFILE]
```

SEE ALSO

```
current_design(2)
dft_drc(2)
get_attribute(2)
insert_dft(2)
preview_dft(2)
remove_attribute(2)
```

```
reset_design(2)
```

set_scan_group

Specifies an unordered group of cells that are not yet connected, but should be kept together within a scan chain. Also identifies existing logic in the current design that is to be designated as a scan segment.

SYNTAX

```
status set_scan_group
scan_group_name
[-access signal_type_pin_pairs]
[-include_elements member_list]
[-serial_routed true | false]
[-segment_length length_of_virtual_segment]
[-clock top_level_clock_port]
[-edge rising | falling]
[-class occ | wrapper | input_wrapper | output_wrapper]
```

Data Types

<i>scan_group_name</i>	string

ARGUMENTS

`scan_group_name`
Specifies the name of the scan group. The scan group must be uniquely identified as a design object. The name of the scan group name must also be unique.

`-access signal_type_pin_pairs`
Lists ordered pairs, each consisting of a scan signal type and a design pin, indicating how the **insert_dft** command is to access the scan group. Valid signal types are ScanClock, ScanMasterClock, ScanSlaveClock, ScanEnable, ScanDataIn, and ScanDataOut. Validation ensures that specified signal types are consistent with the design scan style. Note that this list is mandatory when the **-serial_routed** option is set to true. And this option is not supported when **-serial_routed** is set to false.

`-include_elements member_list`
Displays an unordered list of scan group components. This list can include sequential cells, segment names and design instances.

`-serial_routed true | false`
Identifies whether the scan group is composed of serially-routed sequential cells or that you are specifying an unordered group of sequential cells. The default value is false.

`-segment_length length_of_virtual_segment`
Specifies the length of the virtual segment. This option can be used only when **-serial_routed** is set to true. With this option, you must also use **-access** to specify the correct input and output access pins, and use **-clock** for top-level clock specifications with respect to this segment.

`-clock top_level_clock_port`
 Specifies the top level clock port or pin details with respect to the virtual scan segment.

`-edge rising | falling`
 Specifies the edge of the clock associated with the virtual scan segment. Scan architect will use this information for architecting the scan chains.

`-class occ | wrapper | input_wrapper | output_wrapper`
 Specifies the class of the scan group. The **occ** keyword specifies that the scan group is including OCC clock chain segments. Scan groups of **-class occ** are used only during OCC flows.
 The **wrapper** keyword specifies that the scan group is including wrapper chain segments. Scan groups of **-class wrapper** are used only during core wrapping flows.
 The **input_wrapper** keyword specifies that the scan group is including input wrapper chain segments. Scan groups of **-class input_wrapper** are used only during core wrapping flows.
 The **output_wrapper** keyword specifies that the scan group is including output wrapper chain segments. Scan groups of **-class output_wrapper** are used only during core wrapping flows.

DESCRIPTION

The **set_scan_group** command identifies existing logic in the current design that is to be designated as a scan group. Scan groups implement scan in a particular scan style. DFT Compiler can include scan groups in scan chains by connecting their access pins (when serially routed), or maintain the cells' groups without assuming they are serially connected to each other. The **insert_dft** command does not scan-replace scan group members explicitly.

When the **-serial_routed** option is set to true, the **set_scan_group** command allocates sequential cells to scan groups and identifies design pins as scan group access pins with particular scan signal-type semantics. The **set_scan_configuration** command used with the **-style** option determines the scan group scan styles.

The **set_scan_group** command options are not incremental. A **set_scan_group** option that specifies a scan group name overwrites any previous **set_scan_group** command that uses the same name and has the same scan style.

The **set_scan_group** command accepts a list of members. By default, the list of members is unordered and not serially connected. This list must be ordered when **-serial_routed** is set to true.

The scan group members should belong to the same clock domain and same edge. If the scan group is provided with elements from a different clock domain, an error message is displayed during scan architecting and the scan group specification is discarded.

The **set_scan_group** command does not accept collection input.

The **set_scan_group** command accepts design instances as scan group members. It adds every scannable cell in the design instance to the scan group at the specified position, in alphanumeric order. This specification is valid only when **-serial_routed** is set to false.

The **-include_elements** option cannot contain segment names or design instance names when **-serial_routed** is set to true. When **-serial_routed** is set to true, you must specify only cell names in the **-include_elements** list.

The **-segment_length** option specifies the length of the virtual scan segment. Virtual segments are segments for which you specify the scan segment input and the scan segment output using **-access**, and the scan segment clock association using **-clock** option. If you do not specify **-include_elements** but want specify the length of a segment, use the **-segment_length** option. The tool connects only to the hookup points during scan chain construction and does not trace within the hookup points (or virtual scan segment). The segment length you provide is considered for top-level scan chain balancing.

Note that the post-dft_drc and scan_def generation support is not available when virtual scan segments are present in the scan chain.

The **-clock** option is specifies the top-level clock details which drive the virtual scan segment. If you specify **set_scan_group** with the **-segment_length** option, you must also specify a valid **-clock** specification.

Ensure that the clock name specified as part of the **-clock** option is already defined using the **set_dft_signal** command. Alternatively, run the **set_scan_group** command after creating the test protocol and inferring the clocks and asynchs.

The **-edge** option specifies the controlling edge of the clock with respect to the virtual scan segment. The controlling edge is specified as rising or falling. If **-edge** option is not used, by default, the rising edge of the clock is used as the controlling edge of the virtual scan segment.

When the **-access** option is used, the access pin signal types must be consistent with the segment scan style. Access pin directions must be consistent with their signal types. Access pins cannot be associated with more than one signal type. Scan groups cannot have more than one ScanDataIn or ScanDataOut access pin. Scan groups cannot contain duplicate members; sequential cells cannot belong to more than one scan group. Scan group members cannot also belong to scan chains. In other words, a sequential cell element that belongs to a scan group cannot be specified as part of the **set_scan_path** command.

A scan group does not accept another scan group as a member; nested scan groups are not supported.

Valid scan group error messages are TEST-400 and TEST-700. TEST-400 is displayed when scan group elements belong to more than one clock domain or have the same clock domain but a different edge. TEST-700 is displayed when scan group elements have a non-existing element specified as part of the scan group specification.

Use the **preview_dft** command with the **-bsd** option set to all to review your scan group specifications. Use the **remove_scan_group** command with *scan_group_name* argument to delete an existing scan group specification.

EXAMPLES

The following example identifies an embedded shift register in the multiplexed flip-flop scan style. This specification can be completed in a single **set_scan_group**

execution by including the identification of the ScanEnable access pin in the first **-access** option specification.

```
prompt> set_scan_group my_shift_reg \  
        -access [list ScanDataIn P/B/U7/si ScanDataOut P/B/U9/QN] \  
        -include_elements [list P/B/U7 P/B/U8 P/B/U9] \  
        -serial_routed true  
  
prompt> set_scan_group my_unconnected_group \  
        -include_elements [list U1 U2]
```

Example of a virtual scan segment controlled by the rising edge of the clock tck

```
prompt> set_scan_group virtual_scan_segment \  
        -access [list ScanDataIn P/B/U7/si ScanDataOut P/B/U9/QN] \  
        -segment_length 20 \  
        -clock tck \  
-edge rising \  
        -serial_routed true
```

Example of a virtual scan segment controlled by the falling edge of the clock tck

```
prompt> set_scan_group virtual_scan_segment \  
        -access [list ScanDataIn P/B/U7/si ScanDataOut P/B/U9/QN] \  
        -segment_length 20 \  
        -clock tck \  
-edge falling \  
        -serial_routed true
```

SEE ALSO

```
insert_dft(2)  
preview_dft(2)  
remove_scan_group(2)
```


set_scan_link

Declares a scan link for the current design.

SYNTAX

```
int set_scan_link  
scan_link_name Wire | Lockup  
[-test_mode mode_name]
```

Data Types

scan_link_name string

ARGUMENTS

scan_link_name
 Specifies the name of the scan link. Substitute the name you want for *scan_link_name*.

-test_mode *mode_name*
 Specifies the test mode for the specification. The default is current mode.

DESCRIPTION

Declares a scan link for the current design. Scan links connect scan cells, scan segments, and scan ports within scan chains. DFT Compiler supports scan links that are implemented as wires (type Wire) and scan-out lock-up latches (type Lockup).

This command reserves words for particular types of scan links. Use these words within **set_scan_path** commands to specify scan links of various types. You can use scan links to insert lock-up latches at specific scan chain locations and correct timing problems. You can also use wire scan links to override the automated insertion of lock-up latches. The **scan_link_name** option must uniquely identify the scan link as a design object.

You can use the **-test_mode** option to specify the test mode.

To review your scan link specifications, use the **preview_dft** command with the **-show** option set to **cells**. To create scan links and add them to the current design, use the **insert_dft** command. To delete scan link specifications, use the **remove_scan_link** command.

The **set_scan_link** command does not invalidate **dft_drc** modeling information.

EXAMPLES

The following example declares a lock-up latch named *a_lckup* in Internal_scan mode and uses it to resynchronize the scan outputs of the elements named A and F of chain3.

```
prompt> current_design WC66
prompt> set_scan_link a_lckup Lockup -test_mode Internal_scan
prompt> set_scan_path chain3 {A a_lckup B C D E F a_lckup G}
prompt> insert_dft
```

SEE ALSO

```
current_design(2)
dft_drc(2)
insert_dft(2)
preview_dft(2)
remove_scan_link(2)
report_scan_link(2)
set_dft_signal(2)
set_dont_touch(2)
set_scan_configuration(2)
set_scan_element(2)
set_scan_path(2)
write(2)
test_default_scan_style(3)
```

set_scan_path

Specifies a scan chain for the current design.

SYNTAX

```
integer set_scan_path  
scan_chain_name  
[-ordered_elements ordered_list]  
[-head_elements head_list]  
[-tail_elements tail_list]  
[-include_elements include_list]  
[-infer_dft_signals]  
[-dedicated_scan_out true | false]  
[-complete true | false]  
[-exact_length length]  
[-insert_terminal_lockup true | false]  
[-scan_master_clock clock_name]  
[-edge rising | falling]  
[-scan_slave_clock clock_name]  
[-scan_enable port_name]  
[-scan_data_in port_name]  
[-scan_data_out port_name]  
[-test_mode mode_name]  
[-view view_name]  
[-class scan | wrapper | bsd]  
[-hookup hookup_pin_list]  
[-input_wrapper_cells_only enable | disable]  
[-output_wrapper_cells_only enable | disable]  
[-pipeline_head_registers scan_chain_element_names]  
[-pipeline_tail_registers scan_chain_element_names]  
[-bsd_style global | synchronous | asynchronous]
```

Data Types

<i>scan_chain_name</i>	string
<i>ordered_list</i>	list
<i>head_list</i>	list
<i>tail_list</i>	list
<i>include_list</i>	list
<i>length</i>	integer
<i>clock_name</i>	string
<i>port_name</i>	string
<i>mode_name</i>	string
<i>view_name</i>	string
<i>hookup_pin_list</i>	list

ARGUMENTS

scan_chain_name
Specifies a name for the scan chain.

`-ordered_elements ordered_list`

Defines scan chain elements. This ordered list can include names of sequential cells, design instances, scan segments, and scan links. The input is accepted as a list and not as a collection.

`-head_elements head_list`

Specifies that the list of scan chain elements should be placed at the beginning of the scan chain. This ordered list can include names of sequential cells, design instances, scan segments, and scan links. The input is accepted as a list and not as a collection.

`-tail_elements tail_list`

Specifies that the list of scan chain elements should be placed at the end of the scan chain. This ordered list can include names of sequential cells, design instances, scan segments, and scan links. The input is accepted as a list and not as a collection.

`-include_elements include_list`

Specifies that the list of scan chain elements should be included in the scan chain and the scan architect is free to place them anywhere in the scan chain. This ordered list can include names of sequential cells, design instances, scan segments, and scan links. The input is accepted as a list and not as a collection.

`-infer_dft_signals`

Specifies that the **set_scan_path** command automatically infers the ScanDataIn, ScanDataOut, and ScanEnable DFT signals that are normally specified using the **set_dft_signal** command. This option is only applicable when you are using this command in a scan extraction flow.

Note that if you use this option, and you later use the **remove_scan_path** command to remove scan paths, the signals that were inferred are not removed. You must use the **remove_dft_signal** command to explicitly remove these DFT signals.

`-dedicated_scan_out true | false`

Forces the use of a dedicated scan-out port when set to **true**. The last scan cell in a chain can drive a port in functional mode. By default, the **insert_dft** command uses this port as a scan-out port. When true, the **set_scan_path** command builds a dedicated scan-out port for the scan chain.

`-complete true | false`

Indicates whether DFT Compiler can add components to a specified scan chain. When true, **insert_dft** does not add any other scan cells to this scan chain and treats it as a complete scan chain. When false (the default), **insert_dft** can add more scan cells to this specification to balance scan chains. The new scan cells are embedded at the beginning of the scan chain.

`-exact_length length`

Specifies that **insert_dft** command build the *scan_chain_name* scan chain with the exact sequential length indicated by *length*. The allowed value is positive integer. If the scan requirements cannot be met, the tool generates a warning stating that the specified scan chain requirements cannot be honored and then proceeds with the classic scan architecting.

If you specify **-class bsd**, this option allows you to specify the length of BSD Test Data Register.

`-insert_terminal_lockup true | false`
 Specifies a lockup latch to be associated at the end of the scan chain. Scan architect takes the user-defined lockup latch specification into consideration while building scan chains.

`-scan_master_clock clock_name`
 Specifies a clock to be associated with a scan chain. Scan architect takes the user-defined clock domain into consideration while building scan chains.

`-edge rising | falling`
 Specifies the edge of the `scan_master_clock` associated with the scan chain. Scan architect includes only the elements controlled by the specified edge of the `scan_master_clock` in the scan chain.

`-scan_slave_clock clock_name`
 Specifies a clock to be associated with a scan chain. Scan architect takes the user-defined clock domain into consideration while building scan chains.

`-scan_enable port_name`
 Specifies a scan enable to be associated with a scan chain. Scan architect takes the user-defined scan enable port into consideration while allocating scan enable to scan chains. The specified port is dedicated to the scan chain. It will not be used to drive scan enable pins of flip-flops in other chains. If the internal pins flow is enabled using the **set_dft_drc_configuration** command, a hookup pin can be specified as a scan enable for a scan chain. When you specify **-class wrapper**, the specified port name can be a wrapper shift enable.

`-scan_data_in port_name`
 Specifies a scan in port to be associated with a scan chain. Scan architect takes the user-defined scan in port into consideration while building scan chains. However, if the internal pins flow is enabled using the **set_dft_drc_configuration** command, a hookup pin can be specified as a scan in for a scan chain.

`-scan_data_out port_name`
 Specifies a scan-out port to be associated with a scan chain. Scan architect takes the user-defined scan-out port into consideration while building scan chains. However, if the internal pins flow is enabled using the **set_dft_drc_configuration** command, a hookup pin can be specified as a scan out for a scan chain.

`-test_mode mode_name`
 Indicates the test mode to which the scan path constraint applies.

`-view view_name`
 Indicates the view to which the specification applies. The following views are valid:

- **existing_dft** implies that the specification refers to the existing usage of a scan chain. This used when working with DFT inserted designs. For example, to describe to the tool that chain A is being used, enter the following:

```
set_scan_path A -view existing_dft \
```

`-ordered_elements {reg0 reg1 reg2}`

- **spec** (default) implies that the specification refers to scan chains that the tool must use during DFT insertion. This is used when preparing a design for DFT insertion. For example, to prescribe to the tool that a scan chain A must be used as a Test Enable, you would do the following

set_scan_path A -view spec -ordered_elements {reg0 reg1 reg2}

`-class scan | wrapper | bsd`

Specifies the class of the scan chain.

- **scan** (default) specifies that the chain is a scan chain. Chains of class scan are used only during scan insertion.
- **wrapper** specifies that the chain is a wrapper chain. Chains of class wrapper are used only during wrapper insertion.
- **bsd** specifies that the chain is a BSR chain or a BSD register. These chains and registers are used only during BSD insertion.

`-hookup hookup_pin_list`

Specifies the hookup pin names of the scan chain. The *hookup_pin_list* can contain hierarchical pin names or pin objects.

These pin names are used in defining a bsd register. The signal types of these pins are specified using *set_dft_signal* command. These pin names for a *bsd* chain along with the signal types as defined by *set_dft_signal* command can define a BSD register completely.

By default, capture and update clock pins of the BSD Test Data Register (TDR) are not gated, which means the pins are always driven by TCK.

To gate the clock pins of TDR with a MUX so that the pins are driven by TCK when the instruction that selects the TDR is active and driven by system logic when the instruction is not active, set the following variable to true prior to BSD insertion.

set test_bsd_synthesis_gated_tck true

There is no default value for this option.

`-input_wrapper_cells_only enable | disable`

Specifies whether or not only input wrapper cells should be used in the chain. This option is specific to wrapper chains. The default value of this option is disable.

`-output_wrapper_cells_only enable | disable`

Specifies whether or not only output wrapper cells should be used in the chain. This option is specific to wrapper chains. The default value of this option is disable.

`-pipeline_head_registers scan_chain_element_names`

Specifies an ordered list of connected sequential elements starting from the scan input port. In regular scan mode, the scan data goes through these elements after the scan input port and before the first scan cell of the specified scan chain. In adaptive scan mode, the scan data goes through these

elements after the scan input port and before the adaptive scan decompressor. These elements are automatically excluded from scan architecting. This ordered list must only include instance names of sequential cells. If the scan input port attached to the scan chain does not have a hookup pin specified, the last element that is listed drives the scan chain using the first output pin found. The input is accepted as a list and not as a collection.

`-pipeline_tail_registers scan_chain_element_names`

Specifies an ordered list of connected sequential elements that end at the scan output port. In regular mode scan, the scan data goes through these elements after the last scan cell of the specified scan chain and before the scan output port. In adaptive scan mode, the scan data goes through these elements after the adaptive scan compressor output and before the scan output port. These elements are automatically excluded from scan architecting. This ordered list must only include instance names of sequential cells. If the scan output port attached to the scan chain does not have a hookup pin specified, the first element that is listed observes the scan chain using the first input pin found. The input is accepted as a list and not as a collection.

DESCRIPTION

This command specifies a scan chain for the current design. The command allocates scan cells, scan segments, and scan links to scan chains, and specifies scan chain orderings. The **set_scan_configuration** command **-style** option determines the chain scan style. It can be used either to describe the existing usage or to prescribe the usage during implementation.

Scan chain elements cannot belong to more than one chain. When **set_scan_path** options conflict, the most recent command overrides the previous command.

The **set_scan_path** options are not incremental. A **set_scan_path** option that specifies a scan chain name overwrites any previous **set_scan_path** option that uses the same name.

The **set_scan_path** command accepts design instances as scan chain elements. It assumes you want to add every scannable cell in the design instance to the scan chain at the specified position in alphanumeric order.

The sequential length of a scan chain can be user-defined by using the **-exact_length** option. Use care to ensure that the local specification can be implemented correctly. If the local specification cannot be implemented, the tool issues a warning message and builds the scan chain with classic scan chain architecting.

You can associate a scan chain with a clock domain. Use **-scan_master_clock** to identify the clock domain to be used in building the scan chain. In case you use lssd scan cell, you can specify the slave clock by using **-scan_slave_clock**. This option is only available in descriptive view. You must take into account the clock mixing configuration.

You can also associate a specific clock domain corresponding to a rising or a falling edge of a clock with a scan chain by using the **-edge** along with **-scan_master_clock**. If an edge is specified along with a clock for a scan chain, then

the scan architecting ignores the clock mixing configuration while building that scan chain and only the cells controlled by the specified edge are considered.

You can also associate a scan chain with a specific scan in, scan out and scan enable port by using **-scan_data_in**, **-scan_data_out**, and **-scan_data_enable**, respectively.

Scan segments can be subdesign scan chains. Suppose the hierarchical design instance *instance_name* has a scan chain of **report_dft**. The **-scan_path** option reports as "Complete scan chain #n." You can include this subdesign scan chain in a scan chain by referring to it as **instance_name/n**.

Use the **set_scan_path** command with the **-dedicated_scan_out** option to specify whether scan chains have dedicated scan-out ports.

The **insert_dft** command can insert additional scan elements before identified scan elements, unless you execute the **set_scan_path** command with the **-complete** option set to **true**.

Scan chain orderings must satisfy clock domain constraints asserted by using the **set_scan_configuration** command with the **-clock_mixing** option. The **insert_dft** command resynchronizes nonfunctional scan chains by using lockup latches. DFT Compiler reorders nonfunctional scan chains if you disable lockup latch insertion by using the **set_scan_configuration** command with the **-add_lockup** option.

Use the **set_scan_path** command to add a lockup latch at the end of your scan chain by using the option **-insert_terminal_lockup**.

The **set_scan_path** command does not invalidate **dft_drc** modeling information.

Use the **report_scan_path** command to review your scan chain specification. Use the **insert_dft** command to implement the scan chain. Use the **remove_scan_path** command to delete scan chain specifications.

In the case of multimode scan insertion, if you have previously defined several test modes using the **create_test_schedule** or **define_test_mode** command, the default **set_scan_path** command is not associated with any mode and is not honored. Use the **-test_mode** option to specify the test mode to which you want your scan path constraint to apply.

When using this command in a scan extraction flow, either specify the existing DFT signals using the **set_dft_signal -view existing** command prior to running the **set_scan_path** command, or use **set_scan_path** with the **-infer_dft_signals** option to infer these signals.

To define a BSD register, specify **-class bsd** and specify the register access pins using **-hookup**. You must also specify the signal types of the hookup pins by using the **set_dft_signal** command.

EXAMPLES

The following example defines a complete scan chain named chain3, consisting of sequential cells A, B, and C in reverse alphanumeric order.


```
prompt> current_design WC66
prompt> set_scan_path chain3 {C B A} -complete true
```

The following example shows how to use the **-head_elements**, **-tail_elements**, and **-include_elements** options to specify that the elements from the U4 instance are at the beginning of the chain and the elements from the U5 instance are at the end of the chain. The elements from U3 are ordered and scan architect does not change the order of the elements in this instance during architecting. The **-include_elements** option ensures that the U2/2 segment is included in scan chain 1.

```
prompt> set_scan_path 1 -view spec -head_elements {U4} \
        -include_elements {U2/2} -ordered_elements {U3} \
        -tail_elements {U5}
```

The following example shows how to build scan chains with user-defined sequential lengths and/or clocks in a design with 15 scan cells, 10 of which belong to clock domain A and 15 of which belong to clock domain B. This builds chain1 with 7 sequential elements clocked with A, chain2 with 15 sequential elements clocked with B and chain3 with the 8 remaining elements clocked with A.

```
prompt> set_scan_configuration -clock_mixing no_mix
prompt> set_scan_path chain1 -exact_length 7 -clock A
prompt> set_scan_path chain2 -scan_master_clock B
```

The following examples show how to use the **-edge** option along with the **-scan_master_clock**. First, the **fbset_scan_path** command assigns all elements controlled by the rising edge of the clk clock to chain1, and then assigns all elements controlled by the falling edge of the clk clock to chain2.

If **set_scan_path** specifications assign elements controlled by the same edge of a clock to more than one scan chain, the scan architect tries to assign an equal number of elements to each individual scan chain. Consider **set_scan_path** specifications for chain3 and chain4 below. If there are 10 elements controlled by the rising edge of clock clk1, then chain3 and chain4 will get 5 elements each.

```
prompt> set_scan_path chain1 -view spec -scan_master_clock clk -edge rising
prompt> set_scan_path chain2 -view spec -scan_master_clock clk -edge falling

prompt> set_scan_path chain3 -view spec -scan_master_clock clk1 -edge rising
prompt> set_scan_path chain4 -view spec -scan_master_clock clk1 -edge rising
```

The following example shows how to specify the DFT signals and scan path for a design with existing scan chains:

```
prompt> set_dft_signal -view existing_dft \
        -type ScanDataIn -port test_si
prompt> set_dft_signal -view existing_dft \
        -type ScanDataOut -port test_so
prompt> set_dft_signal -view existing_dft \
        -type ScanEnable -port test_se
prompt> set_dft_signal -view existing_dft \
```

```

        -type MasterClock -port test_clk -timing {45 55}
prompt> set_dft_signal -view existing_dft \
        -type ScanMasterClock -port test_clk -timing {45 55}
prompt> set_scan_path chain0 -view existing_dft \
        -scan_data_in test_si -scan_data_out test_so -scan_enable test_se

```

The following example shows how to specify a scan path such that the first cell of the scan chain is connected to the output of head_reg_2 and the last cell of the scan chain is connected to the input of tail_reg.

```

prompt> set_scan_path 1 -view spec \
        -pipeline_head_registers {head_reg_1, head_reg_2} \
        -pipeline_tail_registers {tail_reg}

```

The following example shows how to specify a scan path for a design with existing scan chains, using the **set_scan_path -infer_dft_signal** option to infer the existing DFT signals:

```

prompt> set_scan_path chain0 -view existing_dft -infer_dft_signals

```

The following example shows how to specify bsd path for use with bsd synthesis. Here in1, in2, en, out1, out2 are the ports of the current design.

```

prompt> set_scan_path -class bsd boundary -view spec \
        -ordered_elements {in1 in2 en out1 out2}

```

The following example shows how to specify a short BSD chain for use with BSD synthesis. In this example in1, in2 are the ports of the current design.

```

prompt> set_scan_path -class bsd short_bsd_chain1 -view spec \
        -ordered_elements {in1 in2}

```

The following example shows how to specify a BSD register for use with BSD synthesis. Here I1/si, I1/so, I1/se, I1/clk, I1/inst_en are active high access pins of the BSD register to be hooked up to BSD logic during BSD synthesis. Pin I1/reset is an active low access pin of the bsd register.

```

prompt> set_dft_signal -type tdi -view spec -hookup_pin I1/si
prompt> set_dft_signal -type tdo -view spec -hookup_pin I1/so
prompt> set_dft_signal -type bsd_shift_en -view spec -hookup_pin I1/se
prompt> set_dft_signal -type capture_clk -view spec -hookup_pin I1/clk
prompt> set_dft_signal -type inst_enable -view spec -hookup_pin I1/inst_en
prompt> set_dft_signal -type bsd_reset -view spec -hookup_pin I1/reset \
        -active_state 0
prompt> set_scan_path -class bsd MY_USER_REG -view spec -exact_length 10 \
        -hookup {I1/si I1/so I1/se I1/clk I1/inst_en I1/reset}

```

SEE ALSO

current_design(2)

```
define_test_mode(2)
dft_drc(2)
insert_dft(2)
preview_dft(2)
remove_scan_path(2)
report_scan_path(2)
set_dft_signal(2)
set_dont_touch(2)
set_scan_configuration(2)
set_scan_element(2)
set_scan_link(2)
write(2)
test_default_scan_style(3)
```

set_scan_register_type

Specifies a list of scan sequential cells from the target library that are to be used by **insert_scan** or **compile -scan** when scan replacing designs or cell instances.

SYNTAX

```
int set_scan_register_type  
[-exact]  
[-type example_scan_seq_cell_list]  
[cell_or_design_list]
```

Data Types

<i>example_scan_seq_cell_list</i>	list
<i>cell_or_design_list</i>	list

ARGUMENTS

-exact

If specified, indicates that the *example_scan_seq_cell_list* is to be honored even during backend delay and area optimization done by **insert_scan** or by subsequent synthesis operations (for example, **compile -incremental**). If **-exact** is not specified, backend optimization can optimize scan cell instances to use scan cells that are not in the *example_scan_seq_cell_list*.

-type *example_scan_seq_cell_list*

Specifies a list of scan sequential cells from the target library, to be used by **insert_scan** or **compile -scan** as the scan equivalent for the current design or for sequential cells in the *cell_or_design_list* during scan replacement.

cell_or_design_list

Specifies a list of sequential nonscan cells or designs that are to be replaced by the scan cells in *example_scan_seq_cell_list* during scan replacement. The default is the current design.

DESCRIPTION

The **set_scan_register_type** command specifies a list of scan sequential cells from the target library to be used by **insert_scan** or **compile -scan** as the scan equivalents for specified nonscan sequential cells during scan replacement.

To determine the current settings resulting from previous use of the **set_scan_register_type** command, execute **report_test -register**. To remove **set_scan_register_type** settings currently in effect, execute **remove_scan_register_type**.

Specifying **-exact** indicates that only cells from the **-type** list can be used as a scan replacements for those cells, even during scan optimization.

To effectively use the **set_scan_register_type** command, it is important to understand the scan insertion process. The process of mapping sequential cells into scan flip-

flops and latches consists of three steps:

1. The **compile -scan** command maps each sequential cell in the generic design description into an initial nonscan latch or flip-flop from the target library. Non-scan sequential cells are mapped to an initial latch or flip-flop from the target library. In the absence of any **set_scan_register_type** specification, the smallest area-cost flip-flop or latch is chosen. For a design or cell instance that has a **set_scan_register_type** setting in effect, the nonscan equivalent of a scan cell in the *example_scan_seq_cell_list* is chosen.
2. The **compile -scan** or **insert_scan** command replaces the nonscan sequential cells with scan sequential cells, using only the scan cells specified by the **set_scan_register_type** command, where applicable. If **compile -scan** or **insert_scan** is unable to use a scan cell from the *example_scan_seq_cell_list*, it uses the best possible matching scan cell from the target library and issues a warning.
3. If non mapping was possible with **fun-id**. The scan replacement table specified using **set_scan_replacement** is searched for a one-to-one mapping for the design's scan style.
4. If the **-exact** option is not used in the **set_scan_register_type** command, **compile -scan** or **insert_scan** attempts to remap each scan sequential cell into another component from the target library to optimize the delay or area characteristics of the circuit. If the **-exact** option is used, optimization is restricted to using the scan cells in the *example_scan_seq_cell_list*.

EXAMPLES

In the following example, the scan register type for the current design is to be SDFLOP in the target library. (The DFLOP is a non-scan equivalent of this SDFLOP.) During scan replacement, the scan sequential cell used is SDFLOP, but during scan cell optimization, other scan cells from the target library could be used.

```
prompt> set_scan_register_type -type SDFLOP
```

In the following example, the scan register type for the current design is to be SDFLOP, and this cell is to be used as an exact scan replacement. During scan replacement, the scan sequential cell used is SDFLOP, and remains so during scan cell optimization.

```
prompt> set_scan_register_type -exact -type SDFLOP
```

The following example indicates that the scan register type for the current design is to be from the list of cells SD1FLOP SD2FLOP. During scan replacement, the scan sequential cell used is either an SD1FLOP or an SD2FLOP.

```
prompt> set_scan_register_type -type {SD1FLOP SD2FLOP}
```

SEE ALSO

```
compile(2)  
current_design(2)  
insert_dft(2)  
set_scan_replacement(2)
```

```
remove_scan_register_type(2)
reset_design(2)
target_library(3)
```

set_scan_replacement

Specifies a table of one-to-one mappings of flip-flops to their equivalent scan flip-flops from the target library that are to be used by **insert_dft** when scan replacing cell instances.

Note that as of 2006.06-SP1, **compile -scan** and **compile_ultra -scan** map to scan cells directly, rather than doing scan replacement. So, this command has no effect on **compile -scan** and **compile_ultra -scan**.

SYNTAX

```
status set_scan_replacement
[-nonscan non_scan_seq_cell_list]
[-lssd lssd_rep_cell]
[-multiplexed_flip_flop muxed_scan_cell]
[-clocked_scan clocked_scan_cell]
[-aux_clock_lssd aux_clock_lssd]
[-combinational combinational_scan_cell]
```

Data Types

<i>non_scan_seq_cell_list</i>	list
<i>lssd_rep_cell</i>	string
<i>muxed_scan_cell</i>	string
<i>clocked_scan_cell</i>	string
<i>aux_clock_lssd</i>	string
<i>combinational_scan_cell</i>	string

ARGUMENTS

-nonscan *non_scan_seq_cell_list*
Specifies the set of cells for which a scan replacement is specified using the **set_scan_replacement** command.

-lssd *lssd_rep_cell*
Specifies the scan replacement flip-flop for all flip-flops specified in the **-nonscan** option in the LSSD scan methodology.

-multiplexed_flip_flop *muxed_scan_cell*
Specifies the scan replacement flip-flop for all flip-flops specified in the **-nonscan** option in the multiplexed flip-flops scan methodology.

-clocked_scan *clocked_scan_cell*
Specifies the scan replacement flip-flop for all flip-flops specified in the **-nonscan** option in the clocked scan methodology.

-aux_clock_lssd *aux_clock_lssd*
Specifies the scan replacement flip-flop for all flip-flops specified in the **-nonscan** option in the corresponding methodology.

-combinational *combinational_scan_cell*
Specifies the scan replacement flip-flop for all flip-flops specified in the

-nonscan option in the corresponding scan methodology.

DESCRIPTION

The **set_scan_replacement** command indicates that the instances of cells in *non_scan_seq_cell_list* for the **-nonscan** option will be replaced by the corresponding scan cell specified in each of the scan style options.

The flip-flops specified in all options except **-nonscan** must be valid scan cells for the corresponding scan methodology.

The **set_scan_replacement** command can be used to specify a one-to-one mapping of nonscan flip-flops to corresponding scan flip-flops for each scan methodology.

The **insert_dft** command uses this mapping table to do scan replacement. If the mapping is not a valid scan cell for the corresponding scan methodology, **insert_dft** will reject the mapping and use other mapping methods, such as function ID or Boolean matching. Boolean matching can be time consuming, but ensures that a scan flip-flop that satisfies the design constraints is selected.

This mapping does not ensure that the design constraints are met before mapping. This command should be used only when you are sure that your constraints will be met by this mapping.

EXAMPLES

In the following example, all instances of FD1 will be replaced by the scan flip-flop FD1S in the multiplexed scan style:

```
prompt> set_scan_replacement -nonscan FD1 -multiplexed_flop_flop FD1S
```

In the following example, the instances of FD1, FD2 and FD3 will be replaced by FDLS in the LSSD scan methodology:

```
prompt> set_scan_replacement -nonscan {FD1 FD2 FD3} -lssd FDLS
```

SEE ALSO

```
current_design(2)
insert_dft(2)
remove_scan_register_type(2)
remove_scan_replacement(2)
report_scan_replacement(2)
reset_design(2)
translate(2)
target_library(3)
```


set_scan_state

Sets the scan state status for the current db design.

SYNTAX

```
int set_scan_state  
unknown | test_ready | scan_existing
```

ARGUMENTS

DESCRIPTION

This command sets the scan state status for the current design. Valid values are **unknown** (the scan state of the design is unknown), **test_ready** (the design is scan-replaced), and **scan_existing** (the design is scan-inserted).

Use this command only on a design that has been scan-replaced using, for example, **compile -scan**, so that the Q outputs of scan flip-flops are connected to the scan inputs and the scan enable pins are connected to logic zero. If there are nonscan elements in the design, use **set_scan_element false** to identify them.

EXAMPLES

The following example illustrates setting the test_ready scan state status for the current design.

```
prompt> set_scan_state test_ready
```

SEE ALSO

```
current_design(2)  
insert_dft(2)  
reset_design(2)  
set_scan_configuration(2)  
set_scan_element(2)  
set_scan_path(2)
```

set_scan_suppress_toggling

Enables and controls the gating by the **insert_dft** command of functional output pins of scan flip-flops in order to suppress the toggling activity on the functional logic during scan shift.

SYNTAX

```
status set_scan_suppress_toggling  
-selection_method manual | auto | mixed  
-include_elements cell_design_ref_list  
-exclude_elements cell_design_ref_list  
-min_slack minimum_timing_slack_after_gating  
-ignore_timing_impact true_or_false  
-total_percentage_gating percentage_value
```

Data Types

<i>cell_design_ref_list</i>	list
<i>minimum_timing_slack_after_gating</i>	float

ARGUMENTS

-selection_method manual | auto | mixed
This option specifies the scan flip-flop selection method for the **insert_dft** command. It takes one of three possible values: manual, auto or mixed. The "manual" value is the default. When this value is used, the **set_scan_suppress_toggling** command expects an explicit list of designs and/or cells to be supplied through the **-include_elements** option. The value "auto" is used to enable the automatic selection of scan flip-flops for gating. Finally, the value could be set to "mixed". In this case, it will enable the combined approach of supplementing a manual specification with the automatic selection.

-include_elements *cell_design_ref_list*
Specifies a list of valid scan flip-flop instances or design objects containing valid scan flip-flops whose functional output pins are to be gated by the **insert_dft** command. The design objects must be of the following types:

- scan flip-flop instances
- hierarchical cells (containing flip-flops)
- references of flip-flops
- designs

-exclude_elements *cell_design_ref_list*
Specifies a list of valid scan flip-flop instances or design objects containing valid scan flip-flops whose functional output pins are not to be gated by the **insert_dft** command. The type of design objects is the same as with the **-include_elements** option.

`-min_slack minimum_timing_slack_after_gating`

Specifies a value between 0 and any positive number up to 1000. This value is the minimum timing slack after scan cell output gating is completed. It helps the **insert_dft** command avoid adding gates on timing critical paths. This option is not to be used when the `-selection_method` is set to manual.

`-ignore_timing_impact true_or_false`

This Boolean option is set to false by default. It could be set to true when the timing impact of the scan cell output gating is to be ignored. This option is not to be used when the `-selection_method` is set to manual.

`-total_percentage_gating percentage_value`

Specifies a value between 0.001 and any positive number up to 100. This indicates the percentage of scan flip-flops to be gated in a design. The option value defaults to 5. This option is not to be used when the `-selection_method` is set to manual.

DESCRIPTION

During scan testing, while scan data shifts through scan chains, the logic driven by the scan flip-flop functional output pins are also toggling. This can cause increased power dissipation. The **insert_dft** command has an option for you to gate the functional output pins of valid scan cells in order to eliminate the toggling activity on the functional logic during scan shift. This is enabled through the **set_scan_suppress_toggling** command. You can opt for a manual gating by supplying your list of design objects. In this case, the **insert_dft** command will add AND-gates at the functional outputs of your valid scan flip-flops. Alternatively, you can let the **insert_dft** command automatically select the best valid scan flip-flops for gating. In this case, you could direct the tool by providing information on timing slack, total percentage of gating. Otherwise, the tool will use the defaults as described above. Finally, you could use a combined approach where you could specify your list of scan flip-flops, and the tool will figure out the rest of valid scan flip-flops until it reaches the total percentage gating number.

EXAMPLES

The following example specifies a list of scan flip-flop instances for the **insert_dft** command. This is the case of discrete power gating of the functional output pins.

```
prompt> set_scan_suppress_toggling -selection_method manual -  
include_elements {U3_*} -exclude_elements [list U3_1 U3_2]
```

The following example specifies that the **insert_dft** command is to insert power gating on the functional output pins of every instance of the DFF cell in the current design:

```
prompt> set_scan_suppress_toggling -include_elements [list [get_references DFF]]
```

The following example specifies that the **insert_dft** command is to insert power gating on the functional output pins of all valid scan flip-flop instances within

the instance of the REGFILE design:

```
prompt> set_scan_suppress_toggling -include_elements [list [get_designs REGFILE]]
```

The following example specifies that the **insert_dft** command is to insert power gating on the functional output pins valid scan flip-flop it automatically selects within 10% of total percentage gating:

```
prompt> set_scan_suppress_toggling -selection_method auto -  
total_percentage_gating 10
```

The following example specifies that the **insert_dft** command is to insert power gating on the functional output pins valid scan flip-flop it automatically selects within 10% of total percentage gating, and not to violate min_slack of 1ns:

```
prompt> set_scan_suppress_toggling -selection_method auto -  
total_percentage_gating 10 -min_slack 1
```

The following example shows a usage of mixed selection method:

```
prompt> set_scan_suppress_toggling -selection_method mixed -min_slack 1 -  
include_elements [list U3_1 U3_2]
```

SEE ALSO

```
insert_dft(2)  
remove_scan_suppress_toggling(2)  
report_scan_suppress_toggling(2)
```

set_scenario_options

Sets the scenario options for one or more scenarios.

SYNTAX

```
status set_scenario_options  
[-scenarios scenario_list]  
[-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.

-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. All scenario options can be applied to active and inactive 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): 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
  No operating condition was set in scenario MODE1
```

```
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 MODE1 only for hold timing 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 -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: MODE1 is active.

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_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_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

```
compile(2)
get_attribute(2)
report_attribute(2)
report_cell(2)
set_compile_directives(2)
```

set_structure

Sets structure attributes on a design or on a list of designs, to determine how the designs are structured during compile.

SYNTAX

```
status set_structure
[true | false]
[-design design_list]
[-boolean true | false]
[-boolean_effort low | medium | high]
[-timing true | false]
```

Data Types

design_list list

ARGUMENTS

true | false

Specifies the value to which the **structure** attribute is to be set. If set to **true** (the default), **compile** adds logic structure to the specified designs by adding intermediate variables factored from the designs' equations.

-design *design_list*

Specifies a list of designs on which the **structure** attribute is to be set. The default value is the current design.

-boolean true | false

Specifies the value to which the **structure_boolean** attribute is to be set. If set to **true**, and if the **structure** attribute is also set to **true**, **compile** uses Boolean (non-algebraic) optimization on the specified designs. The default value is **false**. If **structure** is **false**, **structure_boolean** is ignored.

-boolean_effort low | medium | high

Specifies the value to which the **boolean_effort** attribute is to be set. This attribute specifies the relative amount of CPU time to be spent by Boolean (non-algebraic) optimization during the structuring phase of **compile**. The default value is **low**.

-timing true | false

Specifies the value to which the **structure_timing** attribute is to be set. If **structure** is set to **true**, by default **structure_timing** is also **true**, and **compile** uses timing-driven structuring on the specified designs. Set this attribute to **false** if you do not want timing-driven structuring. If **structure** is **false**, **structure_timing** is ignored.

DESCRIPTION

The **set_structure** command sets structure attributes on a design or on a list of designs, to determine how the designs are structured during **compile**. If

set_structure is called without a value, by default the design is structured.

This optimization step adds logic structure to a design by adding intermediate variables. The **compile** command searches for subfunctions that can be factored out and evaluates these factors based on the size of the factor and the number of times the factor appears in the design. The subfunctions that most reduce the logic are converted to intermediate variables and factored out of the design's equations. When **structure** is **true**, by default **structure_timing** is also **true**, and timing-driving structuring is automatically done unless **structure_timing** is set to **false**. With **structure** set to **true**, setting **structure_boolean** to **true** enables Boolean (non-algebraic) optimization during structuring. If **structure** is **false**, both **structure_boolean** and **structure_timing** are ignored.

To remove the **structure**, **structure_boolean**, **boolean_effort**, and **structure_timing** attributes, use the **remove_attribute** command. To remove all attributes, use the **reset_design** command.

EXAMPLES

In the following example, structuring, timing-driven structuring, and Boolean optimization are enabled on the design named TEST1 and structuring is disabled on the design named TEST2:

```
prompt> set_structure -design TEST1 -boolean
```

```
prompt> set_structure -design TEST2 false
```

In the following example, structuring is enabled and timing-driven structuring is disabled on the design named ADD1:

```
prompt> set_structure -design ADD1 -timing false
```

In the following example, structuring is disabled for all designs read in memory:

```
prompt> set_structure -design [get_designs *] false
```

SEE ALSO

```
compile(2)  
remove_attribute(2)  
reset_design(2)  
set_flatten(2)
```

set_svf

Generates a Formality setup information file for efficient compare point matching in Formality.

SYNTAX

```
Boolean set_svf  
filename  
[-append]  
[-off]
```

ARGUMENTS

filename

Names the file into which Formality setup information will be recorded. You must specify a file name unless the -off option is specified.

-append

Appends to the specified file. If another Formality setup verification file is already open then it will be closed before opening the specified file. If -append is not used then **set_svf** will overwrite the named file, if it exists.

-off

Stops recording Formality setup information to the currently-open file. To resume recording into the same file, you must re-issue the **set_svf** command with the -append option.

DESCRIPTION

This command causes Design Compiler to start recording setup information for Formality, the Synopsys formal verification tool. The SVF ("Setup Verification for Formality") file that is produced is used by Formality during the matching step to facilitate the alignment of compare points. By using the automatically-generated SVF file, the user is relieved of the time-consuming and error-prone process of entering the setup information manually.

To record setup information for formal verification products other than Formality, use the *set_vsdc* command, instead.

The SVF file is used to account for name changes that occur during synthesis as a result of running commands such as *group*, *ungroup*, *uniquify* and *change_names*. Also, certain operations performed by the *compile* command perform transformations that are recorded in the SVF file.

Further information on SVF files can be found in the Formality User Guide.

Once this command is issued, Design Compiler will begin recording all relevant operations. Because information is buffered internally, the file may not be complete until recording is stopped. Recording is stopped by running "set_svf -off" or by exiting *dc_shell*. Running **set_svf** with a new file name will write out and close the original SVF file before starting the new one.

The SVF file is stored in encrypted format.

This command returns a Boolean value indicating whether **set_svf** succeeded (true) or not (false).

Formality setup recording might be enabled by default in Design Compiler's system-wide setup file. If so, then the file name used is "default.svf". If you prefer a different file name then you may run **set_svf** in your setup file or script prior to performing any recordable operations. Alternately, you may disable SVF recording completely by running "set_svf -off".

EXAMPLES

The following example shows a typical invocation of the command

```
prompt> set_svf my_design.svf
```

In the following example the Formality setup information is appended to the existing SVF file, barfile.svf.

```
prompt> set_svf -append barfile.svf
```

SEE ALSO

```
change_names(2)  
compile(2)  
compile_ultra(2)  
group(2)  
set_vsd(2)  
ungroup(2)  
uniquify(2)  
compile_seqmap_propagate_constants(3)  
fsm_auto_inferring(3)
```

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

set_switching_activity

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)  
synlib_dont_get_license(3)
```

set_tap_elements

Specifies the set of TAP state elements for the boundary-scan TAP controller.

SYNTAX

```
int set_tap_elements  
-state_cells cell_names
```

Data Types

cell_names list

ARGUMENTS

```
-state_cells cell_names  
             Specifies the set of TAP state elements cell_names.
```

DESCRIPTION

The **set_tap_elements** command identifies the set of TAP state elements in the boundary-scan TAP controller. The state elements can be specified in any order. A minimum of four state elements are required according to the IEEE 1149.1 standard.

EXAMPLES

The following example specifies four TAP state elements.

```
prompt> set_tap_elements -state_cells {                    tap/simple/  
      tap_state_reg[1],  
          tap/simple/tap_state_reg[3],  
          tap/simple/tap_state_reg[0],  
          tap/simple/tap_state_reg[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

set_target_library_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_test_assume

Sets the **test_assume** attribute to a logic value to be assumed on specified cell output pins throughout test design rule checking.

SYNTAX

```
integer set_test_assume  
zero_or_one_value  
pin_list
```

Data Types

<i>zero_or_one_value</i>	string
<i>pin_list</i>	list

ARGUMENTS

zero_or_one_value
Specifies the fixed value of the output pins. To assume a constant value of logic one, *zero_or_one_value* must be "1", "one", or "ONE." To assume a constant value of logic zero, *zero_or_one_value* must be "0", "zero", or "ZERO."

pin_list
Lists the hierarchical pins in the current design for which values are specified. If more than one pin is specified, they must be enclosed in braces ({ }).

DESCRIPTION

The **set_test_assume** command sets the **test_assume** attribute to *zero_or_one_value* on the cell output pins on *pin_list*. The **dft_drc** command uses **test_assume** to freeze the assumed value on the specified pins throughout test design rule checking.

Use **set_test_assume** to set conditions that would otherwise be unknown to **dft_drc**, such as the state of a non-scan register, or the output of a RAM. Known values are often applied to the design circuitry to facilitate testing by configuring the design circuitry into the desired test mode. For example, during testing, a state could be preloaded into the non-scan registers of an on-chip test controller. This condition could configure the scan chains in the design to form multiple parallel scan chains. Or, if it is known that prior to testing, all address locations of an on-chip RAM are preloaded with a fixed pattern, this pattern can be assumed at the data output pins of the RAM when testing the surrounding functional logic. You can communicate these preloaded values to the testing software through the **test_assume** attribute.

Note: The specified conditions must exist throughout testing, regardless of other changes in the design state. Otherwise, incorrect design rule checking reports can result. If the assumed state is not attainable or not held throughout the scan test sequence, it is considered best practice to use the initialization protocol to provide initialization vectors that perform the assumed setup, instead of using **set_test_assume**, which could result in errors. The **read_test_protocol -section**

test_setup is used to read in this initialization protocol.

Use the **report_test_assume** command to list the **test_assume** attributes on a design.

To remove selected or all **test_assume** attributes from a design, use the **remove_attribute** or **reset_design** command.

EXAMPLES

The following commands specify a fixed internal state to assume for a register in an on-chip test controller:

```
prompt> set_test_assume 0 my_tap/reg3/Q
```

```
prompt> set_test_assume 1 my_tap/reg3/QN
```

The following command specifies that all the data outputs of a RAM cell are fixed throughout testing:

```
prompt> set_test_assume ONE [get_pins h1/ramcell/DOUT*]
```

SEE ALSO

```
current_design(2)
dft_drc(2)
get_attribute(2)
remove_attribute(2)
report_test_assume(2)
reset_design(2)
set_dft_signal(2)
```

set_test_point_element

Controls the configuration of the user-defined control, force, and observe test points.

SYNTAX

```
int set_test_point_element  
list_of_design_pin_objects  
[-  
type control_0 | control_z0 | control_1 | control_z1 | control_01 | control_z01 | f  
orce_0 | force_z0 | force_1 | force_z1 | force_01 | force_z01 | observe]  
[-control_signal control_name]  
[-clock_signal clock_name]  
[-source_or_sink source_or_sink_name]  
[-test_point_enable test_point_enable_name]  
[-test_points_per_source_or_sink n]  
[-test_points_per_test_point_enable n]  
[-scan_source_or_sink enable | disable]  
[-scan_test_point_enable enable | disable]  
[-power_saving enable | disable]
```

Data Types

<i>list_of_design_pin_objects</i>	list
<i>control_name</i>	string
<i>clock_name</i>	string
<i>source_or_sink_name</i>	string
<i>test_point_enable_name</i>	string
<i>n</i>	integer

ARGUMENTS

list_of_design_pin_objects
Specifies the list of design pin objects to which the user-defined test point element specification applies. The pin list is a required argument.

-type control_0 | control_z0 | control_1 | control_z1 | control_01 | control_z01 | force_0 | force_z0 | force_1 | force_z1 | force_01 | force_z01 | observe

| control_z1 | control_01 | control_z01 | 0force_0 | force_z0
| force_1 | force_z1 | force_01 | force_z01 | observe

Specifies the type of test point to insert on each pin object in the list. Specify only one type with each invocation of this command. Observe and Control test points are not supported with LSSD scan style.

-control_signal *control_name*
Specifies the name of the control signal (existing pin or port name) for the test points. Predefine the signal as having either TestMode or ScanEnable signal type. If no control signal is specified, the test points are controlled by any signal with the TestMode signal type. If no TestMode signal is found,

DFT Compiler creates a new TestMode port.

-clock_signal *clock_name*

Specifies the name of the clock signal (existing pin or port name) for the observe, control, and test point enable scan registers. Predefine the signal as having a **clock** signal type. If no clock signal is specified, DFT Compiler creates a new clock port.

-source_or_sink *source_or_sink_name*

Specifies the name of the source signal (for control and force test points) or the sink signal (for observe test points). The source or sink signal can be an existing pin or an existing port. If no source or sink signal is specified, DFT Compiler either creates a new clock port when **-scan_source_or_sink** is disabled, or inserts a control (for control and force test points) or an observe (for observe test points) scan register when **-scan_source_or_sink** is enabled.

-test_point_enable *test_point_enable_name*

Specifies the name of the test point enable signal (only for control test points). The test_point_enable signal can be an existing pin or an existing port. If no test_point_enable signal is specified, DFT Compiler either creates a new test_point_enable port when **-scan_test_point_enable** is disabled, or inserts a test_point_enable scan register when **-scan_test_point_enable** is enabled.

This option is valid only for the control test point types.

-test_points_per_source_or_sink *n*

Specifies the number of test points that can be shared per source signal (for control and force test points) or sink signal (for observe test points). This source signal can be a control scan register or a source input port. Similarly, by default, 8 observe test points can share one sink signal. This sink signal can be an observe scan cell or a sink output port. By default, 8 control or force test points can share one source signal.

-test_points_per_test_point_enable *n*

Specifies the number of test points that can be shared per test point enable signal for control test points. By default, a test point enable signal is used per control test point. This test point signal can be a test point enable scan register or a test point enable input port.

This option is valid only for the control test point types.

-scan_source_or_sink enable | disable

Enables or disables the insertion of control or observe scan registers. When **-scan_source_or_sink** is enabled and no source_or_sink signal is specified, the test data of the control and force test points come from new control scan registers. In this case, data are observed through observe scan registers. When **-scan_source_or_sink** is disabled and no source_or_sink signal is specified, the test data of the control and force test points come from a new source signal (new primary input). In this case, data are observed through a new sink signal (new output port). By default, **scan_source_or_sink** is enabled.

-scan_test_point_enable enable | disable

Enables or disables the insertion of test point enable scan registers. When **-scan_test_point_enable** is enabled and no test_point_enable signal is

specified, the test point enable of the control test points come from new test point enable scan registers. When **-scan_test_point_enable** is disabled and no test_point_enable signal is specified, the test point enable of the control test points come from a new test point enable signal (new input port). By default, scan_test_point_enable is enabled.
This option is valid only for the control test point types.

-power_saving enable | disable

Specifies the insertion of power saving logic structures (AND gates) on the observe XOR trees.

DESCRIPTION

The **set_test_point_element** command specifies the type of test point to insert and the locations in the design that need a test point of this type.

This command specifies a list of pins that require a user-defined test point. The user-defined test point feature fixes scan rule violations (force test points), to reduce test data volume reduction (observe test points) and to improve test coverage (control and observe test points).

EXAMPLES

The following example specifies that the **preview_dft** and **insert_dft** commands consider the following pins for the observe test point insertion. Since **-scan_source_or_sink** is enabled by default, **insert_dft** inserts 2 new observe scan registers to observe the 4 pins. The first 3 pins share one observe scan register and the last pin is observed by the second observe scan register.

```
prompt> set_test_point_element {U0/Q U1/Q U2/Q U3/Q} \  
      -type observe -test_points_per_source_or_sink 3
```

SEE ALSO

```
insert_dft(2)  
preview_dft(2)  
remove_test_point_element(2)  
report_test_point_element(2)
```

set_testability_configuration

Controls configuration of automatically inserted control and observe points. The control and observe test point candidates are identified based on testability analysis.

SYNTAX

```
int set_testability_configuration  
[-type control | observe | control_and_observe]  
[-control_signal control_name]  
[-clock_signal clock_name]  
[-clock_type dominant | dedicated]  
[-max_test_points n]  
[-test_points_per_scan_cell n]  
[-power_saving enable | disable]  
[-max_additional_logic_area n]
```

Data Types

<i>control_name</i>	string
<i>clock_name</i>	string
<i>n</i>	integer

ARGUMENTS

- type control | observe | control_and_observe
Specifies the **control**, **observe**, or **control_and_observe** type (argument required). The **observe** type reduces the number of patterns to apply to the design (Test Data Volume Reduction) by inserting only observe test points. The **control** and **control_and_observe** types focus on test coverage improvement. The optimum test coverage improvement occurs when both control and observe points are inserted in the design.
- control_signal *control_name*
Specifies the name of the control signal for test points. The signal must be predefined as having the TestMode signal type. If the control signal is not specified, the test points are controlled by any signal with the TestMode signal type. If a TestMode signal is not found, DFT Compiler creates a new TestMode port.
- clock_signal *clock_name*
Specifies the name of the clock signal for control and observe scan registers. If a specific port is to be used for the data clock, the signal must be predefined with set_dft_signal -type ScanClock. If a clock signal is not specified, DFT Compiler creates a new clock port.
- clock_type dominant | dedicated
Specifies the clock type to use for control and observe point scan cells. By default, the clock type is **dominant**.
- max_test_points *n*
Specifies the maximum number of test points inserted. The default is 1000.

When the type is **control_and_observe**, control points and observe points are inserted up to the maximum number for each type.

-test_points_per_scan_cell *n*

Specifies the number of test points of the same type (control or observe test points) that can be shared per scan cell. By default, 8 control test points can share one control scan cell and 8 observe test points can share one observe scan cell.

-power_saving enable | disable

Specifies the insertion of power saving logic structures (AND gates) on the observe XOR trees.

-max_additional_logic_area *n*

Specifies the maximum percentage of area overhead due to the observe test point structure insertion in the design.

DESCRIPTION

The **set_testability_configuration** command controls the automatic test point insertion based on testability analysis. You can insert test points at either the RTL or gate level. If you insert observe test points at the RTL level, set the **compile_delete_unloaded_scan_cells** variable to **false** to avoid removing scan cells associated with observe points during compilation.

In the **control** method, only the control test points are inserted. This method might not lead to the optimum test coverage improvement.

In order to get the optimum test coverage improvement, both control and observe test points can be inserted while using the **control_and_observe** method.

```
prompt> set_dft_signal test_point_clock -port port_name
```

EXAMPLES

The following example specifies that a maximum of 1000 control points and 1000 observe points must be inserted. The clock for the control and observe registers is a dedicated test clock.

```
prompt> set_testability_configuration -type control_and_observe \  
      -max_test_points 1000 -clock_type dedicated
```

SEE ALSO

```
report_testability_configuration(2)  
reset_testability_configuration(2)  
set_dft_configuration(2)
```


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`

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
```

```
set_timing_ranges  
1800
```

```

        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

`compile(2)`
`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. This command is supported only in topographical mode.

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.

-techtf_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:

```
set_tlu_plus_files
1804
```



```
conducting_layers
METAL1 metall
...
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 = "metall"
...
}
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_transform_for_retiming

Sets the **transform_for_retiming** attribute on cells in the current design. This can effect both hierarchical cells and sequential leaf cells.

SYNTAX

```
integer set_transform_for_retiming  
cell_list  
multiclass | decompose | dont_retime
```

Data Types

cell_list list

ARGUMENTS

cell_list

Specifies a list of cells on which the **transform_for_retiming** attribute is to be set. If you specify more than one cell name, the names must be enclosed either in quotation marks or in braces ({}).

multiclass | decompose | dont_retime

Specifies the value with which to set the **transform_for_retiming** attribute. There is no default value. One of the values must be specified.

- **multiclass** specifies that the cell will be moved together with any synchronous or asynchronous preset or clear or synchronous load-enable signals.
- **decompose** specifies that synchronous load-enable and synchronous reset logic are made explicit, and only the basic storage register is moved. For example, if a register has a data (D) input, a clock (CLK) input, and a synchronous clear (SD) input with active low polarity and the **transform_for_retiming** attribute is set to *decompose*, it is decomposed into a simple register with D and CLK input and an and gate driving the D input. The inputs of this and gate are the original data net and the synchronous clear net. Only the simple register will be moved for retiming, while the and gate stays in place.
- **dont_retime** specifies that the cell will not be retimed. It may still be mapped to a different library cell during incremental mapping optimizations.

DESCRIPTION

This command sets the **transform_for_retiming** attribute on cells in the current design.

If placed on a sequential, non-hierarchical cell, the attribute determines the way in which this cell will be transformed for retiming.

If you want to disable both retiming and sequential mapping optimizations for a

cell, use the **dont_touch** attribute.

If the attribute is set on a hierarchical cell, it applies to all sequential cells in the hierarchy below this cell, unless the attribute is set on a hierarchical cell in between or on the sequential leaf cell itself. Values of the attribute set on lower levels of hierarchy override those set on higher levels.

The attribute does not effect non-sequential non-hierarchical cells.

If the attribute has neither been set on a sequential leaf cell nor on any of its hierarchical parent cells, the transform that is applied to this cell is determined by the corresponding command line option chosen for the **optimize_registers** command, or by the default setting for the retiming command that you are using.

If the **dont_touch** attribute is set to *true* on a cell, the **transform_for_retiming** attribute does not come into effect on this cell or any of its child cells.

Using the attribute can improve timing results at the cost of increased area for certain designs.

To remove the **transform_for_retiming** attribute, use **remove_attribute**.

EXAMPLES

The data inputs of two synchronous clear register cells R_1 and R_2 are connected to the same net, while the synchronous clear inputs are connected to two different and unrelated clear signals. If **optimize_registers** is invoked with the command line option **-sync_trans multiclass**, it will not move any of these registers backward over the driving cell of the net connected to the data ports, even if it would improve the timing of the design. Setting the **transform_for_retiming** attribute to *decompose* on the register cells allows backward movement to achieve better timing results. However, there are additional gates in the design. Use the following command to achieve this:

```
prompt> set_transform_for_retiming find(cell, {R_1, R_2}) decompose
```

The following command removes the attribute from the two cells:

```
prompt> remove_attribute find(cell, {R_1, R_2}) transform_for_retiming
```

SEE ALSO

```
balance_registers(2)
optimize_registers(2)
set_optimize_registers(2)
set_dont_touch(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  
1810
```

```
reset_design(2)
set_logic_one(2)
set_logic_zero(2)
set_logic_dc(2)
simplify_constants(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

`compile(2)`
`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  
1814
```

-power uW -voltage V -current A <plain

SEE ALSO

`report_units(2)`

set_user_attribute

Sets the value of an attribute on a design or library object.

SYNTAX

```
list set_user_attribute  
object_list  
attribute_name  
attribute_value  
[-bus]  
[-quiet]
```

Data Types

<i>object_list</i>	list
<i>attribute_name</i>	string
<i>attribute_value</i>	attribute value

ARGUMENTS

object_list
Specifies a list of design or library objects on which the attribute is to be set. For details on searching for design and library names, see the man pages for the **get_designs** and **get_libs** commands.

attribute_name
Specifies the name of the attribute to set.

attribute_value
Specifies the value of the attribute. The data type must be the same as the attribute data type.

-bus
Sets the attribute on the bus as a whole, and not on each individual bus member.

-quiet
Turns off warning messages that would otherwise be issued if the attribute or objects are not found.

DESCRIPTION

The **set_user_attribute** command sets the value of an attribute on an object. It is the same as the **set_attribute** command. See the **set_attribute** command man page for further information.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

SEE ALSO

`remove_user_attribute(2)`
`set_attribute(2)`

set_user_budget

Sets user budgets or budget ratios.

SYNTAX

```
string set_user_budget  
-from object_list  
-to object_list  
[-percent]  
value
```

Data Types

<i>object_list</i>	list
<i>value</i>	float

ARGUMENTS

-from *object_list*
Indicates the pins or clocks from which the budget is specified.

-to *object_list*
Indicates the pins or clocks to which the budget is specified.

-percent
Indicates the specified budget value is a percentage.

value
Sets the budget value or percentage.

DESCRIPTION

The **set_user_budget** command is used to specify user directives for budget allocation. You can specify budgets as an absolute value or as a percentage of available constraint to be allocated. These directives are honored by budget allocation. If you don't specify user budgets for a budgeted cell, budget allocation is performed automatically.

The objects specified in either the **-from** or **-to** options must be inputs pins or output pins of budgeted cells. User budgets specified for cells which are not ultimately budgeted are ignored.

EXAMPLES

The following command specifies the budget from clock clk1 to the two outputs is 5.0.

```
prompt> set_user_budget -from clk1 -to {I2/out1 I2/out2} 5.0
```

The following command specifies that the budget from the IN input to the Y of I1

```
set_user_budget  
1818
```

should be 70 percent of the path constraint.

```
prompt> set_user_budget -from I1/IN -to I1/Y -percent 70.0
```

SEE ALSO

`dc_allocate_budgets(2)`

`report_path_budget(2)`

set_utilization

Specifies placement utilization constraint for core area. This command is supported only in topographical mode.

SYNTAX

```
int set_utilization  
util_float
```

Data Types

util_float float

ARGUMENTS

util_float
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 sum of fixed and non-fixed area 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 *util_float* is a float number less than 1, for example, 0.85 for 85% utilization. The default value is 0.60.

DESCRIPTION

This command defines design level physical constraint for utilization. The constraint will be used to generate coarse floorplan during synthesis.

Use this command after loading the physical library and design, but before running synthesis.

There are also other commands that set constraints on core area, such as **set_placement_area**, **set_rectilinear_outline** and **set_aspect_ratio**. Among the four commands, **set_placement_area** has the highest priority, while **set_rectilinear_outline** has higher priority than **set_utilization** and **set_aspect_ratio**. This means if the placement area constraint is set by **set_placement_area** or rectilinear outline constraint is set by **set_rectilinear_outline** on the current design, then the utilization constraint will not be used during synthesis. Furthermore, in this case, utilization will be neither reported by the command **report_physical_constraints**, nor written out by the command **write_physical_constraints**.

EXAMPLES

The following example shows how to set the floorplan utilization to 80%.

```
prompt> set_utilization 0.8
```

SEE ALSO

```
set_placement_area(2)  
report_physical_constraints(2)  
write_physical_constraints(2)
```

set_verification_priority

Sets the **verification_priority** attribute on specified designs, cells, or references, indicating the level of optimization during `compile_ultra`.

SYNTAX

```
status set_verification_priority  
-all  
[-high]  
object_list
```

Data Types

object_list list

ARGUMENTS

`-all` Sets the attribute on all designs (equivalent to `[get_designs *]`).

`-high` Sets the verification priority level. This option maps to the low effort of the optimization effort level and it disables DesignWare ungrouping.

object_list Specifies a list of cells, references, or designs for the attribute to be set.

DESCRIPTION

This command sets the **verification_priority** attribute on the specified objects so that different levels of optimization can be applied during **compile_ultra**. The **compile** command does not ungroup cells or designs with the **verification_priority** attribute.

This attribute is removed with the **remove_verification_priority** command.

EXAMPLES

In the following example, the **verification_priority** attribute is set to **medium** on the *U1* cell:

```
prompt> set_verification_priority U1
```

If the **verification_priority** attribute is specified on a reference object, cells using that reference are not ungrouped during **compile**. In this example, all instances of ADDER in the current design are not ungrouped:

```
prompt> set_verification_priority [get_references ADDER]
```

SEE ALSO

`compile(2)`
`current_design(2)`
`remove_verification_priority(2)`

set_voltage

Applies an operating voltage on a list of power net info objects.

SYNTAX

```
int set_voltage  
max_case_voltage  
[-min min_case_value]  
-object_list list_of_power_nets
```

Data Types

<i>max_case_voltage</i>	float
<i>list_of_power_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_power_nets*
Specifies the list of power nets that will have the operating voltages as specified in this command.

DESCRIPTION

Defines the operating voltage on the power nets so that the parts of the design powered by these power nets are timed and optimized at the specified voltage. If you do not specify any operating voltage for a power net, the part of the design connected by this power net will continue to be timed and optimized based on the available operating condition settings.

If *voltage_ranges* are specified for a power net, the operating voltages of the power net must fall within one of the ranges specified in the *voltage_ranges*. Furthermore, *min_case_voltage* must fall within the same range as the *max_case_voltage*.

To see the operating voltages of power nets, use **report_power_net_info**.

The operating voltage of a power net, once defined, can only be overridden with another **set_voltage**. It can also be cleared using **reset_design**.

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 T
1
prompt> create_power_domain A \
    -object_list [get_cells shutdown_inst] \
    -power_down -power_down_ctrl [get_nets shutdown]
1
prompt> create_power_domain PDM \
    -object_list [get_cells {ibf*/ibuf_sram}] -power_down
1
prompt> create_power_net_info PNT -power
1
prompt> create_power_net_info PNA -power
1
prompt> create_power_net_info PNM -power
1
prompt> connect_power_domain T -primary_power PNT
1
prompt> connect_power_domain A -primary_power PNT \
    -internal_power PNA
1
prompt> connect_power_domain PDM -primary_power PNM \
    -backup_power PNT
1
prompt> set_voltage 0.9 -object_list {PNT PNA}
1
prompt> set_voltage 0.75 -object_list {PNM}
1
```

This example created 3 power domains, T, A, and PDM. Power domain T is powered directly by power net PNT. Power domain A is a shutdown domain with a coarse-grained power switch. The power switch is driven by power net PNT, and produces an internal power supply PNA. All cells in power domain A are powered directly by power net PNA. A power domain PDM is also defined on two macro-cells, ibf0/ibuf_sram, and ibf1/ibuf_sram. These two macros derive their primary power from PNM, and backup power from PNT.

SEE ALSO

```
create_power_net_info(2)
create_power_domain(2)
connect_power_domain(2)
connect_power_net_info(2)
report_power_net_info(2)
report_power_domain(2)
```

set_vsdc

Generates a setup information file in V-SDC format for efficient compare point matching in formal verification tools.

SYNTAX

```
Boolean set_vsdc  
filename  
[-append]  
[-off]
```

ARGUMENTS

filename

Names the file into which verification setup information will be recorded. You must specify a file name unless the *-off* option is specified.

-append

Appends to the specified file. If another verification setup file is already open then it will be closed before opening the specified file. If *-append* is not used then **set_vsdc** will overwrite the named file, if it exists.

-off

Stops recording verification setup information to the currently-open file. To resume recording into the same file, you must re-issue the **set_vsdc** command with the *-append* option.

DESCRIPTION

This command causes Design Compiler to start recording setup information for formal verification tools. (If you are using Formality, the Synopsys formal verification product, you should use the *set_svf* command, instead.) The V-SDC file that is produced may be used by some verification tools during the matching step to facilitate the alignment of compare points. By using the automatically-generated V-SDC file, the user is relieved of the time-consuming and error-prone process of entering the setup information manually.

The V-SDC file is a series of Tcl commands stored in plain-text format.

The V-SDC file is used to account for name changes that occur during synthesis as a result of running commands such as *group*, *ungroup*, *uniquify* and *change_names*. Also, certain operations performed by the *compile* command perform operations that are recorded in the V-SDC file.

Once this command is issued, Design Compiler will begin recording all relevant operations. Because information is buffered internally, the file may not be complete until recording is stopped. Recording is stopped by running "*set_vsdc -off*" or by exiting *dc_shell*. Running **set_vsdc** with a new file name will write out and close the original V-SDC file before starting the new one.

This command returns a Boolean value indicating whether **set_vsdc** succeeded (true) or

not (false).

EXAMPLES

The following example shows a typical invocation of the command

```
prompt> set_vsd my_design.vsd
```

In the following example the verification setup information is appended to the existing V-SDC file, barfile.vsd.

```
prompt> set_vsd -append barfile.vsd
```

SEE ALSO

```
set_svf(2)  
ungroup(2)  
group(2)  
uniquify(2)  
change_names(2)  
compile(2)  
compile_ultra(2)  
fsm_auto_inferring(3)  
compile_seqmap_propagate_constants(3)
```

set_wire_load_min_block_size

Sets the wire load **min_block_size** attribute on the current design.

SYNTAX

```
int set_wire_load_min_block_size  
size
```

Data Types

size float

ARGUMENTS

size

A positive value in the units of the technology library, that specifies the minimum block area of the design and all its subdesigns, to use when selecting the appropriate wire load.

DESCRIPTION

You can use this command only if the **wire_load_model_mode** attribute has been set to *enclosed* using the **set_wire_load_mode** command.

Sets the wire load minimum block area for the current design and all its subdesigns. By default, the cell area of the design is assumed to be at least as large as the value specified for this option when selecting the appropriate wire load to use in **set_wire_load_model** command. For designs with an area smaller than the minimum specified, this command can set a minimum area to influence which wire load is selected. If there are no wire_load_selection tables, a default_wire_load is used if there is one in the library. If the library has neither of the aforementioned, no wire load model is used.

EXAMPLES

The following example sets no wire load model on the subdesign "LOW", leaving it to be selected based on its cell area. Then, the wire load model is set to "20x20" and the wire load model mode is set to *enclosed* for the top-level design "TOP", with a **min_block_size** argument specifying to assume a block size of at least 200.0 for all subdesigns (in this case, LOW).

```
prompt> current_design LOW /* no wire-load set now */  
prompt> current_design TOP  
prompt> set_wire_load_mode enclosed  
prompt> set_wire_load_min_block_size 200.0  
prompt> set_wire_load_model -name "20x20"
```

SEE ALSO

characterize(2)

set_wire_load_min_block_size
1828


```
current_design(2)
report_lib(2)
reset_design(2)
set_wire_load_model(2)
set_wire_load_mode(2)
set_wire_load_selection_group(2)
remove_wire_load_model(2)
set_load(2)
set_local_link_library(2)
link_library(3)
```

set_wire_load_mode

Sets the **wire_load_model_mode** attribute on the current design, specifying how wire load models are to be used to calculate wire capacitance in nets.

SYNTAX

```
status set_wire_load_mode  
mode_name
```

Data Types

```
mode_name      string
```

ARGUMENTS

mode_name

Specifies the value with which to set the **wire_load_model_mode** attribute, thus specifying the mode to be used to handle hierarchical wire load models. Allowed values are as follows:

- The value of *top* (the default) specifies that the wire capacitance of all nets is calculated using the wire load model set on the top-level design.
- The value of *enclosed* specifies that the wire capacitance of each net is calculated using the wire load model set on the smallest subdesign that completely encloses that net. You must specify this mode in order to use the **set_wire_load_selection_group** and **set_wire_load_min_block_size** commands.
- The value of *segmented* specifies that for each net that crosses hierarchical subdesigns, the wire capacitance is calculated for each segment of the net based on the wire load model set on the subdesign that contains that segment. The total wire capacitance of the net is the sum of the wire capacitances of its segments.

DESCRIPTION

The **set_wire_load_mode** command sets the **wire_load_model_mode** attribute on the current design, specifying how hierarchical wire load models are to be used to calculate wire capacitance of nets in the current design. If you use the **set_wire_load_min_block_size** command to set the **min_block_size** attribute and execute this command to set the mode_name to be different from *enclosed*, the **min_block_size** attribute is canceled.

The **report_design** command displays the wire loading model and mode for the current design, and the libraries to which the current design is linked.

To specify distinct wire load models for various hierarchical cells in the top-level design, specify wire load models on the designs referenced by these hierarchical cells; wire load models are reference-specific and not instance-specific. You must also ensure that the **wire_load_model_mode** attribute of the topmost design is not *top*; otherwise, the wire load models selected for lower-level blocks in the design

are ignored.

If you do not specify a **wire_load_model_mode** for a design, **compile** searches for a default **wire_load_model_mode** in the first library in the **link_path**. If that library does not have a default **wire_load_model_mode**, *top* is the default. If the mode for the top-level design is *top*, the wire load model on subdesigns and subclusters has no effect, and the top-level wire load model is used to compute wire capacitance for all hierarchical or top-level nets within a top-level design.

If the mode for the top-level design is either *enclosed* or *segmented*, wire load models on subdesigns and subclusters are used to calculate wire capacitance in nets inside these blocks. When the **auto_wire_load_selection** variable is *true*, if a wire load model is not specified for a subdesign, and if the first library in the **link_path** contains a **wire_load_selection** table, the **compile** command automatically selects a wire load model based on the cell area of the subdesign. If you do not define a **wire_load_selection** table, the wire load model of the parent design or cluster is used. If the top-level mode is *enclosed* or *segmented*, the wire load model of the subdesign is not set by **characterize** unless the subdesign does not have a wire load. When the **auto_wire_load_selection** variable is *false*, if a wire load model is not specified for a subdesign, no wire load model is selected.

If the model mode is *enclosed*, the wire load model of the smallest design that encloses a net completely is used to calculate the wire capacitance of that net. The number of pins on the net equals the number of leaf pins on the net.

EXAMPLES

The following example sets the **wire_load_mode** attribute to *enclosed* for the top-level design named TOP:

```
prompt> current_design TOP

prompt> set_wire_load_mode enclosed
```

SEE ALSO

```
characterize(2)
compile(2)
current_design(2)
remove_wire_load_model(2)
report_lib(2)
reset_design(2)
set_local_link_library(2)
set_wire_load_min_block_size(2)
set_wire_load_model(2)
set_wire_load_selection_group(2)
auto_wire_load_selection(3)
link_library(3)
```

set_wire_load_model

Sets the **wire_load_attach_name** attribute on designs, ports, hierarchical cells of current design, for selecting a wire load model to use in calculating wire capacitance.

SYNTAX

```
status set_wire_load_model  
-name model_name  
[-library lib]  
[-min] [-max]  
[object_list]
```

Data Types

<i>model_name</i>	string
<i>object_list</i>	collection

ARGUMENTS

-name *model_name*
Specifies the name of the wire load model to be used. The model must be defined in the library or in one of the libraries in the **link_library** (including the **local_link_library**). This is a required option.

-library *lib*
Specifies the library that contains the desired wire load model. This is either a library or a collection. The first library in the collection that contains the wire load model is selected.

-min
Indicates that the wire load model or selection group is to be used for minimum delay analysis only. You cannot use the **-min** option to set a different wire load mode or minimum block size, because both minimum and maximum delay analysis always use the same values for these parameters.

-max
Indicates that the wire load model or selection group is to be used for maximum delay analysis only. Any model set for minimum delay analysis is not affected.

object_list
Specifies a list of ports, designs, and/or cells, to which this wire load model is to be assigned. By default, the current design is assigned the wire load model.

DESCRIPTION

The **set_wire_load_model** command sets the **wire_load_attach_name** attribute on the specified ports, designs, or cells specified in the object list, or on the current design, for selecting a wire load model to use in calculating wire capacitance.

Issuing the command without options specifies a wire load model for the top-level design. The **-name** option is required.

When a design is specified, this wire load model is applied to all nets in the design at the top level and those under the hierarchy. If cells are specified, they are searched in the current design. The wire load model set on a cell overrides the wire load model of the design for the cell.

You can use this command to set a port's external wire load model. The wire load model set on a port overrides the wire load model of the design for the net connected to that port.

If no wire load model is specified for a design, a default is selected. The tool searches the first library on the link path (or the first library in the design's **local_link_library**) for a default wire load model to use. When the **auto_wire_load_selection** attribute is set to true, if the library has wire_load_selection tables, the cell area of the current design is used to automatically select the appropriate wire load model from the appropriate wire_load_selection table. If the library has more than one table, it typically specifies that one of the tables is the default (by setting the **default_wire_selection_group** attribute). You can override this value by using the **set_wire_load_selection_group** command. If you did not set the minimum block size by using the **set_wire_load_min_block_size** command, the cell area of the design is assumed to be at least as large as the value specified for this option when selecting the appropriate wire load to use. If there are no wire_load_selection tables, a default_wire_load is used if there is one in the library. If the library has neither of the aforementioned, no wire load model is used.

When the **auto_wire_load_selection** attribute is set to false, and no *model_name* is specified, no wire load model is used.

The **report_design** command displays the wire load model and mode for the current design, and the libraries to which the current design is linked. The **report_wire_load** command provides a more detailed report on the wire load models associated with a design.

The **report_lib** command shows the wire load models that are defined in the specified library, as well as any wire load selection tables that are defined. You can use the **report_port** command to display the wire load model of the ports of a design.

You can specify instance-specific wire load models for hierarchical cells in the current design. The wire load models on the designs referenced by these hierarchical cells are not changed.

If you specify wire load model on subdesigns, you must ensure that the wire load mode of the top design is not **top**; otherwise, the wire load models selected for lower-level blocks in the design are ignored.

If you do not specify a wire load mode for a design, the tool searches for a default wire load mode in the first library in the link path during **compile**. If that library does not have a default wire load mode, **top** is assumed. You can override a mode by using the **set_wire_load_mode** command. For a definition of the **top**, **enclosed**, and **segmented** values of the **wire_load_model_mode** attribute, and an explanation of how a default wire load model is selected, see the man page for the **set_wire_load_mode** command.

Wire resistance and wire area are also calculated according to the rules for wire capacitance.

With the **-min** option, you can use the **set_wire_load_model** command to specify a different wire load model to use for minimum delay analysis. Unless explicitly specified, minimum delay analysis uses the same wire loads that maximum delay analysis uses. You can set different wire loads for minimum delay analysis on designs or ports. You cannot specify a different wire load mode or a different minimum block size for minimum delay analysis only. In these cases these same value is always used for both minimum and maximum analysis.

Wire area is always calculated by using the same wire loads used for maximum delay analysis.

Wire loading models contain all the information required by **compile** to estimate interconnect wiring delays. The following is an example of a wire loading model definition, as it appears in the source text of a library.

```
wire_load("90x90") {    resistance : 0 ;
    capacitance : 1 ;
    area : 0 ;
    slope : 1.64 ;
    fanout_length( 1 , 1.9 ) ;
    fanout_length( 2 , 2.8 ) ;
    fanout_length( 3 , 4.7 ) ;
}
```

The name of this model is 90x90. The fields in this model are defined as follows:

resistance

The resistance per unit length of interconnect wire.

capacitance

The capacitance per unit length of interconnect wire.

area

The net-area per unit length of interconnect wire.

fanout_length (fanout1, length1)

These pairs of numbers specify a lookup table that the tools use to estimate the length of interconnect wire being driven by a cell with a given fanout. You must specify at least one of these pairs (with fanout = 1), and can specify as many additional pairs as necessary to characterize the desired fanout length behavior. Linear interpolation between the two nearest pairs is used to estimate any points that are not in the lookup table.

slope

This value is used to characterize linear fanout or length behavior beyond the last pair specified in the lookup table.

To remove a wire load model from a design, use the **remove_wire_load_model** command, or use the **reset_design** command.

set_wire_load_model

EXAMPLES

The following example sets the wire load model on the top-level design, on which there is no **local_link_library** set:

```
prompt> report_lib tech_lib
```

```
.  
. .  
.
```

Wire Loading Model:

NameLibrary	ResCap	Area	Slope	Fanout	Length
05x05tech_lib	0.00001	0.00000	0.00000	1860	100.3900
10x10tech_lib	0.00001	0.00000	0.00000	3110	10.5300

```
.  
. .  
.
```

```
prompt> link_library = {tech_lib.db}
```

```
prompt> set_wire_load_model -name "10x10"
```

In the following example, the **-library** option specifies a wire load model of 10x10 from the my_lib.db library:

```
prompt> set_wire_load_model -name "10x10" -library my_lib.db
```

The following example sets the 10x10 wire load model on the subdesign named LOW to be used for maximum and minimum delay analysis. Then, the wire load model is set to 20x20 and the wire load model mode is set to **enclosed** for the top-level design TOP. Finally, a different wire load model named 20x20MIN is set at the top level to be used for minimum delay analysis only.

```
prompt> current_design LOW
```

```
prompt> set_wire_load_model -name "10x10"
```

```
prompt> current_design TOP
```

```
prompt> set_wire_load_mode enclosed
```

```
prompt> set_wire_load_model -name "20x20"
```

```
prompt> set_wire_load_model -name "20x20MIN" -min
```

The following example sets no wire load model on the LOW subdesign, leaving it to be selected based on its cell area. Then, the wire load model is set to 20x20 and the wire load model mode is set to **enclosed** for the TOP top-level design, with a **-min_block_size** option specifying to assume a block size of at least 200.0 for all subdesigns (in this case, LOW).

```
prompt> current_design LOW /* no wire-load set now */
```

```
prompt> current_design TOP
```

```
prompt> set_wire_load_mode enclosed
```

```
prompt> set_wire_load_min_block_size 200.0
```

```
prompt> set_wire_load_model -name "20x20"
```

The following sequence of commands describes the external fanout of a port:

```
prompt> set_wire_load_model [get_ports O1] "default_wl"
```

```
prompt> set_port_fanout_number 5 [get_ports O1]
```

Assume U1, U2, and U3 are 3 cells in the design named TOP, and all of them are instances of a subdesign named LOW. The following commands set the 10x10 wire load model on LOW and the 20x20 wire load model on U1 and U2:

```
prompt> current_design TOP
prompt> set_wire_load_model -name "10x10" LOW
prompt> set_wire_load_model -name "20x20" {U1 U2}
```

SEE ALSO

```
characterize(2)
current_design(2)
remove_wire_load_model(2)
report_lib(2)
reset_design(2)
set_load(2)
set_local_link_library(2)
set_port_fanout_number(2)
set_wire_load_min_block_size(2)
set_wire_load_mode(2)
set_wire_load_selection_group(2)
auto_wire_load_selection(3)
link_library(3)
```


set_wire_load_selection_group

Specify a selection group to use for determining a wire load model to be assigned to designs and cells or to a specified cluster. This command is supported only for the *enclosed* wire load mode.

SYNTAX

```
int set_wire_load_selection_group  
[-library lib] [-min]  
[-max]  
group_name [object_list]
```

Data Types

<i>group_name</i>	string
<i>object_list</i>	list

ARGUMENTS

- library *lib***
Specifies the library from which to select the wire load model. This is either a library name or a collection. The first library in the collection that contains the wire load model is selected.
- min**
Specifies to use the selection group for minimum delay analysis only. You cannot use the **-min** option to set a different wire load mode or minimum block size, because both minimum and maximum delay analysis always use the same values for these parameters.
- max**
Specifies to use the selection group for maximum delay analysis. Any model set for minimum delay analysis is not affected.
- group_name**
Specifies the name of the selection group to use to determine the wire load model based on area.
- object_list**
Specifies a list of designs and/or cells, to which the wire load model selection group attributes are to be assigned. If the *object_list* is not specified, the current design is assigned the attributes.

DESCRIPTION

This command sets the **wire_load_model_selection_group** attribute to *group_name*, and the **wire_load_model_selection_group_lib** attribute to the first library in *lib* that contains the group, on the designs and cells specified in *object_list* or on the specified cluster of current design. If none of them is specified, current design is assigned the attributes. You can define a selection group for designs, cells, or a cluster, but not for ports.

The area of the specified selection group is used to determine the wire load model for designs and cells, or cluster. It is supported only for the *enclosed* wire load model mode, which you specify using the **set_wire_load_mode** command. This command has an effect only when the variable **auto_wire_load_selection** is *true*.

To view the wire load associated with a specific cluster, use the **report_clusters** command. For a more detailed report on the wire load models associated with either a design or a cluster, use the **report_wire_load** command. To view the wire loading models defined in a library, as well as any wire_load_selection tables that might be defined there, use the **report_lib** command.

With the **-min** option, you can use this command to specify a wire load selection group to use for minimum delay analysis. Unless explicitly specified, minimum delay analysis uses the same wire loads as does maximum delay analysis. Different wire loads for minimum delay analysis can be set on designs, cells, or clusters. You cannot specify a different wire load mode or a different minimum block size for minimum delay analysis only. In these cases, the same value is always used for both minimum and maximum analysis.

Wire area is always calculated based on the wire loads used for maximum delay analysis.

EXAMPLES

The following example sets the selection group for a sub-design LOW and a hierarchical cell U1/U2 in the top design.

```
prompt> current_design TOP
prompt> set_wire_load_selection_group 2layermetal {LOW U1/U2}
```

The following example sets the selection group for the current design.

```
prompt> current_design TOP
prompt> set_wire_load_selection_group 2layermetal
```

SEE ALSO

```
characterize(2)
current_design(2)
set_load(2)
remove_wire_load_model(2)
report_lib(2)
reset_design(2)
set_local_link_library(2)
set_wire_load_min_block_size(2)
set_wire_load_mode(2)
set_wire_load_model(2)
link_library(3)
```

set_wrapper_configuration

Sets the default wrapper configuration for the current design.

SYNTAX

```
int set_wrapper_configuration  
-class core_wrapper | shadow_wrapper  
[-style dedicated | shared]  
[-core core_list]  
[-dedicated_cell_type WC_D1 | WC_D1_S | none]  
[-shared_cell_type WC_S1 | WC_S1_S | none]  
[-dedicated_design_name design_name]  
[-shared_design_name design_name]  
[-register_io_implementation swap | in_place]  
[-use_dedicated_wrapper_clock true | false]  
[-safe_state 0 | 1 | none]  
[-delay_test true | false]  
[-max_length integer]  
[-chain_count integer]  
[-mix_cells true | false]  
[-input_shift_enable port_name]  
[-output_shift_enable port_name]  
[-maximize_reuse enable | disable]  
[-reuse_threshold threshold_value]  
[-use_system_clock_for_dedicated_wrp_cells enable | disable]  
[-end_of_chain_logic enable | disable]  
[-input_mask_bound mask_bound]  
[-output_mask_bound mask_bound]
```

Data Types

<i>core_list</i>	list
<i>design_name</i>	string
<i>integer</i>	threshold_value
<i>threshold_value</i>	integer
<i>mask_bound</i>	Pin or Port

ARGUMENTS

-class core_wrapper | shadow_wrapper
This option specifies the name of the class for which the wrapper configuration is applicable.
This is a mandatory option.

-style dedicated | shared
This option specifies the style to be used for core wrapping the design.
Default value of the option is dedicated.

-core **core_list**
This option specifies the list of core cells for which the configurtaion is applicable.
Default value of the option is no core cells i.e. the configuration is

applicable to the current design.

`-dedicated_cell_type WC_D1 | WC_D1_S | none`

This option specifies the cell type to be used for dedicated wrapper cells in core wrapping the current design.

Default value of the option is WC_D1.

`-shared_cell_type WC_S1 | WC_S1_S | none`

This option specifies the cell type to be used for shared wrapper cells in core wrapping the current design.

Default value of the option is WC_S1.

`-dedicated_design_name design_name`

This option specifies the name of the design to be used for dedicated wrapper cells in core wrapping the current design.

The specified design should have been specified with `define_dft_design` command.

There is no default for value this option.

`-shared_design_name design_name`

This option specifies the name of the design to be used for shared wrapper cells in core wrapping the current design.

The specified design should have been specified with `define_dft_design` command.

There is no default value for this option.

`-register_io_implementation swap | in_place`

This option specifies the implementation to be used for synthesizing shared wrapper cells in core wrapping the current design.

The default value of this option is swap.

`-use_dedicated_wrapper_clock true | false`

This option specifies if dedicated wrapper clock should be used for shared wrapper cells added by core wrapping.

The default value of this option is false.

`-safe_state 0 | 1 | none`

This option specifies the safe state to be used for wrapper cells added by core wrapping.

The default value of this option is none.

`-delay_test true | false`

This option specifies if additional wrapper test modes that are used for delay testing should be created by core wrapping. In these additional wrapper test modes all wrapper chains contain either input wrapper cells or output wrapper cells. Wrapper chains containing input wrapper cells and wrapper chains containing output wrapper cells are controlled by different wrapper shift enables.

The default value of this option is false.

`-max_length integer`

This option specifies the maximum length of a wrapper chain in all wrapper modes. Wrapper chain lengths in Scan Compression modes are not controlled by this option.

The default value of this option is -1 i.e. wrapper chain length is not

controlled this option.

-chain_count *integer*

This option specifies the maximum number of wrapper chains in all wrapper modes. Number of wrapper chains in Scan Compression modes are not controlled by this option.

The default value of this option is -1 i.e. number of wrapper chains created are not controlled by this option.

When both of the options "-max_length", "-chain_count" are not specified, wrapper chain length or number of wrapper chains are controlled by scan configuration options "-max_length" or "-chain_count".

-mix_cells true | false

This options specifies if wrapper cells for input or output ports can be mixed in a wrapper chain.

If this option is set to false, all wrapper cells in a wrapper chain are either associated with input ports or output ports but not both.

The default value of this option is true i.e. wrapper cells for input and output ports can be mixed in any wrapper chain.

-input_shift_enable *port_name*

This option specifies the wrapper shift enable port for wrapper chains that contain only wrapper cells for input ports.

This option is applicable only when the option "-mix_cells" is set to false. There is no default for this option.

-output_shift_enable *port_name*

This option specifies the wrapper shift enable port for wrapper chains that contain only wrapper cells for output ports.

This option is applicable only when the option "-mix_cells" is set to false. There is no default for this option.

-maximize_reuse enable | disable

This option specifies the tool to reuse IO registers in the fan in or fan out cone of ports as shared wrapper cells. These IO registers need not be on the sensitive path of the ports i.e. any combinational logic is allowed between the ports and the IO registers.

Dedicated wrapper cells are not added to ports that are not register bounded. Dedicated wrapper cells are added to ports that are connected to clock gatng cells or registers inside a DFT model.

Currently the tool expects to see the netlist of a DFT model in addition to the model as the model doesn't show the complete association of pins to the registers inside the model. If netlist is not loaded for a model cell, the tool assumes that the model has only combinational logic.

When a dedicated wrapper cell is added to a port that is connected to IO registers, user specified clock is used for the dedicated wrapper cell. In the absence of such specification, the most used clock domain of IO registers is used as the clock of the dedicated wrapper cell. If no such clock is found, a dedicated wrapper clock is used.

If an IO register is common to both an input port and a output port, the IO register is classified as an input IO register.

These IO registers are also used as normal scan cells in Internal_scan mode. Options '-style shared', '-register_io_implementation in_place', '-mix_cells false' are required with this option.

The default value of this option is disable.

`-reuse_threshold threshold_value`

This option specifies the maximum number of IO registers that can be reused as shard wrapper cells.

This option is applicable only when the option "`-maximize_reuse`" is set to enable.

When the number of IO registers detected for a port exceed the specified threshold value, a dedicated wrapper cell is added to the port.

The default value of this option is 1.

If the value of the option is set to 0, all IO registers associated with a port are reused as shared wrapper cells.

`-use_system_clock_for_dedicated_wrp_cells enable | disable`

This option affects wrapper functionality only when the option '`-style shared`' is set.

When a dedicated wrapper cell is added to a port and this option is enabled, the tool uses the system clock of IO registers of the port as the clock for the dedicated wrapper cell.

If the port is connected to a single IO register and the tool adds a dedicated wrapper cell to the port, the clock of the associated IO register is used as the clock for the dedicated wrapper cell.

Default wrapper clock is used for a dedicated wrapper cell if the associated port is not connected to any IO registers.

The tool issue an error if more than one IO registers are found for a port and all of these IO registers don't use the same clock domain and user didn't specify any clock for the port. The command is terminated in such a case.

If user specifies a clock for such a port, the tool treats the error as a warning and uses the specified clock as the clock for the dedicated wrapper cell.

The default value of this option is disable.

`-end_of_chain_logic enable | disable`

This option specifies the tool to add end of chain logic to all wrapper chains of all modes. Here is the description of the logic added to wrapper chains when this option is enabled.

All wrapper chains are driven with the following logic.

`masked_wsi = (~wse & wrapper_mode & input_mask_bound) | wsi`

The following logic is added to all wrapper chain outputs.

`masked_wso = ((~wse & wrapper_mode & input_mask_bound) | output_mask_bound) | wso`

wse - wrapper chain scan enable
wrapper_mode - the test mode in which the wrapper chain is active
wsi - wrapper chain's original scan input
masked_wsi - wrapper chain's masked(new) scan input
wso - wrapper chain's original scan output
masked_wso - wrapper chain's masked(new) scan output

The default value of this option is disable i.e. no end of chain logic is added to wrapper chains.

`-input_mask_bound mask_bound`

This option specifies the mask bound to be used in end of chain logic added by the tool for wrapper chain inputs.

The logic added is shown above, `mask_bound` can be a pin or a port.

This option is effective only if `-end_of_chain_logic` is enabled.

There is no default value for this option.

`-output_mask_bound mask_bound`

This option specifies the mask bound to be used in end of chain logic added

by the tool for wrapper chain outputs.
The logic added is shown above, `mask_bound` can be a pin or a port.
This option is effective only if `-end_of_chain_logic` is enabled.
There is no default value for this option.

DESCRIPTION

This command specifies the default configuration to be used by the core wrapper or shadow wrapper application.

Use the **`reset_wrapper_configuration`** command to reset the current wrapper configuration of the design.

EXAMPLES

The following example specifies that the `WC_D1_S` cell is to be used as dedicated wrapper cell, `WC_S1_S` to be used as shared wrapper cell with safe value 0 in core wrapping.

```
prompt> set_wrapper_configuration -class core_wrapper \  
        -dedicated_cell_type WC_D1_S -shared_cell_type WC_S1_S \  
        -safe_state 0  
prompt> set_wrapper_configuration -class core_wrapper \  
        -style shared -register_io_implementation in_place -mix_cells false \  
        -maximize_reuse enable -reuse_threshold 4
```

SEE ALSO

```
insert_dft(2)  
preview_dft(2)  
reset_wrapper_configuration(2)  
report_wrapper_configuration(2)  
set_boundary_cell(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)
```


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)`

shell_is_in_topographical_mode

Determines if the Design Compiler shell was invoked in topographical mode.

SYNTAX

status **shell_is_in_topographical_mode**

ARGUMENTS

The **shell_is_in_topographical_mode** command has no arguments.

DESCRIPTION

This command is used to check if the shell is in topographical mode. The command returns 1 if the shell is in topographical mode, or a 0 otherwise.

EXAMPLES

The following example shows how to check the mode in which the shell has been invoked:

```
prompt> if {[shell_is_in_topographical_mode]} {\
        echo "You invoked dc_shell in Topographical mode" \
} else {\
        echo "You invoked dc_shell in normal mode" \
}
```

SEE ALSO

dc_shell(1)

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" \
}
```

shell_is_in_xg_mode

Determines if the shell is in XG mode.

SYNTAX

status **shell_is_in_xg_mode**

ARGUMENTS

The **shell_is_in_xg_mode** command has no arguments.

DESCRIPTION

This command is used to check if the shell is in XG mode. The command returns 1 if shell is in XG mode, or 0 otherwise. For use in Tcl-based dc_shell only. Since XG is the only mode, this command will be obsoleted in the future.

EXAMPLES

The following example shows how to determine the mode in which the shell has been invoked:

```
prompt> if {[shell_is_in_xg_mode]} {\
           echo "You invoked dc_shell in XG mode" \
} else {\
           echo "You invoked dc_shell in normal mode" \
}
```

simplify_constants

Propagates constants and other information in the current design.

SYNTAX

```
status simplify_constants  
[-boundary_optimization]
```

ARGUMENTS

-boundary_optimization
Indicates that **simplify_constants** is to propagate information across all hierarchical boundaries in the design. Use the **set_boundary_optimization** command to perform boundary optimization on a particular set of subdesigns.

DESCRIPTION

This command propagates constants and unconnected information inside each module of the current design and simplifies the logic. When **-boundary_optimization** is used, this information is propagated across all boundaries.

EXAMPLES

In the following example, using library A, the constants are propagated in the current design:

```
prompt> link_library = {A}  
prompt> target_library = {A}  
prompt> read_file -format verilog my_design.v  
prompt> current_design  
prompt> simplify_constants
```

SEE ALSO

```
compile(2)  
set_boundary_optimization(2)  
set_logic_one(2)  
set_logic_zero(2)  
set_unconnected(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)`

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)`

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)
```

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)
report_partitions(2)
set_attribute(2)
set_compile_partitions(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)`
`MasterInstance(3)`
`report_partitions(2)`
`set_attribute(2)`
`set_compile_partitions(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)
```

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  
1864
```

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(2)  
context_check_status(3)  
syntax_check_status(3)
```

translate

Translates a design from one technology to another.

SYNTAX

```
int translate  
[-preserve_structure]
```

SYNTAX

```
int translate  
[-preserve_structure]
```

ARGUMENTS

-preserve_structure
Specifies that the structure of a design is to be preserved during **translate**.
The default is false.

DESCRIPTION

The **translate** command translates all components of the current design to components found in the target library. If the current design is hierarchical, all subdesigns below the current design are translated. With this command, designs are converted from one technology to another while preserving as much of the original gate structure as possible. Each cell in the design is replaced with a functionally-equivalent cell from the target library. If no corresponding cell is found, the cell is converted to a Synopsys generic logic form. The replacement cell selection is influenced by preferring or disabling specific library cells using the **set_prefer** and **set_dont_use** commands, and by specifying the types of registers using the **set_register_type** command.

The target libraries are specified by the **target_library** variable. The **local_link_library** of the top-level design is set to **target_library** after the design is linked.

The **translate** command does not operate on cells or designs that have the **dont_touch** attribute. After the translation process, cells that are not successfully translated are reported.

EXAMPLES

In the following example, a design is translated from library A to library B.

```
prompt> link_library = {A}  
  
prompt> read -f edif test.edif  
  
prompt> target_library = {B}
```

```
prompt> translate
```

SEE ALSO

```
compile(2)  
current_design(2)  
set_dont_touch(2)  
set_dont_use(2)  
set_prefer(2)  
set_register_type(2)  
set_local_link_library(2)  
uniquify(2)  
target_library(3)
```

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)

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

```
compile(2)  
current_design(2)  
set_boundary_optimization(2)  
set_dont_touch(2)  
set_dont_use(2)  
uniquify_naming_style(3)  
uniquify_keep_original_design(3)
```

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]  
[-add]  
[-remove]  
cell_list
```

ARGUMENTS

-name *bound_name*
Specifies name of the bound. The bound should be a move bound.

-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)`

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_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

```
compile(2)  
create_clock(2)  
set_drive(2)  
translate(2)  
current_design(3)
```

use_test_model

Specifies the subdesigns that will use the Core Test Language (CTL) models during scan architect and insertion.

SYNTAX

```
status use_test_model  
[-true design_list]  
[-false design_list]
```

ARGUMENTS

```
-true design_list  
    Specifies the set of designs for which test models are to be used.  
  
-false design_list  
    Specifies the set of designs for which test models are not to be used.
```

DESCRIPTION

This command is used to specify whether or not the test models are to be used by scan architect and insertion. The use of test models increases the capacity and runtime for the tool.

EXAMPLES

The following example specifies that test models are to be used for the des1 and des2 designs, and test models are not to be used for the des3 and des4 designs:

```
prompt> use_test_model -true {des1 des2} -false {des3 des4}
```

SEE ALSO

```
current_design(2)  
dft_drc(2)  
insert_dft(2)  
preview_dft(2)  
read_test_model(2)  
reset_design(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

`read_db(2)`
`search_path(3)`

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)
 - verilog** - IEEE Standard Verilog
 - svsim** - SystemVerilog netlist wrapper
 - vhdl** - IEEE Standard VHDL
- The **-write -f svsim** command writes out only the netlist wrapper, not the gate-level DUT. To write out the gate-level DUT, use **write -f verilog**. For further information, see the *SystemVerilog User Guide*.
- 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
- .v** - verilog
- .sv** - svsim
- .vhd** - vhdl

-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

```
write_file(2)
view_write_file_suffix(3)
```

write_bsd1

Generates the boundary-scan description language (BSD1) file for a boundary-scan design.

SYNTAX

```
int write_bsd1  
[-naming_check VHDL | BSD1 | none]  
[-output file_name]  
[-effort low | medium | high]
```

Data Types

file_name string

ARGUMENTS

-naming_check VHDL | BSD1 | none
Specifies the type of naming checks to perform on the design. *VHDL* (the default) checks for both VHDL and BSD1 reserved words; *BSD1* checks for BSD1 reserved words only. *none* disables all naming checks.

-output *file_name*
Specifies the name of the output BSD1 file. The default is *top_level_design_name.bsd1*.

-effort low | medium | high
Controls the effort used to search for implemented instructions. The time taken increases exponentially for a sequential search on instruction registers (IRs) of length 8 or more. The *low* effort uses only heuristics. The *high* effort enforces sequential search. The default *medium* effort first uses heuristics and then random opcode generation for limited iterations.

DESCRIPTION for all modes

Generates the BSD1 file for a boundary-scan design. The command invokes the Synopsys ANSI/IEEE Std. 1149.1 Compliance Checker under the hood. If a valid port-to-pin map exists for the boundary-scan design and if no errors occur during compliance checking, the **write_bsd1** command generates the BSD1 output file for the boundary-scan design.

You can change the default suffix name, **bsd1**, by setting the **test_bsd1_default_suffix_name** variable with another suffix name.

By default, the BSD1 file line length is 74 characters per line. You can specify a different line length using the **test_bsd1_max_line_length** variable. The specified maximum line length should not be less than 50 or more than 132. If you specify a line length outside this range, **write_bsd1** uses the default instead.

EXAMPLES

The following example generates the BSDL output file test.bsd for a boundary-scan design while performing BSDL naming checks on the identifiers used in the design.

```
prompt> write_bsd1 -naming_check BSDL -out test.bsd
```

SEE ALSO

```
check_bsd(2)  
current_design(2)  
read_pin_map(2)  
remove_pin_map(2)  
set_bsd_compliance(2)  
test_bsd1_default_suffix_name(3)  
test_bsd1_max_line_length(3)
```

write_compile_script

Writes a compile script for the specified design.

SYNTAX

```
int write_compile_script  
[-absolute_paths]  
[-hierarchy]  
[-no_reports]  
[-refining]  
-destination pass  
[-update design_list]  
[design]
```

Data Types

<i>pass</i>	string
<i>design_list</i>	string

ARGUMENTS

- absolute_paths**
Uses absolute path names (starting with the ACS working directory, as specified by the **acs_work_dir** variable) when specifying files within the compile script. By default, this command uses relative path names.
- hierarchy**
Writes a compile script for each subdesign in the hierarchy that does not have a **dont_touch** attribute. By default, this command writes a compile script only for the specified design.
- no_reports**
Specifies not to generate reports. By default, the compile script generates the following reports:
- timing (report_timing)
 - area (report_area)
 - constraints (report_constraints)
 - qor (report_qor)
- refining**
Specifies that the compile script is for an incremental compile, rather than a full compile. By default, the compile script is written for a full compile.
- destination *pass***
Specifies which ACS pass you are in. The command generates the subdirectory name using this value.
- update *design_list***
Specifies the designs to be updated in the already mapped design.

design

Specifies the design to generate a compile script for. You can specify only one design. An error occurs if the design does not exist in Design Compiler memory. If you do not specify a design, ACS uses the current design.

DESCRIPTION

Writes the following files to the `passn/scripts` directory under the ACS working directory (as specified by the `acs_work_dir` variable):

- `env` (This file contains the design environment.)
- `design.suffix` (This file contains the compile script.)

The **`acs_compile_script_suffix`** variable determines the *suffix* value. By default, the suffix is "autoscr".

If the data directories for the specified pass do not exist, this command creates them.

Automated Chip Synthesis supports the following options for compile script generation. These options are listed below in priority order:

1) Use a user-defined partition compile script. The user-defined partition compile scripts must be located in `scripts/passn/partition_name.scr`. 2) Use the default compile script with a user-defined partition compile. The user-defined partition compile strategies must be located in `scripts/passn/partition_name.compile`. 3) Use the default compile script with a user-defined default compile. The user-defined default compile strategy must be located in `scripts/passn/default.compile`. 4) Use the default compile script with the compile strategy indicated by the ACS compile attributes on the compile partitions. See the **`acs_compile_attributes`** man page for information about the ACS compile attributes.

The default compile script performs the following tasks:

1) Sets the design environment. 2) Reads the pre-compile `.db` file for the specified design. 3) For each subdesign, a) Reads the post-compile `.db` file. b) Places a `dont_touch` attribute on the subdesign. 4) Sets the current design to the specified design. 5) Links the design. 6) Applies constraints from the file `passn/constraints/design.con`. 7) Compiles the design using the selected compile strategy. 8) Saves all modified designs (top-level and non-dont_touched subdesigns). 9) Generates reports (optional).

The default compile strategy is:

```
# initial compile
set_cost_priority -delay
set_flatten false
compile -map_effort medium

# incremental compile
compile_limit_down_sizing = true
compile -map_effort high -incr
```

SEE ALSO

current_design(2)
acs_compile_script_suffix(3)
acs_constraint_file_suffix(3)
acs_global_user_compile_strategy_script(3)
acs_override_script_suffix(3)
acs_user_compile_strategy_script_suffix(3)
acs_work_dir(3)

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

`analyze(2)`
`define_design_lib(2)`
`elaborate(2)`
`get_design_lib_path(2)`
`read_file(2)`
`report_design_lib(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_file

Writes a design netlist or schematic from memory to a file.

SYNTAX

```
int write_file  
[-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)

verilog - IEEE Standard Verilog

vhdl - IEEE Standard VHDL

If you specify **verilog** as the output format and set the **enable_cell_based_verilog_writer** variable to **true**, the Milkyway cell-based hierarchical Verilog output (HVO) writer is used to write designs. If dc shell is under UPF mode, UPF related pragma will be output right after retention registers and isolation cells.

-hierarchy
Causes all designs in the hierarchy to be written, starting from the designs specified in *design_list*. If *design_list* is not specified then the current design's hierarchy is written. You can use this option with any other option except the **-power** option. If a design hierarchy does not completely link then unresolved design references are not written out. Interface logic modules (ILMs) are not written out unless *all* of the designs in the hierarchy are ILMs.

-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
.v - verilog
.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

List of designs or design files to write. If *design_list* is omitted then the current design is used as the design list. If the design list -- including any designs that are implicitly included by the *-hierarchy* option or by specifying file names -- contains ILMs, the ILMs are automatically excluded from the list of designs to write. (ILMs may be written, however, if *all* of the specified designs are ILMs.) Designs are not removed from memory after they have been written; to remove a design from memory you may use the **remove_design** command.

-library library_name

Writes the Synopsys database format object to the specified library.

DESCRIPTION

This command writes designs from memory to disk. If a design is modified in the tool, you must use the **write_file** command to save the design.

EXAMPLES

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_file {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_file 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_file -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_file -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_file -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_file 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_file top.ddc  
Error: 'top' doesn't specify a unique design  
Please use complete specification: full_file_name:design_name  
prompt> write_file /osi4/bill/dc/top.ddc  
Writing to file '/osi4/bill/dc/top.ddc'
```

The following example shows the use of the Milkyway cell-based hierarchical Verilog writer where the Milkyway design cell top is exported to a Verilog file named hvo.v.

```
prompt> write_file -format verilog -output hvo.v top
```

SEE ALSO

```
change_names(2)  
define_name_rules(2)  
read_file(2)  
view_write_file_suffix(3)
```

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_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_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_makefile

Writes a makefile that defines the dependencies and commands required to compile the specified design.

SYNTAX

```
int write_makefile  
[-absolute_paths]  
[-dependencies depends]  
-destination pass  
[-target target_name]  
[-lsf [-bsubargs bsub_args]]  
[-update design_list]  
[design]
```

Data Types

<i>depends</i>	string
<i>pass</i>	string
<i>target_name</i>	string
<i>bsub_args</i>	string
<i>design_list</i>	string
<i>design</i>	string

ARGUMENTS

- absolute_paths
Uses absolute path names (starting with the ACS working directory, as specified by the **acs_work_dir** variable) when specifying files within the makefile. By default, this command uses relative path names.
- dependencies *depends*
Specifies the dependencies for creating a design database. By default, the design database depends on the post-compile design databases for each compile partition in its hierarchy, the design's constraint file, and the design's compile script.
- destination *pass*
Specifies which ACS pass you are in. The command generates the subdirectory name using this value.
- target *target_name*
Specifies the makefile target. By default, the target is "all".
- lsf
Specifies that the makefile will be run using the Load Sharing Facility (LSF) bsub command when running gmake. By default, the makefile is created to run under the UNIX make or gmake command.
- bsubargs *bsub_args*
Specifies the arguments for the LSF bsub command. These arguments are passed directly to the LSF software. Automated Chip Synthesis does not validate

these arguments. This argument is valid only when used with the **-lsf** option.

-update *design_list*

Specifies the designs to be updated in the already mapped design.

design

Specifies the design to generate a makefile for. You can specify only one design. An error occurs if the design does not exist in Design Compiler memory. If you do not specify a design, ACS uses the current design.

DESCRIPTION

Creates a makefile that defines how to compile the specified design. A makefile defines the dependencies and commands required to build a target (in this case, compile a design). The makefile is written to *passn/file_name* under the ACS working directory (as specified by the **acs_work_dir** variable). The **acs_makefile_name** variable determines the *file_name* value. By default, the file name is "Makefile".

The makefile reads the .db files for each design (compile partitions and the top-level design) from the *passn/db/pre_compile* directory and writes the resulting design to the *passn/db/post_compile* directory. The **dont_touch** attribute setting has the following effect:

- If a design has the **dont_touch** attribute set, the makefile copies the design database from the pre-compile directory to the post-compile directory.
- If a design does not have the **dont_touch** attribute set, the makefile compiles the design using its constraints file (*passn/constraints/design.con*) and compile script (*passn/scripts/design.autoscr*).

If the data directories for the specified pass do not exist, this command creates them.

SEE ALSO

current_design(2)
acs_compile_script_suffix(3)
acs_constraint_file_suffix(3)
acs_db_suffix(3)
acs_makefile_name(3)
acs_work_dir(3)

write_milkyway

Writes out the design to Milkyway database format.

SYNTAX

```
status write_milkyway  
-output filename  
[-overwrite]  
[-scenario scenario_list]
```

Data Types

filename string

ARGUMENTS

-output *filename*
 Specifies the design file name to be written. For example, the following design files are under the CEL view in the *design_lib* Milkyway design library:

```
design_lib/CEL/design1_pre_route:1  
design_lib/CEL/design1_pre_route:2  
design_lib/CEL/design1_post_route:1
```

 The *design1_pre_route* and *design1_post_route* are the *filename* arguments. The 1 and 2 after the colons (:) are the version numbers of the file names. This is a required argument.

-overwrite
 Overwrites the existing version of the design under the CEL view. This option does not increment the version of the existing file name specified in the **-output** option.

-scenario *scenario_list*
 Specifies a list of scenarios to save.

DESCRIPTION

The **write_milkyway** command writes the design into the Milkyway database. The command writes the hierarchical netlist and synthesis constraints. If floorplan, placement, and routing data exists, this information is also written into the Milkyway design library. If the specified Milkyway design library does not exist, the library is created by **write_milkyway**. This command does not write a design in the Milkyway database if the design is not uniquified or not mapped.

The **write_milkyway** command requires that a variable be set to specify where to find the Milkyway design library. The **mw_design_library** variable must be set before using the **write_milkyway** command:

```
prompt> set mw_design_library <string>
```

The directory name can be a relative name or an absolute name. For a single session, only one Milkyway design library can be accessed. You cannot read or write to different Milkyway design libraries.

Internally, this command writes the design to a CEL view in the Milkyway design library. The exception is when the current design is an Interface Logic Model (ILM), in which case this command writes it to the ILM view. The UNIX structure of the sample the Milkyway design library (CEL/ILM view) is as follows:

```
my_design_lib/                (Milkyway design library)
my_design_lib/CEL/            (CEL view)
my_design_lib/CEL/design1_john:1 (John's CEL data for design1)
my_design_lib/CEL/design1_mike:1 (Mike's CEL data for design1)
my_design_lib/ILM/            (ILM view)
my_design_lib/ILM/design1_john:1 (John's ILM data for design1)
```

Assume the **mw_design_library** variable is set to *my_design_lib*. The UNIX file called *my_design_lib/CEL/design1_john:1i* is created when you run the following command:

```
prompt> write_milkyway -output design1_john
```

For LEF or DEF users: when you create *mw_design_lib*, you must create a TF and attach it to the *mw_design_lib*. This is a manual process that takes time. The **write_milkyway** command provides an easy way, under-the-hood, to convert the PDB into TF by setting the following variable:

```
prompt> set mw_cel_without_fram_tech true
```

Do not set **mw_cel_without_fram_tech** if you already have the *mw_design_lib* with the original TF, otherwise, the original TF is overwritten by the **write_milkyway** command.

EXAMPLES

In the following example, the contents of the design are written to the *MW_DESIGN_LIB* design library. The saved design file is called *RISC_CORE*.

```
prompt> set mw_design_library MW_DESIGN_LIB
prompt> write_milkyway -output RISC_CORE
```

For LEF or DEF users only: in the following example you convert LEF into a PDB file called *cdn.pdb*. No TF file exists. The **write_milkyway** command automatically creates *mw_design_lib cdn_test*. It also generates the TF file from *cdn.pdb*, and writes the CEL *cdn_test_design*. Set **mw_cel_without_fram_tech** to **true**, to avoid preparing the TF manually.

```
prompt> set physical_library "cdn.pdb"
prompt> set mw_design_library "cdn_test"
prompt> set mw_cel_without_fram_tech true;
```

```
prompt> read_def cdn.def  
prompt> write_milkyway -out cdn_test_design
```

In the following example, an ILM is created and saved to the ILM view in the `mw_design_library MW_DESIGN_LIB`. The **write_milkyway** command saves to the ILM view because the design in-memory after the **create_ilm** command is an Interface Logic Model.

```
prompt> set mw_design_library MW_DESIGN_LIB  
prompt> create_ilm -physical  
prompt> write_milkyway -output RISC_CORE
```

SEE ALSO

```
create_ilm(2)  
mw_design_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 in SPEF format or as a Tcl script containing **set_load** and **set_resistance** commands.

SYNTAX

```
status write_parasitics
[-output file_name]
[-format reduced | distributed]
[-min]
[-ratio ratio_number]
[-script]
```

Data Types

<i>file_name</i>	string
<i>ratio_number</i>	float

ARGUMENTS

-output *file_name*
Specifies the name of the output file to which parasitics for the current design are written.
If a file is not specified, the parasitics are written to a name in the following format:
design_name.<format_name>.
where *design_name* is the name of the current design.

-format reduced | distributed
Specifies whether to write parasitics in reduced or distributed SPEF format. One resistance and capacitance per net exists in reduced format. In distributed format, the entire RC tree is written out for each net.

-min
Writes parasitics for the minimum operating condition. By default, the parasitics are written for the maximum operating conditions.

-ratio *ratio_number*
Specifies the ratio used when writing the pie model description for every driver in the net. The ratio specified by this command must be in the range 0.0 - 1.0. By default, the nets in reduced format are generated with a ratio equal to 0.5. The capacitor closest to the driver is 0.5 (the ratio value) times the total net capacitance in the pie model description.

-script
Writes out **set_load** and **set_resistance** commands instead of an SPEF file.

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 in reduced SPEF format to a file called top.reduced. If there are minimum and maximum operating conditions, parasitics for both conditions are written.

```
prompt> write_parasitics -format reduced -output top
```

```
*SPEF 1.0
*DESIGN "filter"
*DATE "Tue May 28 10:01:39 1996"
*VENDOR "SYNOPTIS INC"
*PROGRAM "Synopsys Design Compiler cmos"
*VERSION "3.5a-SI2-SCM"
*DESIGN_FLOW "SYNTHESIS"
*DIVIDER /
*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_partition

Writes the database for a design into the Automated Chip Synthesis data structure.

SYNTAX

```
int write_partition  
-type pre | post  
[-destination pass]  
[-hierarchy]  
[design]  
[-format db | ddc]
```

ARGUMENTS

-type pre | post

Specifies which Synopsys database format (.db) file to write. The valid keywords are **pre**, for the precompile .db type, and **post**, for the postcompile .db type. The **pre** and **post** values are mutually exclusive. If you specify the **pre** value, the command writes the .db file in the *passn/db/pre_compile* directory. If you specify the **post** value, the command writes the .db file in the *passn/db/post_compile* directory.

-destination *pass*

Specifies which Automated Chip Synthesis pass you are in. Substitute the name of the pass you want for *pass*. The command generates the subdirectory name using this value.

-hierarchy

Writes .db files for all compile partitions and master instance reference designs in the hierarchy of the specified design. By default, the command writes .db files for the specified design and for the master instance reference designs.

design

Specifies the design to write. Specify only one design, substituting it for *design_name*. An error occurs if the specified design does not exist in Design Compiler memory. If you do not specify a design, Automated Chip Synthesis uses the current design.

DESCRIPTION

Writes the database for a design into the Automated Chip Synthesis data structure. For use only in *dc_shell-t* (Tcl mode of *dc_shell*). The command writes the design database to the file *design.suffix* in the Automated Chip Synthesis directory structure. The **acs_db_suffix** variable determines the *suffix* value. By default, the suffix is *db*. The path for the .db file is determined by the command arguments.

If the data directories for the specified pass do not exist under the Automated Chip Synthesis working directory (as specified by the **acs_work_dir** variable), the command creates them.

SEE ALSO

`current_design(2)`
`read_partition(2)`
`write(2)`
`acs_db_suffix(3)`
`acs_work_dir(3)`
`MasterInstance(3)`

write_partition_constraints

Writes out the timing constraints for a design.

SYNTAX

```
int write_partition_constraints  
[-hierarchy]  
-destination pass  
[-update design_list]  
[design]
```

ARGUMENTS

-hierarchy

Writes a constraint file for each subdesign in the hierarchy that does not have a dont_touch attribute. By default, this command writes a constraint file only for the specified design.

-destination *pass*

Specifies which Automated Chip Synthesis pass you are in. Substitute the name you want for *pass*. The command generates the subdirectory name using this value.

-update *design_list*

Specifies the designs to be updated in the already-mapped design. Substitute the list you want for *design_list*.

design

Specifies the design to generate constraints for. You can specify only one design, substituting it for *design*. An error occurs if the design does not exist in Design Compiler memory. If you do not specify a design, Automated Chip Synthesis uses the current design.

DESCRIPTION

Writes out the timing constraints for a design. For use only in dc_shell-t (Tcl mode of dc_shell). The command writes the constraints present on the design database to the file *passn/constraints/design.suffix* under the Automated Chip Synthesis working directory (as specified by the **acs_work_dir** variable).

The **acs_constraint_file_suffix** variable determines the suffix value. By default, the value is *con*.

If the data directories for the specified pass do not exist, this command creates them.

SEE ALSO

```
current_design(2)  
write_environment(2)  
acs_constraint_file_suffix(3)
```

acs_work_dir(3)

write_physical_constraints

Writes the script of Tcl commands to export the current physical constraint settings. This command is supported only in topographical mode.

SYNTAX

```
status write_physical_constraints  
-output tcl_file  
[-no_site_row]  
[-pre_route]
```

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.

-no_site_row
 Indicates that site rows commands are not dumped in the output file. By
 default, site row commands are written out to the output file.

-pre_route
 Writes pre-routes to the output file. By default, pre-routes are not written
 to the output file.

DESCRIPTION

The **write_physical_constraints** command generates a script file containing the Tcl commands used to set the physical constraints that have been applied to the design. The script file is saved in the file specified with the **-output** option.

You can use the generated script file to reapply the physical constraints at a later time.

The output script file includes the following commands:

```
create_die_area  
set_placement_area  
set_utilization  
set_aspect_ratio  
set_rectilinear_outline  
set_port_side  
set_port_location  
set_cell_location  
create_placement_blockage  
create_wiring_keepouts
```

```
create_voltage_area
create_site_row
create_bounds
create_net_shape
```

For the physical constraints of pre-routes, the corresponding command is **create_net_shape**. In order to write out the commands, specify the **-pre_route** option.

Note that because of priority considerations, commands specifying the same type of information are not all written out. The following two rules help determine which commands are in effect and are written out:

- Among the four commands that set core area constraints the **set_placement_area** command has the highest priority, and **set_rectilinear_outline** has higher priority than **set_utilization** and **set_aspect_ratio**.
- Between the two commands that set port constraints, **set_port_location** has a higher priority than **set_port_side**.

For example, assume the following constraints are set:

```
set_placement_area -coordinate {10 10 1000 1000}
set_utilization 0.9
set_port_location clk -coordinate {100 200}
set_port_side -side top {clk reset}
```

Invoke the following command:

```
prompt> write_physical_constraints -output script.tcl
```

The result is that the following constraints are written to the script.tcl output file:

```
set_placement_area -coordinate {10 10 1000 1000}
set_port_location clk -coordinate {100 200}
set_port_side reset -side top
```

The utilization and the port side of the port clock are omitted, due to the constraint priority rules.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example command writes the physical constraints into the file named

```
write_physical_constraints
1926
```

script.tcl:

```
prompt> write_physical_constraints -output script.tcl
```

SEE ALSO

create_bounds(2)
create_net_shape(2)
create_placement_blockage(2)
create_site_row(2)
create_voltage_area(2)
create_wiring_keepouts(2)
extract_physical_constraints(2)
report_physical_constraints(2)
set_aspect_ratio(2)
set_cell_location(2)
set_port_location(2)
set_port_side(2)
set_placement_area(2)
set_utilization(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_rtl_load

Writes a script of RTL load commands for the current design.

SYNTAX

```
int write_rtl_load  
[-format dctl | dcsh]  
[-output file_name]
```

Data Types

file_name string

ARGUMENTS

-format *dctl* | *dcsh*
Specifies that the script is to be written in *dctl* or *dcsh* mode. By default, the script is written in the format of the mode from which this command is executed.

-output *file_name*
Specifies that the script is to be written to the specified file. If **-output** is not specified, the script is written to standard output.

DESCRIPTION

The **write_rtl_load** command writes a script of **set_rtl_load commands**. You can use this command to display the RTL loads on the current design, or to write a **set_rtl_load** script file to apply to another design.

By default, the **write_rtl_load** command writes commands to standard output. The **-output** option redirects the output to the specified disk file.

EXAMPLES

In the following example, the RTL loads of the current design are written to the file *test.scr*.

```
prompt> write_rtl_load -output test.scr
```

In the following example, the RTL loads of the current design are written in *dcsh* format to standard output.

```
prompt> write_rtl_load
```

SEE ALSO

```
set_rtl_load(2)  
write_script(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_scan_def

Writes scan chain information in SCANDEF format for performing scan chain reordering using place and route tools.

SYNTAX

```
int write_scan_def  
[-output output_command_file]  
[-expand_elements List_of_design_instance_names_that_should_be_expanded]
```

ARGUMENTS

-output *output_command_file*
Specifies the name of the output file that contains the scan chain information for the place and route tool. The default name is *current_design.def*, in the current directory.

-expand_elements *List_of_design_instance_names_that_should_be_expanded*
This option specifies a list of instances abstracted by test (CTL) models whose scan segments must be treated as 'flat' fully visible segments when generating the chip-level SCANDEF file.
The default behavior is to treat all instances abstracted by test (CTL) models as black boxes, i.e. represented in SCANDEF by "BITS" construct. This is unchanged from the current behavior of **write_scan_def** command.

DESCRIPTION

Generates scan chain information in SCANDEF format required for third party place and route tools to perform placement-based scan chain ordering. Prerequisites to using this command are as follows:

1. The design must contain scan elements. If it does not, add them by executing **insert_dft**. Perform design rule checking with **dft_drc**, and correct any design rule violations. Ensure to have a valid test protocol before performing **dft_drc**. If there is no test protocol information, then create one by using the command **create_test_protocol**.
2. Execute **dft_drc** to extract non-default scan chain information.

write_scan_def uses the scan chain information extracted by **dft_drc** to generate this output file:

- The data file that contains the scan chain information for the place and route tool, named by default *current_design.def*, in the current directory; you can use the **-out** option to change the filename from the default name. This file is generated in the appropriate format for the place and route tool specified, and is used as input by the place and route tool to generate the place and route report file.

After executing **write_scan_def**, you run the place and route tool with the SCANDEF file generated by **write_scan_def**.

3. This option specifies a list of instances abstracted by test (CTL) models whose scan segments must be treated as 'flat' fully visible segments when generating the chip-level SCANDEF file. The default behavior is to treat all instances abstracted by test (CTL) models as black boxes, i.e. represented in SCANDEF by "BITS" construct. This is unchanged from the current behavior of **write_scan_def** command. In a flow where the user uses block-level test-models for chip-level DFT insertion, the user needs to write a SCANDEF at the block-level before writing out a test-model in order to make the SCANDEF information available at the chip level for expansion. In a flow where the user uses block-level netlist for chip-level DFT insertion and doesn't write SCANDEF before writing the netlist, block level scan constraints (like `scan_path` or `scan_group`) will not be considered when expanding at chip level.

Please note that when the block-level test-models are used at chip level DFT insertion and when **-expand** is used to expand these block-level information, **check_scan_def** command is not supported. **check_scan_def** is supported when the complete netlist information is available. Otherwise, **check_scan_def** when used in such cases will show that scan chain tracing FAILED.

EXAMPLES

The following example writes out a scan information file named HUNSLEY.def for the place and route tool for the current design HUNSLEY, using the default scan configuration.

```
prompt> current_design HUNSLEY
prompt> create_test_protocol
prompt> dft_drc
prompt> insert_dft
prompt> dft_drc
prompt> write_scan_def
prompt> check_scan_def
```

The following example writes out a scan information file named HUNSLEY.def with -expand option.

```
prompt> current_design HUNSLEY
prompt> create_test_protocol
prompt> dft_drc
prompt> insert_dft
prompt> dft_drc
prompt> write_scan_def -o scan.def -expand [list block1 block2]
prompt> check_scan_def
```

SEE ALSO

```
create_test_protocol(2)
dft_drc(2)
preview_dft(2)
insert_dft(2)
set_scan_configuration(2)
```

check_scan_def(2)

write_script

Writes shell commands to save the current settings.

SYNTAX

```
int write_script
[-hierarchy]
[-no_annotated_check] [-no_annotated_delay] [-no_cg]
[-full_path_lib_names] [-nosplit]
[-format dctl | dcsh]
[-include loop_breaking]
[-output file_name]
```

ARGUMENTS

-hierarchy

Writes commands for all designs in the hierarchy.

-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 dctl | 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]

```

In the following example settings on all the designs in the hierarchy are written to standard output. An example of the output file is also provided.

```
prompt> write_script -hierarchy
```



```
#####

# Created by write_script -format dtc1 on Mon Mar 20 07:21:11 2006

#####

# Set the current_design #
current_design top

remove_wire_load_model
set_dont_touch [get_cells mid1]

# Set the current_design #
current_design mid

remove_wire_load_model
set_dont_touch [get_cells low1]

# Set the current_design #
current_design low

remove_wire_load_model
1
```

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_test

Formats the test patterns for the current design into one or more test vector files.

SYNTAX

```
status write_test  
[-output output_vector_file_name]  
[-format stil | stil_testbench | wgl_serial | verilog]
```

Data Types

output_vector_file_name string

ARGUMENTS

-output *output_vector_file_name*
Specifies as output the name of the path and base file name for the test program files. By default, test program files are written in the current directory, using the current design name as the base file name.

-format stil | stil_testbench | wgl_serial | verilog
Selects the format for the output test program. Valid values for *test_program_format* are as follows:

- Specifying **stil** generates STIL patterns.
- Specifying **stil_testbench** generates STIL patterns and a STILDPV Verilog test bench that uses the generated STIL patterns.
- Specifying **wgl_serial** generates patterns in serial WGL format.
- Specifying **verilog** generates Native Verilog test bench.

DESCRIPTION

The **write_test** command assembles the test patterns generated for the current design in Core Test Language (CTL) by the **create_bsd_patterns** command into a series of test program files suitable for running on automated test equipment (ATE) or simulators.

The output test program is intended as input to manufacturing ATE. The program specifies the stimuli to be applied to a fabricated chip and specifies the responses that a fully working chip generates. If during the testing process, the chip does not generate the correct responses when the stimuli are applied, the chip is faulty and should be discarded.

By default, all test vectors are placed into a single test program file.

The organization of the test program, in terms of the precise boundary scan test sequence employed, is determined by the boundary scan test protocol for the design inferred by the **create_bsd_patterns** command.

By default, the values used for the 4 timing options are those specified in the test protocol file for the design. In the protocol inferred by the **create_bsd_patterns** command, the default timing values are controlled by the environment variables: **test_default_period**, **test_bsd_default_strobe**, **test_bsd_default_delay**, and **test_bsd_default_bidir_delay**. In addition, the timing of any clock signals can again be specified in a test protocol file or can be specified directly by using the **set_dft_signal** command.

EXAMPLES

The following example writes out a test program for a design named *HUNSLEY* using the STIL format:

```
prompt> current_design HUNSLEY
prompt> write_test -format stil
Getting test program from design.
Writing test patterns to file HUNSLEY.stil.
1
```

The following example creates a STIL test bench:

```
prompt> write_test -format stil_testbench
Getting test program from design.
Writing test patterns to file HUNSLEY.stil.
Generating Test Bench ...
1
```

The following example creates a Native Verilog test bench:

```
prompt> write_test -format verilog
Getting test program from design.
Writing native Verilog test bench to file HUNSLEY_verilog_test.v.
1
```

SEE ALSO

```
check_bsd(2)
create_bsd_patterns(2)
current_design(2)
insert_dft(2)
set_dft_signal(2)
search_path(3)
test_bsd_default_bidir_delay(3)
test_bsd_default_delay(3)
test_bsd_default_strobe(3)
test_default_period(3)
```

write_test_model

Writes a test model file.

SYNTAX

```
int write_test_model  
[-format ctl | ddc]  
[-names verilog | verilog_single_bit]  
[-output model_file]  
[-inclusive]  
[-design design_name]
```

Data Types

<i>model_file</i>	string
<i>design_name</i>	string

ARGUMENTS

- format** *ctl* | *ddc*
Specifies the format of the test model file. A test model can be stored in a design *.ddc* file or as ASCII text in a CTL file. When **ddc** is specified, a copy of the DC design information and the CTL model are written out to a file. The default format is **ddc**.
- names** *verilog* | *verilog_single_bit*
Specifies the form of the names used in the CTL (ASCII) model. By default, names are not changed from internal representation. Names can be modified as Verilog names or as Verilog names compatible with the usage of the **verilogout_single_bit** environment variable. In all cases the internal representation is not changed.
- output** *model_file*
Specifies the name of the output test model file. By default, the output file name is *design_name.format*.
- inclusive**
Expands the include statement when writing out an ASCII CTL file. By default the value is false, and the included contents are written out to separate files.
- design** *design_name*
Specifies the name of the design for which the test model is written. The default is the current design.

DESCRIPTION

The **write_test_model** command writes test models to disk files. When the format is specified as **ddc**, a single test model for the specified design is written out as a *.ddc* file. The file contains the design interface definition and the test model. The **write_test_model** command ignores all implementation information, in particular all

subdesign information. For a .ddc file that contains only a design interface definition and test models, the **write** and **write_test_model** commands are synonymous.

When the format is specified as **ctl**, a single test model is written to a CTL ASCII file. The CTL format is primarily used for exporting test models outside of a Synopsys environment. The **write_test_model** command does not modify the version of the design that is in memory.

The **write_test_model** command is insensitive to the environment variable **test_stil_netlist_format**.

EXAMPLES

The following example writes out the test model to disk in .ddc format:

```
prompt> write_test_model -format ddc -output ALARM_SM_2.ddc
Writing test model file
'/remote/dtgnat/felixng/DFTC/bbs/alpha/dft_tutorial/ctl/
ALARM_SM_2.ddc'...
1
```

The following example writes out the test model to disk in CTL format:

```
prompt> write_test_model -format ctl -output ./ctl/ALARM_SM_2.ctl
Writing test model file
'/remote/dtgnat/felixng/DFTC/bbs/alpha/dft_tutorial/ctl/
ALARM_SM_2.ctl'...
1
```

SEE ALSO

```
list_test_models(2)
read_test_model(2)
remove_test_model(2)
report_test_model(2)
write(2)
current_design(3)
```

write_test_protocol

Writes a test protocol file.

SYNTAX

```
int write_test_protocol  
[-design design_name]  
[-output file_name]  
[-test_mode mode_name]  
[-names format_name]  
[-instruction instruction_name]
```

Data Types

<i>design_name</i>	string
<i>file_name</i>	string
<i>mode_name</i>	string
<i>format_name</i>	string

ARGUMENTS

- design *design_name***
Specifies the design name for which the test protocol is written. The default is the current design.
- output *file_name***
Specifies the name of the text file to which the test protocol is written. If the *file_name*, *format*, and output files are not specified, the file is named *design_name.spf*.
- test_mode *mode_name***
Specifies the test mode from which the protocol is generated.
- names *format_name***
Specifies the form of the names used in the STIL protocol. Valid values for *format_name* are as follows:
- The value of **dc_shell** instructs DFT Compiler to write a STIL protocol where the design objects are not changed.
 - The value of **verilog** instructs DFT Compiler to write a STIL protocol where the names of the design objects follow the naming rules of the Verilog format. Vectored ports are allowed.
 - The value of **verilog_single_bit** instructs DFT Compiler to write a STIL protocol where the names of the design objects follow the naming rules of the Verilog format. The STIL protocol file will not contain vectored ports, but it will have single bit ports.
Names can be unchanged from internal representation (the default). They can be modified as Verilog names or as Verilog names compatible with the usage of the **verilogout_single_bit** environment variable. In all cases, the internal representation is not changed. This option takes effect only in conjunction

with **-test_mode** options, when Hierarchical Scan Synthesis is used. In all other cases, the form of the names is determined by the setting of the **test_stil_netlist_format** variable.

DESCRIPTION

The **write_test_protocol** command outputs the test protocol from memory to disk in text format. This allows you to customize the test protocol.

The scan test protocol formally defines the process by which a design is tested, so the sequence of scan-in vectors, parallel vectors, and scan-out vectors is performed for each test pattern. The protocol defined for a design is used to drive scan test and design rule checking.

If the test protocol format and file name are not specified for this command, the default file names are *design_name.spf*.

If the **-test_mode** option is used, the protocol is generated from the CTL model attached to the design. The mode name specified indicates which protocol information is extracted from the several test modes that CTL describes.

In presence of On Chip Clock Controller, the protocols in both PLL clocks bypassed and non-bypassed modes will be included in the same protocol file.

EXAMPLES

The following example writes a test protocol to the *newUPC1.spf* file:

```
prompt> write_test_protocol -output newUPC1.spf  
Writing test protocol file 'newUPC1.spf'...
```

The following example writes a test protocol for the **UPC1** design to the default file named *UPC1.spf*:

```
prompt> write_test_protocol  
Writing test protocol file 'UPC1.spf'...
```

SEE ALSO

```
create_test_protocol(2)  
current_design(2)  
dft_drc(2)  
read_test_protocol(2)  
remove_test_protocol(2)
```