DESIGN COMPILER TOPOGRAPHICAL COMMANDS

Invoking Design Compiler in Topographical Mode	
unix% dc_shell –topographical	# Interactive shell
dc_shell-topo> start_gui	# Open GUI from shell
dc_shell-topo> stop_gui	# Close GUI, return to shell
unix% design_vision –topographical	# Open GUI directly
unix% dc_shell -topo –f dc.tcl | tee –i dc.log	# Batch mode

Helpful UNIX-like DC-shell commands
pwd; cd; ls;
history; !!; !7; !report
sh <UNIX_command>
printenv
get_unix_variable <UNIX_variable>

Settings Applied at Startup
These settings can all be included in the .synopsys_dc.setup file, OR,
they can be split up into “commom_setup.tcl” and “dc_setup.tcl” files, which are
then sourced by the “run script”, called “dc.tcl”, as done by the “Reference
Methodology” generated scripts
set ADDITIONAL_SEARCH_PATH “./libs/sc/LM ./rtl ./scripts”
set TARGET_LIBRARY_FILES sc_max.db
set ADDL_LINK_LIBRARY_FILES IP_max.db
set SYMBOL_LIBRARY_FILES sc.sdb
set MW_DESIGN_LIB MY_DESIGN_LIB
set MW_REFERENCE_LIB_DIRS “./libs/mw_lib/sc ./libs/mw_libs/IP”
set TECH_FILE ./libs/tech/cb13_6m.tf
set TLUPLUS_MAX_FILE ./libs/tlup/cb13_6m_max.tluplus
set MAP_FILE ./libs/tlup/cb13_6m.map
set_app_var search_path "$search_path $ADDITIONAL_SEARCH_PATH"
set_app_var target_library $TARGET_LIBRARY_FILES
set_app_var link_library "* $target_library $ADDL_LINK_LIBRARY_FILES"
set_app_var symbol_library $SYMBOL_LIBRARY_FILES
set_app_var mw_reference_library $MW_REFERENCE_LIB_DIRS
set_app_var mw_design_library $MW_DESIGN_LIB
get_app_var -list -only_changed_vars *
Only create new Milkyway design library if it doesn't already exist

```
if {![file isdirectory $mw_design_library ]} {
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 -tech2itf_map $MAP_FILE
history keep 200
set_app_var alib_library_analysis_path ../ ;# ALIB files
define_design_lib WORK –path ./work
set_svf <my_filename.svf>
set sh_enable_page_mode false
suppress_message {LINT-28 LINT-32 LINT-33 UID-401}
set alib_library_analysis_path [get_unix_variable HOME]
alias h history
alias rc “report_constraint -all_violators”
```

TCL Commands and Constructs

```
set PER 2.0
echo $PER # Result: 2.0
set MARG 0.95
expr $PER * $MARG # expr: *, /, +, -, >, <, =, <=, >=
set pci_ports [get_ports A]
set pci_ports [get_ports “Y??M Z*”]
echo “Effctv P = [expr $PERIOD * $MARGIN]”
# Result with soft quotes: “Effctv P = 1.9”
echo {Effctv P = [expr $PERIOD * $MARGIN]}
# Result with hard quotes:
# “Effctv P = [expr $PERIOD * $MARGIN]”
# Tcl Comment line
set COMMENT in_line ; # Tcl inline comment
set MY_DESIGNS “A.v B.v Top.v”
foreach DESIGN $MY_DESIGNS {
read_verilog $DESIGN}
for {set i 1} {$i < 10} {incr i} {
read_verilog BLOCK_$i.v}
Getting Help in DC shell
help –verbose *clock # Lists command options
create_clock –help # Lists command options
man create_clock # Displays command man page
printvar *_library # Lists variable and values
man target_library # Displays variable man page
```

More Help
Using DC “man” from UNIX prompt unix% alias dcman “/usr/bin/man \ –M $SYNOPSYS/doc/syn/man” unix% dcman target_library

Design Constraints
reset_design ############## CLOCKS ############################### create_clock -period 2 –name Main_Clk [get_ports Clk1] create_clock –period 2.5 –waveform {0 1.5} [get_ports Clk2] create_clock –period 3.5 –name V_Clk; # VIRTUAL clock create_generated_clock –name DIV2CLK -divide_by 2 -source [get_ports Clk1] \ [get_pins I_DIV_FF/Q] set_clock_uncertainty –setup 0.14 [get_clocks *] set_clock_uncertainty –setup 0.21 –from [get_clocks Main_Clk] –to [get_clocks Clk2] set_clock_latency –max 0.6 [get_clocks Main_Clk] set_clock_latency –source –max 0.3 [get_clocks Main_Clk] set_clock_transition 0.08 [get_clocks Main_Clk] ################ I/O TIMING ######################## set_input_delay -max 0.6 -clock Main_Clk [all_inputs] set_input_delay –max 0.3 –clock Clk2 –clock_fall –add_delay [get_ports “B E”] set_input_delay -max 0.5 -clock –network_latency_included V_Clk [get_ports “A C F”] set_output_delay -max 0.8 -clock –source_latency_included Main_Clk [all_outputs] set_output_delay -max 1.1 -clock V_Clk [get_ports “OUT2 OUT7] ################ ENVIRONMENT ###################### set_max_capacitance 1.2 [all_inputs] set_load 0.080 [all_outputs] set_load [expr [load_of slow_proc/NAND2_3/A] * 4] [get_ports OUT3] set_load 0.12 [all_inputs] set_input_transition 0.12 [remove_from_collection [all_inputs][get_ports B]] set_driving_cell –lib_cell FD1 –pin Q [get_ports B] set_operating_conditions –max WCCOM ############## CLOCK TIMING EXCEPTIONS ######## set_clock_group –logically_exclusive | -physically_exclusive | -asynchronous \ –group CLKA –group CLKB set_false_path -from [get_clocks Asynch_CLKA] -to [get_clocks Asynch_CLKB] set_multicycle_path –setup 4 –from –from A_reg -through U_Mult/Out –to B_reg set_multicycle_path –hold 3 –from –from A_reg -through U_Mult/Out –to B_reg ############## OTHERS ################################### set_isolate_ports –type inverter [all_outputs]

```
set_register_replication –num_copies 3 [get_cell B_reg]
set register_replication_naming_style "%s_rep%d“
set_scan_configuration \
-style <multiplexed_flip_flop | clocked_scan | lssd | aux_clock_lssd>
```

Checking and Removing Constraints and Directives
report_clock; report_clock -skew
report_design
report_port –verbose
report_wire_load
report_path_groups
report_timing_requirements –ignored
report_auto_ungroup
report_isolate_ports
report_interclock_relation
write_script –output <constraints.tcl>
report_physical_constraints
check_timing
reset_path –from FF1_reg
remove_clock
remove_clock_transition
remove_clock_uncertainty
remove_input_delay
remove_output_delay
remove_driving_cell
reset_physical_constraints

Misc. Reports
Generate A library report file
read_db library_file.db
list_libs
redirect –file reports/lib.rpt {report_lib <libname>}
report_hierarchy [-noleaf]
Arithmetic implementation and
resource-sharing info
report_resources
List area for all cells in the design
report_cell [get_cells –hier *]

Syntax Checking

```
Unix% dcprocheck constr_file.con

Physical Constraints
set_aspect_ratio 0.5 # Height/Width
set_utilization 0.7
set_port_side {R} Port_N
set_placement_area -coordinate {0 0 600 400}
set_port_location -coordinate {0 40} PortA
set_cell_location -coordinate {400 160} –fixed -orientation {N} RAM1
create_placement_blockage -name Blockage1 -coordinate {350 110 600 400}
create_die_area -polygon {{0 0} {0 400} {200 400} {200 200} {400 200} {400 0} {0 0}}
create_bounds -name “b1" -coordinate {100 100 200 200} INST_1
create_site_row -coordinate {10,10} -kind CORE -space 5 -count 3 {SITE_ROW#123}
create_net_shape -type wire -net VSS -origin {0 0} -length 10 -width 2 -layer M1
create_wiring_keepouts -name "my_keep1" -layer "METAL1" -coord {12 12 100 100}
report_physical_constraints
```

Conservative Output Load Algorithm

```
Used for “load budgeting” if actual output load values are
not known. Finds the largest max_capacitance value
in the library and applies that value as a conservative
output load
set LIB_NAME ssc_core_slow
set MAX_CAP 0
set OUTPUT_PINS [get_lib_pins $LIB_NAME/*/* \
-filter "direction == 2"]
foreach_in_collection pin $OUTPUT_PINS {
set NEW_CAP [get_attribute $pin max_capacitance]
if {$NEW_CAP > $MAX_CAP} {
set MAX_CAP $NEW_CAP
}
}
set_load $MAX_CAP [all_outputs]
```

Run Script
(called “dc.tcl” in the RM scripts)

```
read_verilog {A.v B.v TOP.v} or
read_vhdl {A.vhd B.vhd TOP.vhd} or
read_ddc MY_TOP.ddc or
analyze –format verilog {A.v B.v TOP.v}
elaborate MY_TOP –parameters “A_WIDTH=8, B_WIDTH=16”
current_design MY_TOP
link
if {[check_design] ==0} {
echo “Check Design Error”
exit # Exits DC if a check-design error is encountered
} # Continue if NO problems encountered
write –f ddc –hier –out unmappedd/TOP.ddc
redirect –tee –file reports/precompile.rpt {source –echo -verbose TOP.con}
redirect –append –tee –file reports/precompile.rpt {check_timing}
source <Physical_Constraints_TCL_file> or # Source tcl constraints, if available, or
extract_physical_constraints <DEF_file> # Extract and apply from an existing
# DEF floorplan file
group_path -name CLK1 -critical_range <10% of CLK1 Period> –weight 5
group_path -name CLK2 -critical_range <10% of CLK2 Period> –weight 2
group_path –name INPUTS –from [all_inputs]
group_path –name OUTPUTS –to [all_outputs]
group_path –name COMBO –from [all_inputs] –to [all_outputs]
set_fix_multiple_port_nets –all –buffer_constants
************************************************************
* “Compile Flow” commands go here – see next page
************************************************************
check_design
report_constraint –all_violators
report_timing [ -delay <max | min> ]
[ -to <pin_port_clock_list> ]
[ -from <pin_port_clock_list> ]
[ -through <pin_port_list> ]
[ -group]
[ -input_pins ]
[ -max_paths <path_count> ]
[ -nworst <paths_per_endpoint_count >]
[ -nets ]
[ -capacitance ]
[ -significant_digits <number>]
report_qor
set verilogout_no_tri true
```

```
change_names –rule verilog –hier
write –f verilog –hier –out mapped/TOP.v
write –f ddc –hier –out mapped/TOP.ddc
write_sdc TOP.sdc
write_scan_def –out TOP_scan.def
exit
```

Object Retrieval and Manipulation (Collection Commands)

```
get_ports, get_pins, get_designs
get_cells, get_nets, get_clocks
get_nets –of_objects [get_pins FF1_reg/Q]
get_libs <lib_name>
get_lib_cells <lib_name/cell_names>
get_lib_pins <lib_name/cell_name/pin_names>
all_inputs, all_outputs, all_clocks, all_registers
all_connected
all_fanin, all_fanout
all_ideal_nets
set pci_ports [get_ports pci_*]
echo $pci_ports #  _sel184
query_objects $pci_ports #  {pci_1 pci_2 ...}
get_object_name $pci_ports #  pci_1 pci_2 ...
sizeof_collection $pci_ports #  37
set pci_ports [add_to_collection $pci_ports \
[get_ports CTRL*]]
set all_inputs_except_clk [remove_from_collection \
[all_inputs] [get_ports CLK]]
compare_collections
index_collection
sort_collection
foreach_in_collection my_cells [get_cells -hier * \
-filter “is_hierarchical == true”] {
echo “Instance [get_object_name $cell] is hierarchical”
}
# Filtering operators: ==, !=, >, <, >=, <=, =~, !~
filter_collection [get_cells *] “ref_name =~ AN*”
get_cells * -filter “dont_touch == true”
get_clocks * -filter “period < 10”
# List all cell attributes and redirect output to a file
redirect –file cell_attr \
{list_attributes –application –class cell}
```

```
# Grep the file for cell attributes starting with dont_
UNIX% grep dont_ cell_attr | more
# List the value of the attribute dont_touch
get_attribute <cell_name> dont_touch
# Example: Identify glue cells in the current design
set GLUE_CELLS \
[get_cells * -filter “is_hierarchical == false”]
Syntax Checking
Unix% dcprocheck run_script.scr
```

Compile Flow

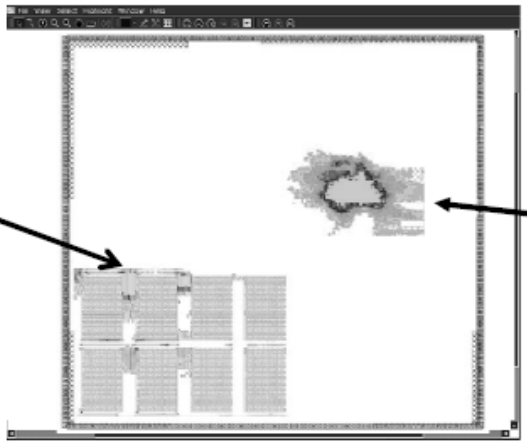
```
# Licenses required to take advantage of all Design Compiler optimization
# features: DC-Ultra, DesignWare, DC-Extension, DFTC
# Enable multi-core optimization, if applicable:
set_host_options –max_cores <#>
# Prevent specific sub-designs from being ungrouped:
set_ungroup <top_level_and/or_pipelined_blocks> false
# If needed: Disable boundary optimization on specific sub-designs:
set_boundary_optimization <cells or designs> false
# If needed: Exclude specific cells/design from adaptive retiming (-retime)
set_dont_retime <cells_or_designs> true
# If needed: Maintain registered pipeline outputs if required
set_dont_retime [get_cells U_Pipeline/R12_reg*] true
# Enable register retiming if your design(s) contain(s) pure pipelines:
set_optimize_registers true –design My_Pipeline_Subdesign
# First compile: Disable optimizations as needed with “-no_” options:
compile_ultra –scan –retime –timing [-no_boundary] \
[-no_autoungroup] [-no_design_rule] \
[-no_seq_output_inversion] [-congestion]
report_constraint –all
report_timing
report_congestion
# Apply more focus on violating critical paths, as necessary
group_path –name <group_name> -from <path_start> -to <path_end> \
–critical range <10% of max delay goal> -weight 5
# Perform an incremental ultra compile:
compile_ultra –scan –timing –retime –incremental [-congestion]
```

DFT Flow

```
# Prior to the first compile set the
# scan cell style
set_scan_configuration –style ..
# Perform the first test-ready compile
compile_ultra –timing –retime -scan …
# Before the next compile:
# Read in the scan specification file
# (See “Some Scan Specification Commands” below)
source scan_spec.tcl
# Check for DFT rule violations
dft_drc
# Preview the scan chains
preview_dft
# Insert and optimize scan
insert_dft
compile_ultra –timing –retime –scan … –incremental
# After the final compile check the DFT QoR
# and write out the scan DEF file
dft_drc –coverage_estimate
write_scan_def -out <my_design.def>
Some Scan Specification Commands
set_scan_state test_ready ; # Not needed if using ddc format
set_dft_configuration ...
set_dft_signal ...
set_scan_path ...
set_scan_configuration ...
create_test_protocol
```

Congestion Analysis after Synthesis
Text Report: (report_congestion)
Both Dirs: Overflow = 3621 Max = 13 (1 GRCs) GRCs = 2247 (2.73%)
H routing: Overflow = 1724 Max = 13 (1 GRCs) GRCs = 1139 (1.38%)
V routing: Overflow = 1897 Max = 8 (1 GRCs) GRCs = 1108 (1.34%)

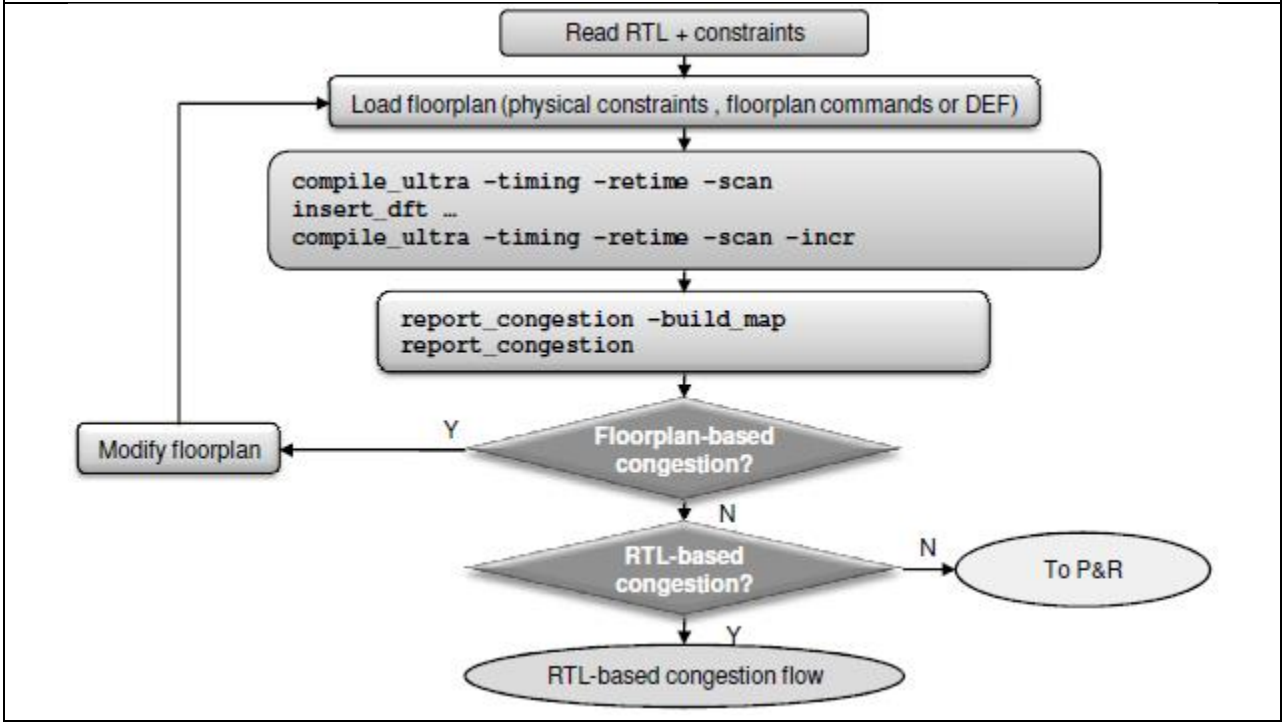
GUI Congestion map:
(Layout Window:
`View->Map Mode`)



Congestion around macro edges: Floorplan issue

Congestion in the placeable core area: Possible RTL issue

RTL-based Congestion alleviation flow



RTL-based Congestion alleviation flow