SYNOPSYS®
Predictable Success

Design Compiler Reference Methodology Training

2010.03-SP1

CONFIDENTIAL INFORMATION

The following material is being disclosed to you pursuant to a non-disclosure agreement between you or your employer and Synopsys. Information disclosed in this presentation may be used only as permitted under such an agreement.

LEGAL NOTICE

Information contained in this presentation reflects Synopsys plans as of the date of this presentation. Such plans are subject to completion and are subject to change. Products may be offered and purchased only pursuant to an authorized quote and purchase order. Synopsys is not obligated to develop the software with the features and functionality discussed in the materials.

Design Compiler Reference Methodology Training

- Overview
- Scripts and Customization
- Highlights of Recent Changes
- Roadmap

Design Compiler Reference Methodology Training

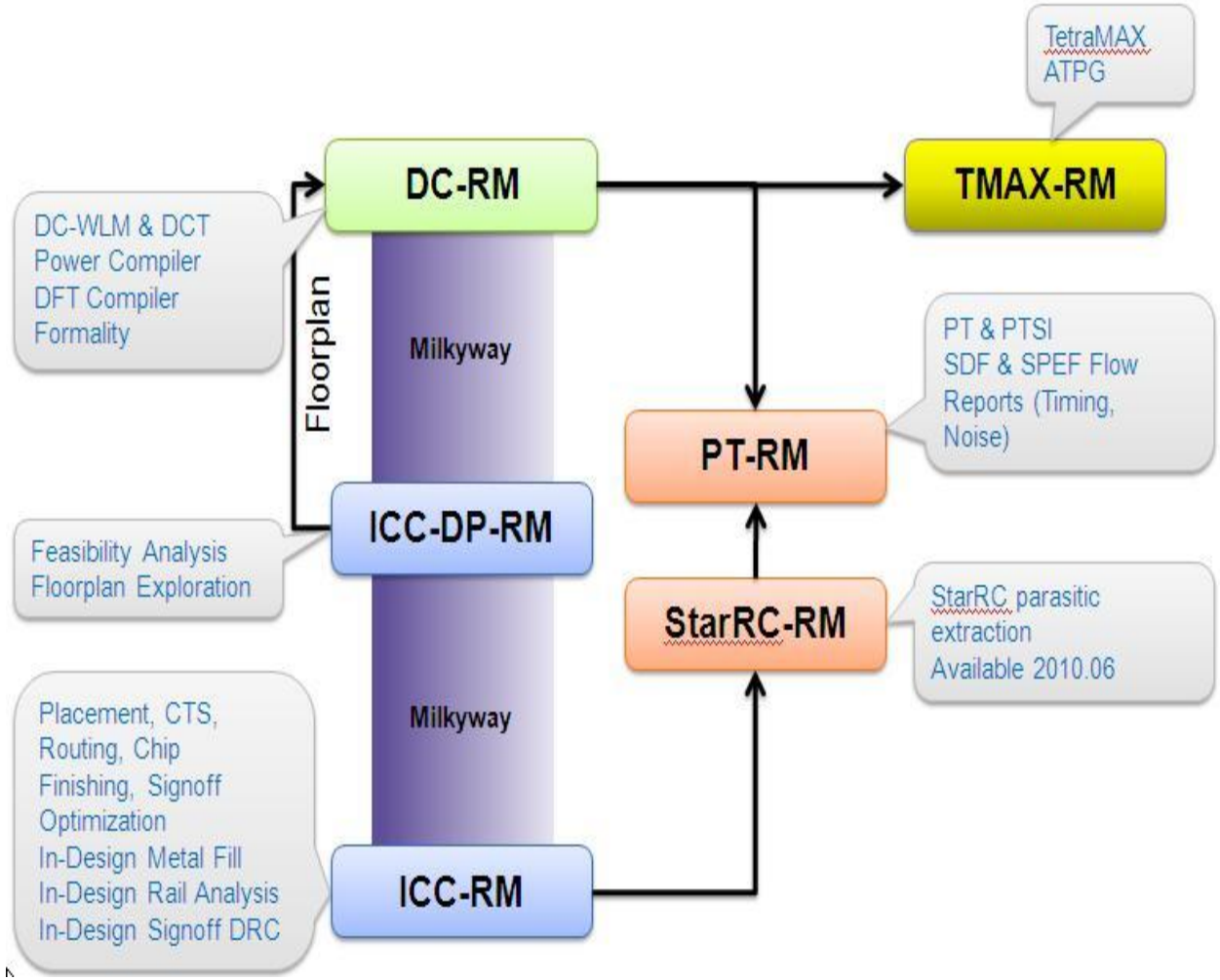
Overview

- Scripts and Customization
- Highlights of Recent Changes
- Roadmap

Reference Methodology Portfolio

Products Included

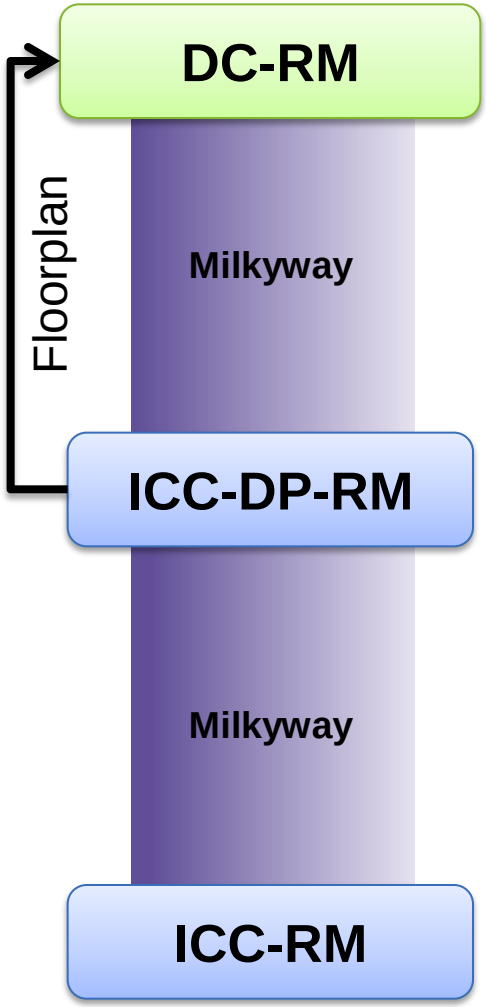
- **DC-RM**
  - Design Compiler
  - DFT Compiler
  - Power Compiler
  - Formality
- **PT-RM**
  - PrimeTime
  - PrimeTime SI
  - PrimeTime VX
  - PrimeTime PX
- **ICC-DP-RM**
  - ICC-DP
- **ICC-RM**
  - IC Compiler
  - In-Design
    - PrimeRail
    - IC Validator
- **TMAX-RM**
  - TetraMAX
- StarRC (FY'Q3 '10)



DC Reference Methodology (DC-RM)

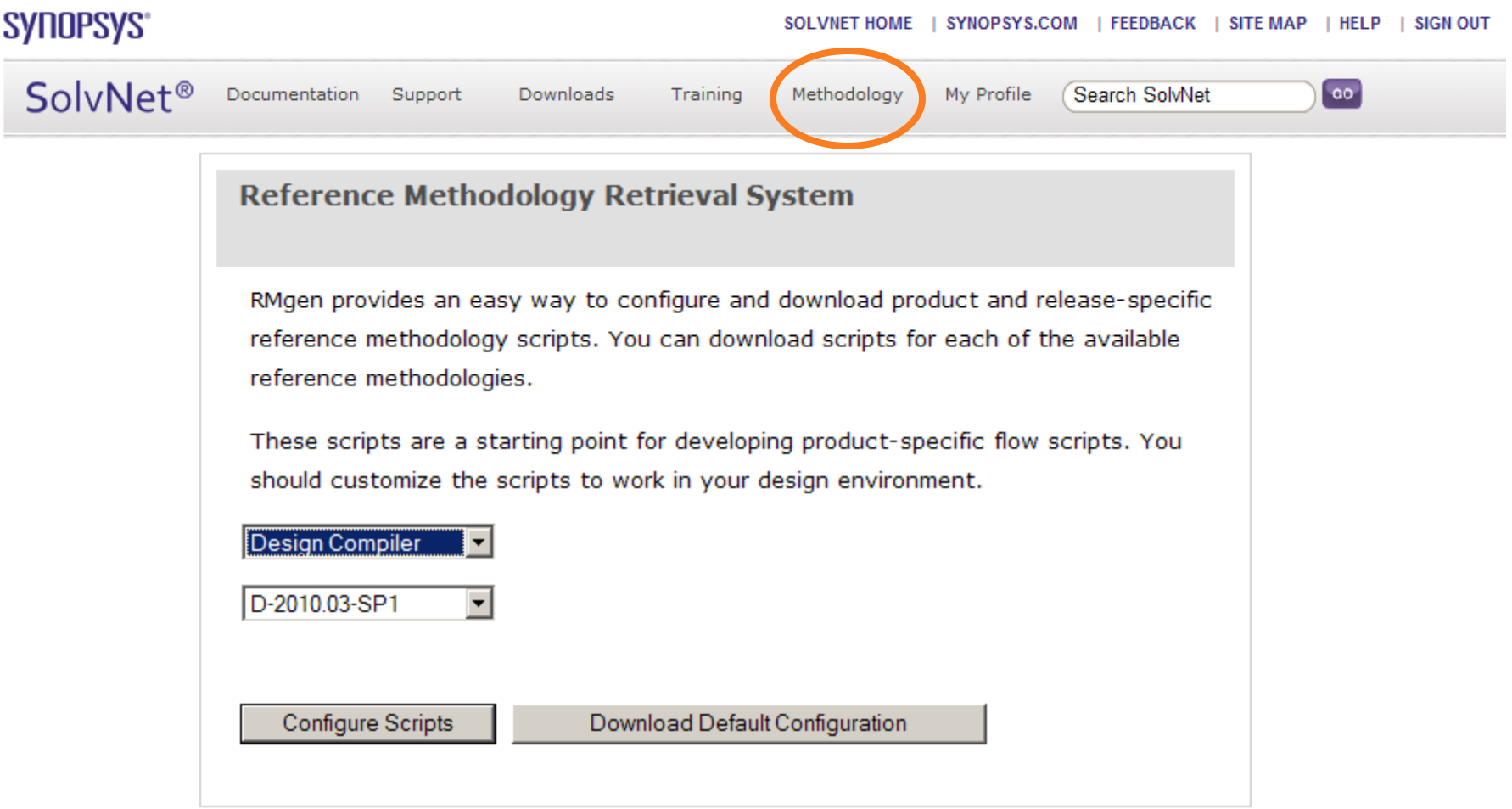
Do your scripts look like this?

- DC Reference scripts show you the latest recommended synthesis methodology
- Top-down and hierarchical synthesis flow covering: DCT, Power Compiler, DFT Compiler, and Formality
- Continue the implementation flow by using the IC Compiler Reference Methodology (ICC-*DP-RM & ICC-RM)
- Download it now from SolvNet using RMgen
  https://solvnet.synopsys.com/rmgen



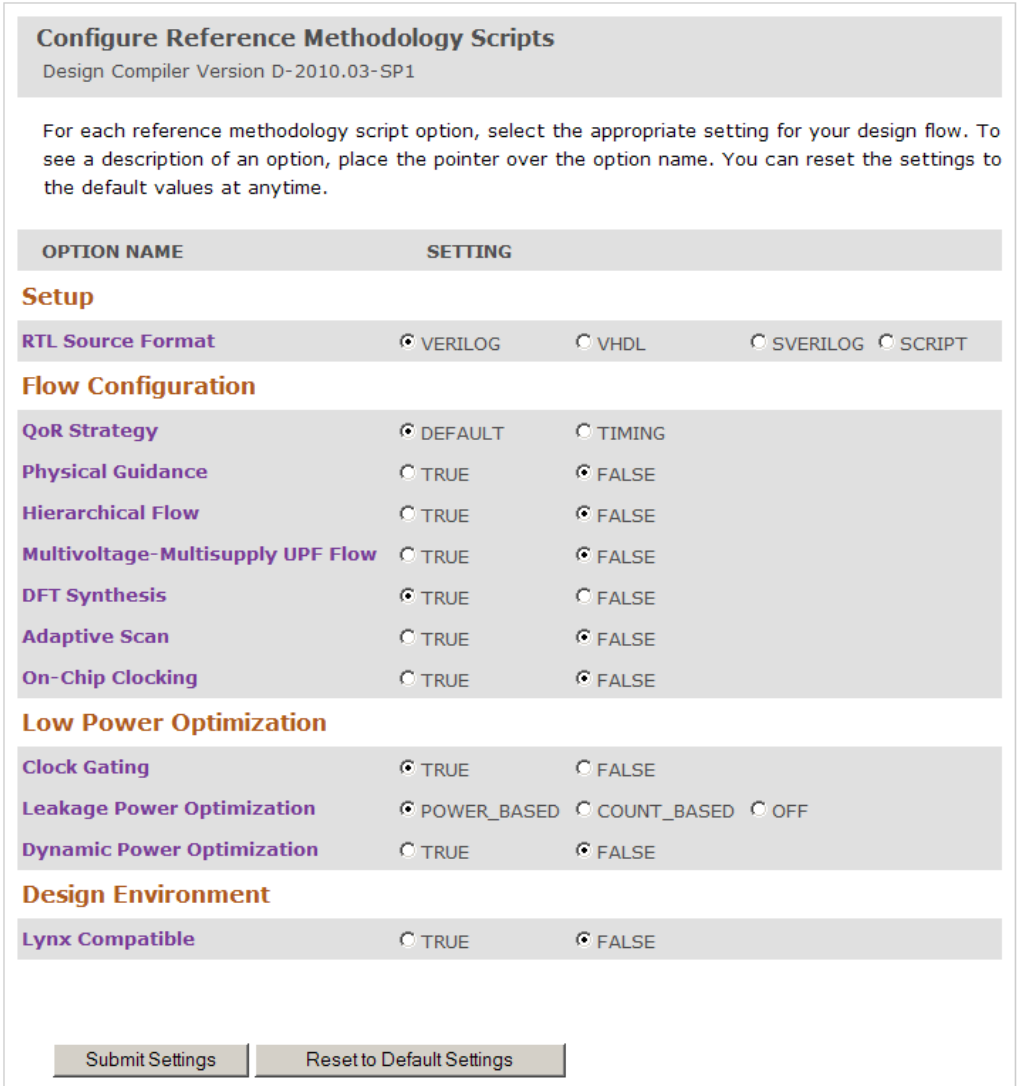
* Design Planning

Reference Methodologies on SolvNet



https://solvnet.synopsys.com/rmgen

Configure RM Scripts for Download



Configure Reference Methodology Scripts

Design Compiler Version D-2010.03-SP1

For each reference methodology script option, select the appropriate setting for your design flow. To see a description of an option, place the pointer over the option name. You can reset the settings to the default values at anytime.

OPTION NAME	SETTING			
Setup				
RTL Source Format	(•) VERILOG	( ) VHDL	( ) SVERILOG	( ) SCRIPT
Flow Configuration				
QoR Strategy	(•) DEFAULT	( ) TIMING		
Physical Guidance	( ) TRUE	(•) FALSE		
Hierarchical Flow	( ) TRUE	(•) FALSE		
Multivoltage-Multisupply UPF Flow	( ) TRUE	(•) FALSE		
DFT Synthesis	(•) TRUE	( ) FALSE		
Adaptive Scan	( ) TRUE	(•) FALSE		
On-Chip Clocking	( ) TRUE	(•) FALSE		
Low Power Optimization				
Clock Gating	(•) TRUE	( ) FALSE		
Leakage Power Optimization	(•) POWER_BASED	( ) COUNT_BASED	( ) OFF	
Dynamic Power Optimization	( ) TRUE	(•) FALSE		
Design Environment				
Lynx Compatible	( ) TRUE	(•) FALSE		

Submit Settings | Reset to Default Settings

What is the DC-RM Release Schedule?

- Reference Methodologies follow the tool release naming convention (2010.03, 2010.03-SP1, etc.)
- The RM release on SolvNet is 3-4 weeks after the tool release
- New tool features in major releases may be delayed to the RM service pack releases when the feature is proven mature
- Additional service pack RM versions are released, as needed
  - New tool capabilities in the service pack
  - New RM features

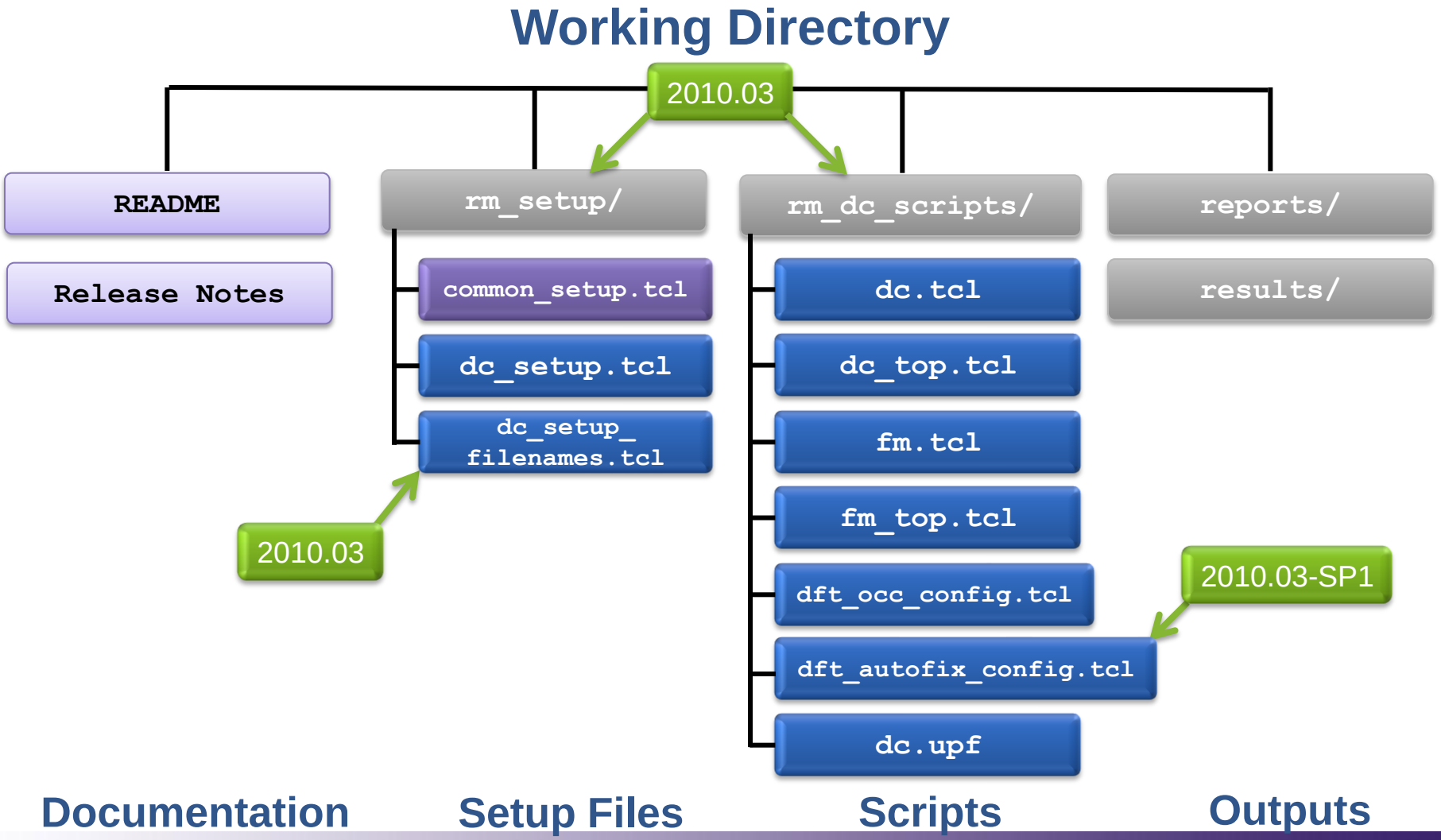
What is the DC-RM Usage Model?

- Users should start by selecting and downloading the RM flow of interest from SolvNet
- Users should use the RM scripts as a template for developing their flow
- Default settings in scripts are for the most complete flow
  - Power optimization and DFT
  - Floorplan input for DC-T
- Script is documented with embedded comments. Setup is designed to be clear and self-explanatory
  - README is included with high-level information
- Script is designed to automatically work for both DC and DCT modes
  - **`shell_is_in_topographical_mode`** command used to automate flow

Design Compiler Reference Methodology Training

- Overview
- Scripts and Customization
- Highlights of Recent Changes
- Roadmap

DC Reference Methodology Directory Structure



Running the DC Reference Methodology

1. Specify library, data inputs and options in setup files:

```
common_setup.tcl
```

```
dc_setup.tcl
dc_setup_filenames.tcl
```

2. Modify example run script for your design.

```
rm_dc_scripts/dc.tcl
```

3. Run modified script.

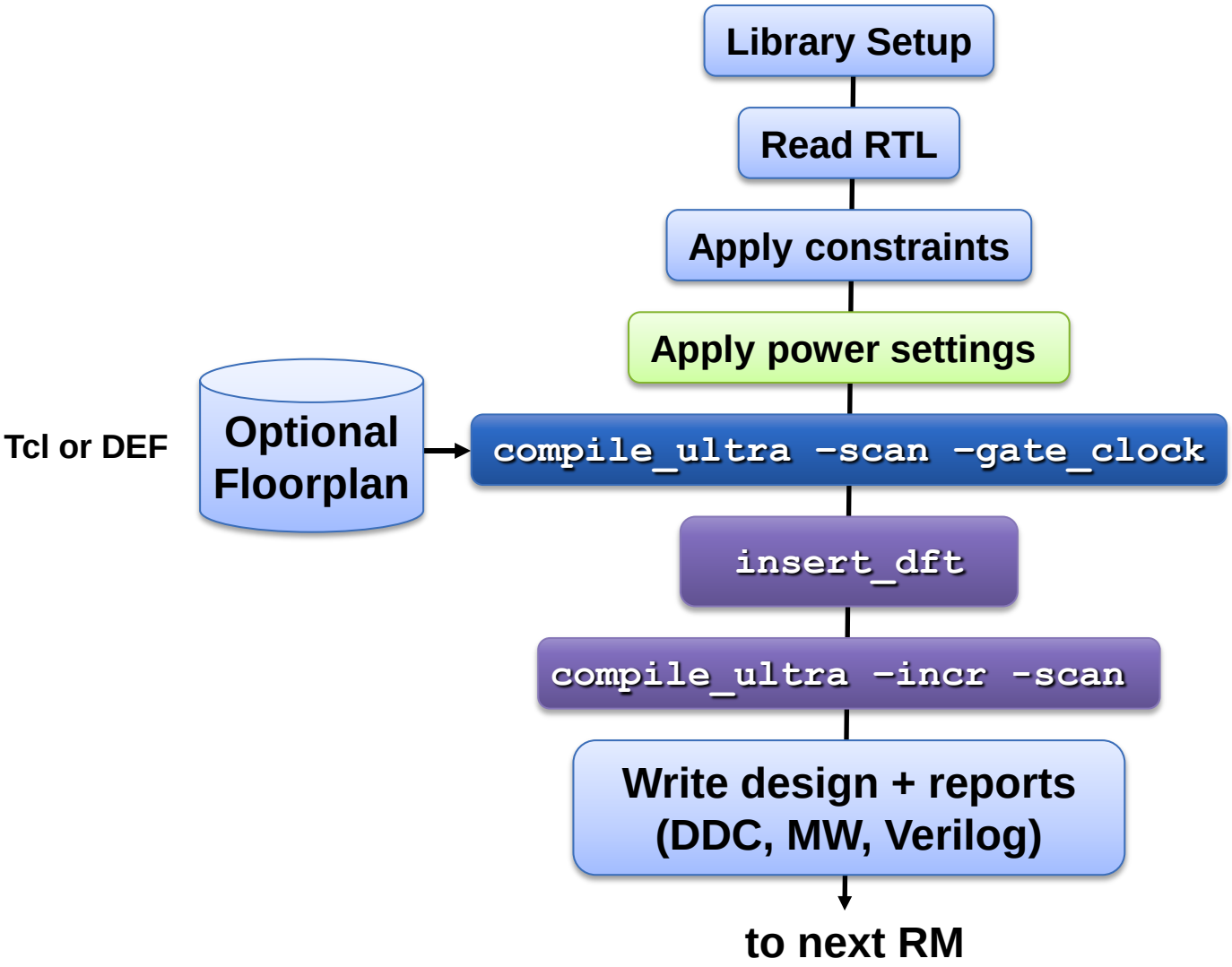
```
dc_shell -topo -f rm_dc_scripts/dc.tcl | tee dc.log
```

4. Check results:

```
reports/
```

```
results/
```

DC-RM Script Components



Setup Common RM Design Variables (common_setup.tcl)

```
set DESIGN_NAME                 ""   ;#  The name of the top-level design.

set ADDITIONAL_SEARCH_PATH      ""   ;#  Additional search path to be added
                                     #  to the default search path
set TARGET_LIBRARY_FILES        ""   ;#  Target technology logical libraries
set ADDITIONAL_LINK_LIB_FILES   ""   ;#  Extra link logical libraries
set MIN_LIBRARY_FILES           ""   ;#  List of max min library pairs "max1 min1 \
                                                                       max2 min2"
set MW_REFERENCE_LIB_DIRS       ""   ;#  Milkyway reference libraries

set TECH_FILE                   ""   ;#  Milkyway technology file
set MAP_FILE                    ""   ;#  Mapping file for TLUplus
set TLUPLUS_MAX_FILE            ""   ;#  Max TLUplus file
set TLUPLUS_MIN_FILE            ""   ;#  Min TLUplus file

set MW_POWER_NET                "VDD" ;#
set MW_POWER_PORT               "VDD" ;#
set MW_GROUND_NET               "VSS" ;#
set MW_GROUND_PORT              "VSS" ;#
```

Edit library setup info used by all RM scripts

- Common setup file all reference methodologies
- Common variables to set up libraries

Setup DC-RM Filename Variables (dc_setup_filenames.tcl)

2010.03

```
############################
# Milkyway Library Names #
############################

set DCRM_MW_LIBRARY_NAME                ${DESIGN_NAME}_LIB
set DCRM_FINAL_MW_CEL_NAME              ${DESIGN_NAME}_DCT

###############
# Input Files #
###############
set DCRM_SDC_INPUT_FILE                 ${DESIGN_NAME}.sdc
set DCRM_CONSTRAINTS_INPUT_FILE         ${DESIGN_NAME}.constraints.tcl

###########
# Reports #
###########

set DCRM_CONSISTENCY_CHECK_ENV_FILE     ${DESIGN_NAME}.compile_ultra.env
set DCRM_FINAL_QOR_REPORT               ${DESIGN_NAME}.mapped.qor.rpt
...
```

Customize the filenames referenced in the DC-RM

Setup DC Variables (dc_setup.tcl)

Further customize DC settings at top of dc_setup.tcl

```
source common_setup.tcl

##########################################################################
# Setup Variables
##########################################################################

# Point to a central cache of analyzed libraries
set_app_var alib_library_analysis_path .

# Enable multicore optimization to improve runtime
# set_host_options -max_cores 4

set RTL_SOURCE_FILES  ""    # Enter the list of source RTL files

set REPORTS_DIR "reports"
set RESULTS_DIR "results"
```

- Top part of **dc_setup.tcl** has some settings you can customize
- Will work without any changes

Setup Logical Libraries (dc_setup.tcl)

```
set_app_var target_library $TARGET_LIBRARY_FILES

set_app_var synthetic_library dw_foundation.sldb
set_app_var link_library "* $target_library \
                          $ADDITIONAL_LINK_LIB_FILES $synthetic_library"

# Set min libraries if they exist
foreach {max_library min_library} $MIN_LIBRARY_FILES {
  set_min_library $max_library -min_version $min_library  }
```

Library setup in dc_setup.tcl uses info from common_setup.tcl and should not need changes

Setup Milkyway Libraries (dc_setup.tcl)

```
set mw_reference_library ${MW_REFERENCE_LIB_DIRS}
set mw_design_library ${DCRM_MW_LIBRARY_NAME}

set mw_site_name_mapping [list CORE unit Core unit core unit]

if {[shell_is_in_topographical_mode]} {
  create_mw_lib   -technology $TECH_FILE \
                  -mw_reference_library $mw_reference_library \
                  $mw_design_library

  open_mw_lib     $mw_design_library

  set_tlu_plus_files -max_tluplus $TLUPLUS_MAX_FILE \
                     -min_tluplus $TLUPLUS_MIN_FILE \
                     -tech2itf_map $MAP_FILE
  check_tlu_plus_files
}
```

Designed to work automatically with settings in **common_setup.tcl**

Read RTL/database (dc.tcl)

```
##########################################################################
# Setup for Formality verification
##########################################################################

set_svf ${RESULTS_DIR}/${DCRM_SVF_OUTPUT_FILE}

##########################################################################
# Read in the RTL Design
#
# Read in the RTL source files or read in the elaborated design (DDC).
##########################################################################

define_design_lib WORK -path ./WORK
analyze -format verilog $RTL_SOURCE_FILES
elaborate $DESIGN_NAME

# OR

# read_ddc $DESIGN_NAME.elab.ddc
# current_design $DESIGN_NAME

link
```

2010.03

Select RTL SCRIPT when downloading the RM scripts if you have an RTL read script

Logical Constraints & Path Groups (dc.tcl)

```
##########################################################################
# Apply Logical Design Constraints
##########################################################################

source -echo -verbose ${DCRM_CONSTRAINTS_INPUT_FILE}

# set_app_var timing_enable_multiple_clocks_per_reg true

# Apply Operating Conditions
# set_operating_conditions -max <max_opcond> -min <min_opcond>

##########################################################################
# Create Default Path Groups
##########################################################################
set ports_clock_root [filter_collection [get_attribute [get_clocks] sources]
object_class==port]
group_path -name REGOUT -to [all_outputs]
...
```

Set up OP CONDS

Edit if you set up path groups

Path groups can improve QoR especially with uncertain I/O constraints

Consistent with ICC-RM

Power Optimization Flow (dc.tcl)

```
##########################################################################
# Power Optimization Section
##########################################################################
    ######################################################################
    # Clock Gating Setup
    ######################################################################

    # Default clock_gating_style suits most designs.  Change only if necessary.
    # set_clock_gating_style -positive_edge_logic {integrated} ...
    # set compile_clock_gating_through_hierarchy true
    # set_clock_gate_latency -clock <> -stage <#> -fanout_latency {...}

    ######################################################################
    # Apply Power Optimization Constraints
    ######################################################################

    # Include a SAIF file, if possible, for power optimization
    # read_saif -auto_map_names -input ${DESIGN_NAME}.saif \
                                -instance < DESIGN_INSTANCE > -verbose

    set_max_leakage_power 0
    # set_max_dynamic_power 0

    if {[shell_is_in_topographical_mode]} {
      # Use the following command to enable power prediction using clock tree estimation.
      # set_power_prediction true -ct_references <LIB CELL LIST>
    }
```

Set up clock gating

Include SAIF if available

Apply power optimization constraints

Uncomment power prediction if desired

Physical Constraints (dc.tcl)

```
if {[shell_is_in_topographical_mode]} {

# Apply Physical Design Constraints

  # set_fuzzy_query_options -hierarchical_separators {/ _ .} \
  #                         -bus_name_notations {[] __ ()} \
  #                         -class {cell pin port net} \
  #                         -show

if {[file exists [which ${DCRM_DCT_DEF_INPUT_FILE}]]} {
    extract_physical_constraints ${DCRM_DCT_DEF_INPUT_FILE}
  }

if {[file exists [which ${DCRM_DCT_FLOORPLAN_INPUT_FILE}]]} {
    read_floorplan ${DCRM_DCT_FLOORPLAN_INPUT_FILE}
  }

if {[file exists [which ${DCRM_DCT_PHYSICAL_CONSTRAINTS_INPUT_FILE}]]} {
    set_app_var fuzzy_matching_enabled true
    source -echo -verbose ${DCRM_DCT_PHYSICAL_CONSTRAINTS_INPUT_FILE}
    set_app_var fuzzy_matching_enabled false
  }
}
```

Default fuzzy matching works for most designs. Edit if needed

compile_ultra (dc.tcl)

2010.03-SP1

```
# Save the environment snapshot for Consistency Checker

# write_environment -consistency -output \
          ${REPORTS_DIR}/${DCRM_CONSISTENCY_CHECK_ENV_FILE}

#######
# Compile the Design
#
# Recommended Options:
#      -scan
#      -gate_clock
#
#      -retime
#      -timing_high_effort_script
#
#      -congestion
#      -spg
#############################################################

#compile_ultra -scan -gate_clock -check_only
compile_ultra -scan -gate_clock
```

Uncomment environment snapshot to debug correlation issues

Improve power and area

Improve timing

Improve design closure

Check for missing DCT data

insert_dft (dc.tcl)

```
##########################################################################
# DFT Compiler Optimization Section
##########################################################################

    ######################################################################
    # DFT Signal Type Definitions
    ######################################################################

    # set_dft_signal -type ScanDataOut -port SO -view spec
    # set_dft_signal -type ScanDataIn -port SI -view spec
    ...
    source -echo -verbose ${DCRM_DFT_SIGNAL_SETUP_INPUT_FILE}

    ######################################################################
    # DFT for Clock Gating
    ######################################################################

    # set_dft_drc_configuration -clock_gating_init_cycles 1
    # set_dft_signal -view spec -type <ScanEnable|TestMode> -port <port> -usage clock_gating
    # set_dft_connect <LABEL> -type clock_gating_control -source <DFT signal> [-target ...]
```

Define DFT signals

Specify dedicated ScanEnable/TestMode signal for clock gating

insert_dft (dc.tcl)

```
##########################################################################
# DFT Configuration
##########################################################################

set_dft_insertion_configuration -preserve_design_name true
set_dft_insertion_configuration -synthesis_optimization none
set_scan_configuration -clock_mixing mix_clocks

##########################################################################
# DFT AutoFix Configuration
##########################################################################

# Please refer to the dc.dft_autofix_config.tcl file included with the
# Design Compiler Reference Methodology scripts for an example...
# source -echo -verbose ${DCRM_DFT_AUTOFIX_CONFIG_INPUT_FILE}

##########################################################################
# DFT Adaptive Scan Compression Configurat
##########################################

set_dft_configuration -scan_compression enable
set_scan_compression_configuration -xtolerance default -min_power true
# set_dft_signal -view spec -type TestMode -port scan_compression_enable
```

2010.03-SP1

Configure DFT Autofix if needed

Configure Adaptive Scan Compression

insert_dft (dc.tcl)

```
##########################################################################
# DFT On-Chip Clocking (OCC) Configuration
##################################

# The following setting is used to enable OCC insertion/integration capability
# Include this setting, at the top of the ${DCRM_DFT_OCC_CONFIG_INPUT_FILE}
# set_dft_configuration -clock_controller enable

# Source the design-specific OCC setup file
source -echo -verbose ${DCRM_DFT_OCC_CONFIG_INPUT_FILE}

##########################################################################
# DFT Additional Setup
##########################

# Add any additional design-specific DFT constraints here

##########################################################################
# DFT Test Protocol Creation
##########################################################################

create_test_protocol
```

Configure on-chip clocking if used in your flow

Add any other settings needed for test protocol creation

2010.03

insert_dft (dc.tcl)

```
##########################################################################
# DFT Scan Chain Insertion
##########################################################################

dft_drc                    > .../${DCRM_DFT_DRC_CONFIGURED_SUMMARY_REPORT}
dft_drc -verbose           > .../${DCRM_DFT_DRC_CONFIGURED_VERBOSE_REPORT}
report_scan_configuration    > .../${DCRM_DFT_SCAN_CONFIGURATION_REPORT}
report_dft_insertion_configuration > ${DCRM_DFT_PREVIEW_CONFIGURATION_REPORT}

preview_dft              > .../${DCRM_DFT_PREVIEW_DFT_SUMMARY_REPORT}
preview_dft -show all -test_points all > .../${DCRM_DFT_PREVIEW_DFT_ALL_REPORT}

insert_dft

##########################################################################
# DFT Incremental Compile
# Only required if scan chain insertion has been performed.
#
# Include the -timing_high_effort_script option here if this option
# was used during the full compile_ultra optimization step.
##########################################################################

compile_ultra -incremental -scan
```

Writing Out the DFT Files (dc.tcl)

change_names before any final write

Customize output filenames in **dc_setup_filenames.tcl**

```
change_names -rules verilog -hierarchy

##########################################################
# DFT Write out Test Protocols and Reports
##########################################################
write_scan_def -output    ... ${DCRM_DFT_FINAL_SCANDEF_OUTPUT_FILE}
check_scan_def >          ... ${DCRM_DFT_FINAL_CHECK_SCAN_DEF_REPORT}
write_test_model -format ctl -output ... ${DCRM_DFT_FINAL_CTL_OUTPUT_FILE}

report_dft_signal > ${REPORTS_DIR}/${DCRM_DFT_FINAL_DFT_SIGNALS_REPORT}

# DFT outputs for regular scan
write_test_protocol -test_mode Internal_scan -output
${DCRM_DFT_FINAL_PROTOCOL_OUTPUT_FILE}
report_scan_path > ${REPORTS_DIR}/${DCRM_DFT_FINAL_SCAN_PATH_REPORT}
current_test_mode Internal_scan
dft_drc > ${REPORTS_DIR}/${DCRM_DFT_DRC_FINAL_REPORT}

# DFT outputs for adaptive scan compression
write_test_protocol -test_mode ScanCompression_mode -output
${RESULTS_DIR}/${DCRM_DFT_FINAL_SCAN_COMPR_PROTOCOL_OUTPUT_FILE}
current_test_mode ScanCompression_mode
report_scan_path > ${REPORTS_DIR}/${DCRM_DFT_FINAL_SCAN_COMPR_SCAN_PATH_REPORT}
dft_drc > ${REPORTS_DIR}/${DCRM_DFT_DRC_FINAL_SCAN_COMPR_REPORT}
```

Writing Out the Design Files (dc.tcl)

```
##################################################
# Write out Design
##################################################

set_svf -off

write -format ddc -hierarchy -output ...${DCRM_FINAL_DDC_OUTPUT_FILE}
write -f verilog -hierarchy -output ... ${DCRM_FINAL_VERILOG_OUTPUT_FILE}

##################################################################################
# Write out Design Data
#######################

if {[shell_is_in_topographical_mode]} {

  write_floorplan -all ... ${DCRM_DCT_FINAL_FLOORPLAN_OUTPUT_FILE}

  write_parasitics -output ... ${DCRM_DCT_FINAL_SPEF_OUTPUT_FILE}

  write_sdf ... ${DCRM_DCT_FINAL_SDF_OUTPUT_FILE}
}
write_sdc -nosplit ... ${DCRM_FINAL_SDC_OUTPUT_FILE}
```

Customize output filenames in **dc_setup_filenames.tcl**

2010.03

write_physical_constraints should no longer be used

Writing Out the Final Reports (dc.tcl)

```
##################################################
# Generate Final Reports
##################################################

report_qor > ${REPORTS_DIR}/${DCRM_FINAL_QOR_REPORT}
report_timing -transition_time -nets -attributes -nosplit >
${REPORTS_DIR}/${DCRM_FINAL_TIMING_REPORT}

if {[shell_is_in_topographical_mode]} {
  report_area -physical -nosplit > ${REPORTS_DIR}/${DCRM_FINAL_AREA_REPORT}
} else {
  report_area -nosplit > ${REPORTS_DIR}/${DCRM_FINAL_AREA_REPORT}
}

if {[shell_is_in_topographical_mode]} {
  report_congestion > ${REPORTS_DIR}/${DCRM_DCT_FINAL_CONGESTION_REPORT}
}
# Use SAIF file for power analysis
# read_saif -auto_map_names -input ${DESIGN_NAME}.saif -instance <
DESIGN_INSTANCE > -verbose

report_power -nosplit > ${REPORTS_DIR}/${DCRM_FINAL_POWER_REPORT}
report_clock_gating -nosplit > ${REPORTS_DIR}/${DCRM_FINAL_CLOCK_GATING_REPORT}
```

Customize output filenames in **dc_setup_filenames.tcl**

Design Compiler Reference Methodology Training

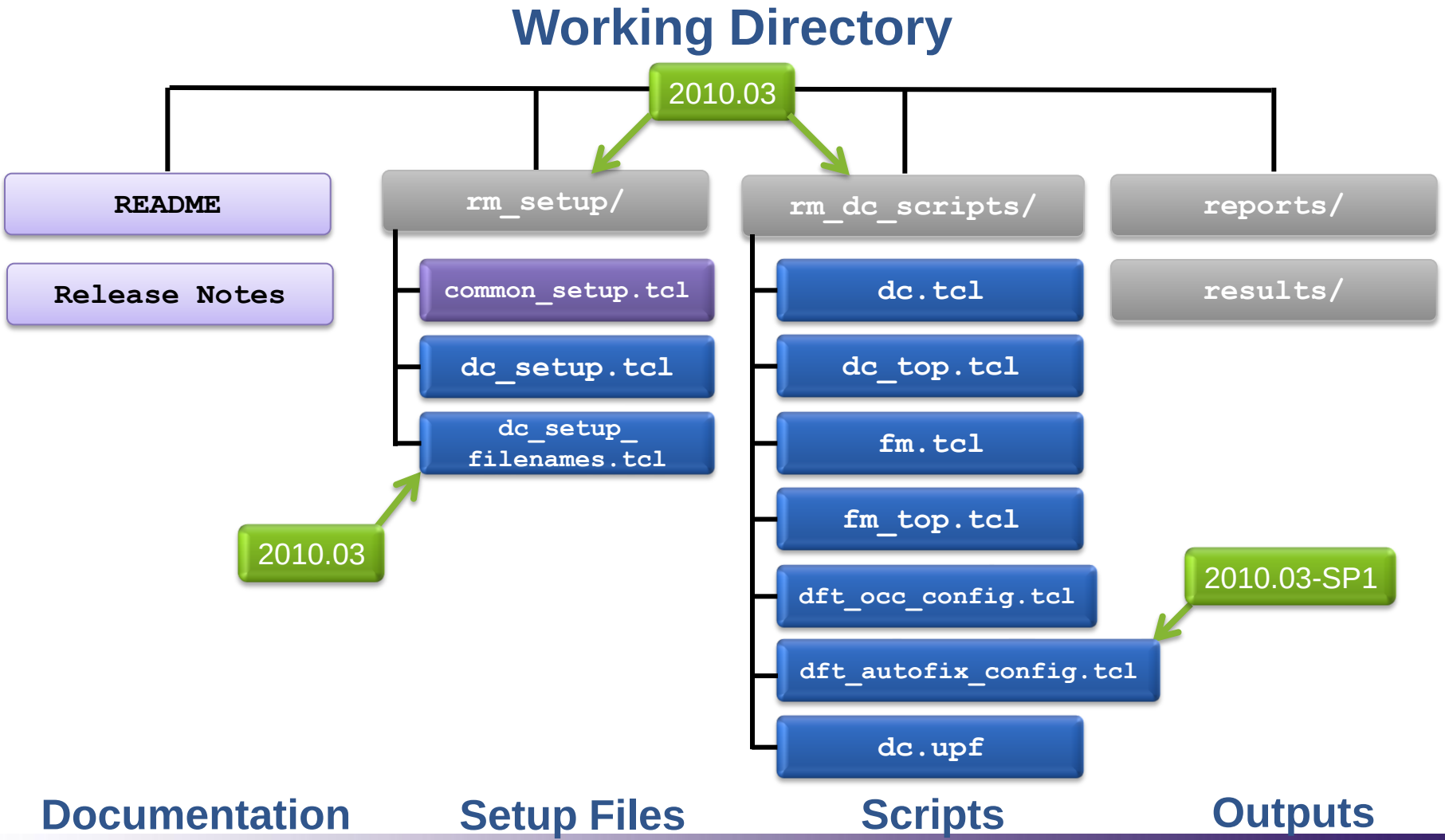
- Overview
- Scripts and Customization
- Highlights of Recent Changes
- Roadmap

2010.03

Highlights of 2010.03

- Lynx-compatible directory structure
- Design Compiler
  - Easy customization of all filenames
  - RTL read script support
  - read_floorplan/write_floorplan support
  - Congestion map from batch mode
  - Check interface timing for ILM creation
- Formality
  - Reporting setup status
  - Analyze failing or hard verification points

DC Reference Methodology Directory Structure



2010.03

Customize all Filenames

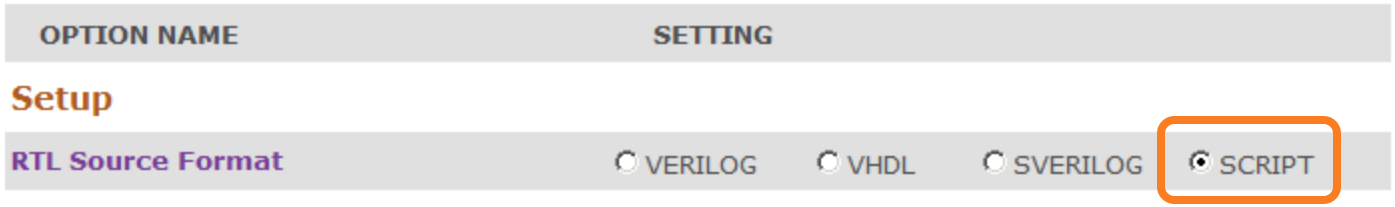
`rm_setup/dc_setup_filenames.tcl`

```
###############
# Input Files #
###############

set DCRM_SDC_INPUT_FILE                    ${DESIGN_NAME}.sdc
...
############
# Reports #
############

set DCRM_FINAL_QOR_REPORT                  ${DESIGN_NAME}.mapped.qor.rpt
set DCRM_FINAL_TIMING_REPORT               ${DESIGN_NAME}.mapped.timing.rpt
...
```

- Default names are backwards compatible with older DC-RM releases
- Variables are omitted if generated scripts don't use them
- Recommend to always source the generated `dc_setup_filenames.tcl` script then add your customizations afterwards in `dc_setup.tcl`

2010.03

RTL Read Script Support



rm_setup/dc_setup_filenames.tcl

```
set DCRM_RTL_READ_SCRIPT     ${DESIGN_NAME}.DC.read_design.tcl
set FMRM_RTL_READ_SCRIPT     ${DESIGN_NAME}.FM.read_design.tcl
```

- Many users already have a script to read the RTL in DC and Formality
- Useful for mixed Verilog/VHDL designs

2010.03

read_floorplan Support

```
if {[shell_is_in_topographical_mode]} {

# Apply Physical Design Constraints

  # set_fuzzy_query_options -hierarchical_separators {/ _ .} \
  #                         -bus_name_notations {[] __ ()} \
  #                         -class {cell pin port net} \
  #                         -show

if {[file exists [which ${DCRM_DCT_DEF_INPUT_FILE}]]} {
    extract_physical_constraints ${DCRM_DCT_DEF_INPUT_FILE}
  }

if {[file exists [which ${DCRM_DCT_FLOORPLAN_INPUT_FILE}]]} {
    read_floorplan ${DCRM_DCT_FLOORPLAN_INPUT_FILE}
  }

if {[file exists [which ${DCRM_DCT_PHYSICAL_CONSTRAINTS_INPUT_FILE}]]} {
    set_app_var fuzzy_matching_enabled true
    source -echo -verbose ${DCRM_DCT_PHYSICAL_CONSTRAINTS_INPUT_FILE}
    set_app_var fuzzy_matching_enabled false
  }
}
```

2010.03

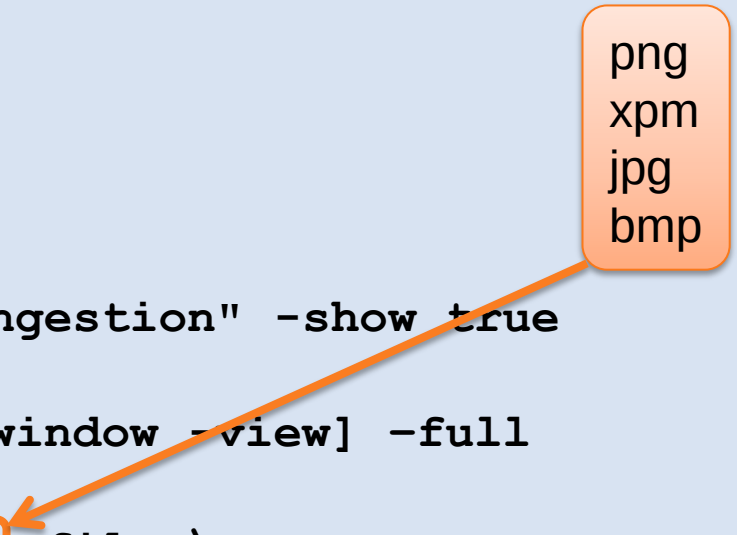
write_floorplan Support

```
##########################################################################
# Write out Design Data
##########################################################################

if {[shell_is_in_topographical_mode]} {

  write_floorplan -all ... ${DCRM_DCT_FINAL_FLOORPLAN_OUTPUT_FILE}

  write_parasitics -output ... ${DCRM_DCT_FINAL_SPEF_OUTPUT_FILE}

  write_sdf ... ${DCRM_DCT_FINAL_SDF_OUTPUT_FILE}
}
write_sdc -nosplit ... ${DCRM_FINAL_SDC_OUTPUT_FILE}
```

- **write_physical_constraints** should no longer be used

Congestion Map from Batch Mode

```
if {[info exists env(DISPLAY)]} {
  gui_start

  report_congestion -build_map

  gui_show_map -map "Global Route Congestion" -show true

  gui_zoom -window [gui_get_current_window -view] -full

  gui_write_window_image -format png -file \
   ${REPORTS_DIR}/${DCRM_DCT_FINAL_CONGESTION_MAP_OUTPUT_FILE}

  gui_stop
}
```



- Ensure that DISPLAY UNIX environment setting is set

2010.03

Check ILM Interface Timing

```
##################################################################
# Create Design Compiler ILM for Hierarchical Flow
##################################################################

write_interface_timing -nosplit ${DCRM_FINAL_INTERFACE_TIMING_REPORT}

create_ilm

write_interface_timing -nosplit ${DCRM_FINAL_ILM_INTERFACE_TIMING_REPORT}


compare_interface_timing \
        ${DCRM_FINAL_INTERFACE_TIMING_REPORT} \
        ${DCRM_FINAL_ILM_INTERFACE_TIMING_REPORT} \
        -output ${DCRM_FINAL_ILM_CHECK_INTERFACE_TIMING_REPORT} \
        -nosplit -sort_by_worst

write -format ddc -hierarchy -output \
       ${RESULTS_DIR}/${DCRM_FINAL_ILM_DDC_OUTPUT_FILE}
```

- Complete info: https://solvnet.synopsys.com/retrieve/029724.html

2010.03

Formality: Report Setup Status

```
##############################################################
# Report design statistics, design read warning messages,...
##############################################################

# report_setup_status

##############################################################
# Match compare points and report unmatched points
##############################################################

match
report_unmatched_points > ...${FMRM_UNMATCHED_POINTS_REPORT}

##############################################################
# Verify and Report
##############################################################

if { ![verify] }  { ...
```

Check all setup before running time consuming operations: match & verify

2010.03

Formality: Analyze Points

```
##########################################################
# Verify and Report
##########################################################

if { ![verify] }  {
  save_session -replace   .../${FMRM_FAILING_SESSION_NAME}
  report_failing_points >.../${FMRM_FAILING_POINTS_REPORT}
  report aborted >        .../${FMRM_ABORTED_POINTS_REPORT}
  analyze_points -all >   .../${FMRM_ANALYZE_POINTS_REPORT}
}
```

- Runs heuristic analysis on failing or hard verification points
- Provides next step suggestions to help debug the issue

2010.03-SP1

Highlights of 2010.03-SP1

- Design Compiler
  - Snapshot for Consistency Checker
  - Design Compiler Graphical Physical Guidance to IC Compiler
- Power Compiler
  - Multithreshold voltage constraint
- DFT Compiler
  - Autofix configuration file

Consistency Checker Support

2010.03-SP1

```
##########################################################################
# Save the compile environment snapshot for Consistency Checker
##########################################################################

# write_environment -consistency -output \
                    ${REPORTS_DIR}/${DCRM_CONSISTENCY_CHECK_ENV_FILE}

##########################################
# Compile the Design
##########################################
compile_ultra -scan -gate_clock
```

Also enable this right before the **place_opt** command in the following ICC-RM script: **place_opt_icc.tcl**

- Use the Consistency Checker utility to identify DCT/ICC environment differences that can contribute to correlation mismatches
- Complete info: https://solvnet.synopsys.com/retrieve/026366.html

Physical Guidance

2010.03-SP1

```
##############################
# Compile the Design
#
# Recommended Options:
...
#     -spg
#
# Use the -spg option to enable Design Compiler Graphical
# to save physical guidance information and pass this
# information to IC Compiler...
...
#########################################################

# compile_ultra -scan -gate_clock -spg -check_only

compile_ultra -scan -gate_clock -spg
```

Select the “Physical Guidance” configuration on SolvNet for both the DC-RM and the ICC-RM.

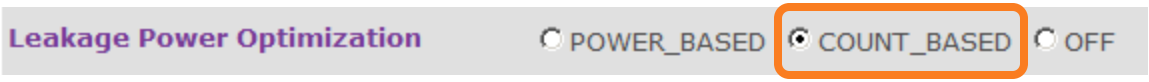
2010.03-SP1

Multithreshold Voltage Constraint

```
##################################################
# Power Optimization Section
##################################################
...
if {[shell_is_in_topographical_mode]} {
  # For limiting the number of low Vth cells in t
  # set a multithreshold voltage constraint...

# Remove any incompatible max_leakage_power or max_dynamic_power constraints
  remove_attribute [current_design] max_leakage_power
  remove_attribute [current_design] max_dynamic_power

  # Replace the following to set the threshold voltage groups in the libraries

  # set_attribute <my_hvt_lib> default_threshold_voltage_group HVT -type string
  # set_attribute <my_lvt_lib> default_threshold_voltage_group LVT -type string

  # Modify the following to set the multithreshold voltage constraint

  # set_multi_vth_constraint -lvth_groups { LVT } -lvth_percentage <% value>
}
```

Also put these same settings in the following ICC-RM scripts:

place_opt_icc.tcl
clock_opt_psyn_icc.tcl
route_opt_icc.tcl



DFT Autofix Configuration

2010.03-SP1

- New file: **dc.dft_autofix_config.tcl**
- Example settings for configuring DFT Autofix
- Uncomment and modify file if Autofix is needed

```
##############################################################
# DFT AutoFix Configuration
##############################################################

# Please refer to the DFT Compiler Scan User Guide, Chapter 7,
# "Advanced DFT Architecture Methodologies", "Using AutoFix" section.

# Please refer to the dc.dft_autofix_config.tcl file included with the
# Design Compiler Reference Methodology scripts for an example of a
# design-specific AutoFix configuration.

    # Source the design-specific AutoFix setup file
    # source -echo -verbose ${DCRM_DFT_AUTOFIX_CONFIG_INPUT_FILE}
```

Design Compiler Reference Methodology Training

- Overview
- Scripts and Customization
- Highlights of Recent Changes
- Roadmap

DC RM Roadmap

RM Capabilities	2007.03 -SP	2007.12	2007.12-SP1	2007.12-SP2	2008.09-SP4	2009.06 & 2009.06-SP1
•Baseline RM (2007.03) •DFT with Scan Compression	√	√	√	√	√	√
•Multivoltage + MTCMOS	√	√	√	√	√	√
•Formality RM		√	√	√	√	√
•Hierarchical Synthesis			√	√	√	√
•UPF Based Methodology •Congestion Optimization				√	√	√
•UPF Based Hierarchical Methodology					√	√
•DFT Enhancements: – High X-tolerant adaptive scan compression – Low power gating – On-chip clocking						√

DC RM Roadmap (cont..)

RM Capabilities	2010.03	2010.03-SP1	2010.03-SP*
•DCU/DCG Enhancements – Floorplan reading and writing updates – Timing high-effort incremental optimization – Congestion map snapshot •Low-Power Enhancement: SDC v1.8 support •Formality Enhancement: Analysis for failing and aborted pins	√	√	√
•DCU/DCG Enhancements – Synopsys Physical Guidance support – Consistency check for DC/ICC correlation •DFT Enhancement: Support for autofix •Low-Power Enhancement: %LVt count based optimization		√	√
•DCU/DCG Enhancements – Floorplan editing – Pipeline retiming reference flow – MCMM support •DFT Enhancements – Pin-limited test – Control of isolation signals for multivoltage flow – Matching DFT partitions to power domains for MV flow – Test by power domains •Low-Power Enhancements – Improved IEEE 1801™ multivoltage hierarchical flow – Export design environment for MVRC verification – Leakage-only support for MCMM			√

DC RM Roadmap (cont..)

RM Capabilities	2010.12
•DCU/DCG Enhancements – Support for Block ECO Flow (optimizable ILMs) – Feasibilty flow •Low-Power Enhancements – Support for instance-based clock-gating	√

Predictable Success