Optional -> BLUE

Steps -> YELLOW

Sub-Categories -> GREEN

#To create user define log file and invoke gui

icc2 –out <name.log> -gui

#Set CPU cores usage

set_host_options –max_cores 8

#Save block in a different name and continue working on that newly saved block name

set_app_options –name design.morph_on_save_as –value true

#Initialize Design

create_lib –technology –ref_libs

initialize_floorplan –boundary {} –shape

#Sanity Checks

get_design_checks #List all design checks

check_design –checks {dp_pre_floorplan pre_placement_stage timing physical_constraints}

check_design –checks dp_pre_floorplan #issues related to design planning before floorplanning

check_design –checks pre_placement_stage #atomic checks -> design mismatch, scan chain, mv design, timing, rp constraints

DMM – Mismatch

DFT –scan chain not defined

TCK- test clock

(TCK-001) Unconstrained endpoints -> can be due to false path, unclocked, case constant, etc.

(TCK-002) Clock pins have no fan-in -> can be due to missing clock definition, pin is blocked by case analysis, disable timing false path.

#(TCK-012) Input port with no input delay specified

#Floating pins, Multi-driven inputs/ports, undriven i/o, pin direction mismatches,

Library Integrity Check – perform consistency check b/w logical and physical library

- Cells with missing PG pins
- Cells dimensions not an integer multiple of its site
- Cells with missing LEF (MX2X1)
- Cells PG pins geometry
- Cells with missing direction

Timing

- this command will check whether the pins/ports has it's corresponding I/O delays and also checks for the clock definition exists for all flop pins.
- Checks whether the cells used in the design have been defined in the timing library
- Endpoint is unconstrainted. Reason: no arrival path, unclocked, false path, launch and capture are from different domains (TCK-001), combinational FB loop is detected (TCK-011)
- Report_disable_timing (l-loop breaking, c-case_analysis, f-false net arc) # check for which cell arc's what flag are given
- Input port has no clock_relative delay

Design (LINT checks) – validate the entire design hierarchy

- Unconstrainted_endpoints (can be clock not reaching a cell or missing output delays
- Undriven Inputs
- Undriven Ouputs
- Unloaded Nets
- Nets connected to multiple pins on same cell
- Output pins connected to PG nets
- Instances with multiple input pins tied together
- Floating Instance terminals
- Uplaced I/O pins
- Floating I/O pins

- Netlist uniqueness

Track definitions are defined

report_timing: Reports the timing informations of the current design (WLM), by default the worst setup path in each clock group.

report_qor:

Splits the design based on Scenario/Timing path group

- Levels of logic
- Critical Path's Length, Slack and it's Clk Period
- TNS
- NVP
- Worst/Total/No of Hold violations
- Cell Count – Combinational and Sequential cell count ,buff/inv count, Cell count
- Cell Area – Combinational and Sequential cell count ,buff/inv area, Cell area
- Design Rules – total no of nets/violations, Max tran, Max cap violations

Report_qor –summary

- WNS, TNS, NVE for each scenarios
- Cell area
- Nets with DRC violations

It reports the statistics/QoR of the current design viz, it's timing info, cell count, details like combinational and non- combinational elements, total area of the current design. This will also reports any DRV's present.

report_constraint:

- Setup/Hold violations based on the clock groups and predefined timing paths (reg2reg, in2reg, in2out, reg2out, clock_gating, async)
- Max Tran & Max Cap violations

It reports constraints informations/violations in the current design such as WNS, total negative slacks, DRC violations etc. The report includes whether or not the constraints are violated: by how much it is violated and the worst violating object.

check_design –checks dp_pre_floorplan

Tech file is correct

Layer directions are set or not

check_design –checks dp_pre_create_placement_abstract

Checks is constraint (SDC, UPF) files is specified

check_design –checks dp_pre_block_shaping

Checks if core area and block boundaries are valid

check_design –checks dp_pre_power_insertion

#Preferred metal layers are defined

#No preferred routing direction defined for layers

check_timing #undefined clocks, undefined constraints, gated_clock, generated_clocks, unconstrainted_endpoints (can be clock not reaching a cell or missing output delays) false path,

check_library #

report_constraint (setup, hold, max trans, max cap)

#Mega checks are dp_pre_floorplan pre_placement_stage, pre_clock_tree_stage & pre_route_stage

#ZIC

set_app_options –name time.delay_calculation_style –value zero_interconnect

report_timing –slack_lesser_than 0

report_qor –summary (WNS, TNS, NVE)

report_timing –report_by _scenarios “S1 S2 S3”

#Check Timing Constraints

check_timing

```
foreach_in_collection mode [all_modes] {
current_mode $mode
report_exceptions
report_case_analysis
report_disable_timing
}
#Set metal layers and direction for design
set_ignored_layers -max_routing_layer M8 -min_routing_layer M2
set_attribute [get_layers M1 M3 …] routing_direction horizontal
set_attribute [get_layers M2 M4 …] routing_direction vertical
# In layer M1 and layers above M8 RC and congestions estimation is ignored and it also prevents routing above M8 and below M2

#Hierarchy Color and Hierarchy Exploration
set_colors –cycle_color
#View->Assistants->Hierarchy Exploration (OR)
explore_logic_hierarchy –create_module_boundary

#Manually create voltage area for MV design
create_voltage_area –power_domain –region
set_voltage_area <PD_RISC_CORE> -is_fixed
#Automatic voltage area creation
shape_blocks
#define hard keep out margin (even for LS and ISOcells)
```

set_shaping_options –guard_band_size 10

#UPF

ICC2 is MV aware. It will create a default UPF -> a SN, SP & CSN

#Floorplan

#Set macro placement options

plan.place*

plan.macro.*

plan.macro.auto_macro_array (default is true)

plan.macro.spacing_rule_heights

plan.macro.spacing_rule_widths

(OR)

plan.macro.min_macro_keepout #Set minimum macro spacing

create_placement -floorplan

(OR) Do manual macro placement

#Create automatic soft/hard blockages by controlling the widths of blockages:

plan.place.auto_generate_blockages

plan.place.atuo_generate_blockages_soft_blockages_channel_width

plan.place.auto_generate_blockages_hard_blockages_channel_width

#Fix/Unfix the Macros

set_fixed_objects [get_flat_cells –filter is_hard_macro==true] (or)

set_fixed_objects [get_cells –physical_context –filter design_type==macro] (or)

set_app_options –list {place.coarse.fix_hard_macros true}

set_fixed_objects [get_flat_cells –filter is_hard_macro==true] unfix

#Automatic FP with macros and std. cells

create_placement –floorplan [-congestion]

#Pin Placement (use if loose pin constraints are given or none)

set_block_pin_constrains –self –allowed_layers “M3 M4” –sides “1 2 3” |-exclude_side “4 5 6” –corner_keep_out_distance

set_individual_pin_constraints –ports

place_pins –self

#Placement Blockage/Keepout margin/Bounds

create_keepout_margin –type hard –outer {5 0 5 0} RAM5 #For one macro

create_keepout_margin –type hard –outer {5 5 5 5) [get_cells my_macro]

get_keep_out_margins

report_keep_out_margin

remove_keep_out_margins

create_placement_blockage –boundary {{ } {}} –name LL_CORNER –type hard | soft

create_bounds –name –type –boundary #Timing fixes

check_design –checks physical_constraints

#Floorplan object checks – layers, bounds, placement blockages, keepout margin, site row checks (height multiple of site height, IO pad cell overlap, macro fixed or not, macro overlapping, cell orientation

#PG routing/Pattering

set_pg_statergy

create_pg_mesh_pattern

compile_pg #create mesh

#Remove PG nets

remove_routes –stripe –net_types {power ground} #PG straps

remove_routes –lib_cell_pin_connect –net_types {power ground} #PG rails

(OR) right-click net -> selected related objects-> nets

remove_physical_objects [get_selection]

#Checks after PG routing

check_pg_drc

check_pg_missing_vias

check_pg_connectivity

#Create FP def (1st Pass Synthesis)

write_floorplan –format icc –output <name.fp> -net_types {power ground} –include_physical_status {fixed locked}

#To apply FP later

source <name.fp>/<floorplan.tcl>

#Setting up and Updating MMMC constraints

create_mode M1

create_corner C1

create_scenario –mode M1 –corner C1 –name M1_C1

write_script

current_scenario

all_scenarios

report_scenarios, report_corners, report_pvt

remove_duplicate_timing_contexts

#All scenarios are active by default, so all scenarios have to be deactivated and then activate required scenarios

set_scenarios_status * -active –false

set_scenario_status <scenario_name> -active true

set_scenario_status *corner_FAST –setup false

set_scenario_status *mode_TEST* –leakage_power false –dynamic_power false

set_scenario_status <name> -setup true –hold false –max_transition true –max_capacitance true –active true

#Find currently active scenarios

report_scenarios

(OR)

get_attribute [get_scenarios] active

(OR)

get_scenarios –filter active&&setup

get_scenarios –filter active&&hold

#Apply freeze for clock ports to avoid performing HFNS on clock nets

set_freeze_ports –clock | -data

(OR)

report_ideal_network

remove_ideal_network –all

#Fix Ports of design

get_attribute –objects [get_ports *] –name physical_status

set_fixed_objects [get_ports *]

#PLACEMENT

#Pre Placement checks

check_design –checks pre_placement_stage #atomic checks -> design mismatch, scan chain, mv design, timing, rp constraints

(TCK-001) Unconstrained endpoints -> can be due to false path, no ceck, unclocked , case constant, etc.

(TCK-002) Clock pins have no fan-in -> can be due to missing clock definition, pin is blocked by case analysis, disable timing false path.

#(TCK-012) Input port with no input delay specified

check_design –checks physical_constraints

#Floorplan object checks – layers, bounds, placement blockages, keepout margin, site row checks (height multiple of site height, IO pad cell overlap, macro fixed or not, macro overlapping, cell orientation

#Placement Setting

set_app_options –name opt.optimize_scan_chain

set_app_options -name place.coarse.continue_on_missing_scandef

set_app_options -name place.coarse.max_density -value 0 ;#default is 0, so that the tool will decide automatically

set_app_options -name place.coarse.congestion_layer_aware

set_app_options -name place.coarse.auto_density_control

set_app_options -name place.coarse.fix_hard_macros

set_app_options -name place.coarse.tns_driven_placement #Timing driven placement

set_app_options -name place.coarse.auto.timing_control #This enables the above command also

create_placement –buffering_aware_timing_driven

set_app_optionos place_opt.flow.clock_aware_placement # This also enabled when icg_optimization is turned on

#Placement Stages (place_opt)

1. initial_place # Wire length driven placement & scan chain opt.
2. initial_drc # Remove buffer trees & HFNS and logic DRC
3. initial_opto # Quick timing opt.
4. final_place # Incremental & final timing driven placement
5. final_opto # Full-scale opt.

#Set app options for placement

report_app_options {place.* place_opt*}

set_app_options –list {

place_opt.initial_drc.global_route_based 1

place_opt.congestion.effort high

place.coarse.congestion_layer_aware

place.coarse.pin_density_aware

opt.timing.effort

opt.common.max_fanout

place.coarse.congestion_driven_max_util 0.93

place_opt –from <stage_name> –to <stage_name>

#If scan chains are present in design

read_def <name.scandef>

check_scan_chain

report_scan_chains

#Set keep out margin for Std. cells

set_placement_spacing_label –name{X} –side both –lib_cells [get_lib_cells /*AOI*]

set_placement_spacing_rule –labels {X X} {0 2} #Spacing from 0 to 2 sites in prohibited

(OR)

Use set_keep_out_margin

#Find & Enable/Disable Spare Cell placement

get_cells –hier –filter is_spare_cells

place.coarse.enable_spare_cell_placement (default: true)

#Adding end cap cells

create_boundary_cells -left_boundary_cell {saed32_rvt|saed32_rvt_std/SHFILL3_RVT} -right_boundary_cell {saed32_rvt|saed32_rvt_std/SHFILL3_RVT}

set_fixed_objects [get_cells -hierarchical *boundary*]

#Adding tap cells

create_tap_cells -lib_cell {saed32_rvt|saed32_rvt_std/SHFILL3_RVT} -distance 60 -pattern stagger

set_fixed_objects [get_cells -hierarchical *tap*]

#Adding spare cells if they are not given in netlist They will automatically placed if they are given in netlist during placement stage

add_spare_cells –cell get_cells –hierarchical –filter is_spare_cell] -
boundary {} –num_instances 20 v

legalize_placement

set_placement_status legalize_only [get_flat_cells –filter _spare_cell]

#Adding IO Buffers

proc insert_io_buffer {buf} {

foreach_in_collection each_port [get_ports *] {

set port_name [get_object_name $each_port]

insert_buffer –lib_cell $buf [get_ports $port_name]

}

magnet_placement [get_ports *] –mark_fixed [get_cells $buff]

}

insert_io_buffer NBUFFX4_HVT

legalize_placement

magnet_placement –mark_legalize_only –cells [get_flat_cells *I_CONTEXT_MEM*] I_CONTEXT_MEM/I_COTEXT_RAM_3_1

#Creating Path Groups

Placing lO paths in separate path groups allows final placement work on reg-to-reg paths, independent of large violating l/O paths

foreach scenario [all_active_scenarios] {

current_scenaio $scenario

group_path –name REG2REG –from [all_registers] –to [all_registers] -weight

… }

get_path_groups

#Preroute layer optimization

set_app_options –name place_opt.flow.optimize_layers –value true

#Find purpose of lib cells in design

report_lib_cells –colums {name valid_purpose{ -objects [get_lib_cells “*/*BUFF”]

#To restrict use of certain cells in design

set_lib_cells_purpose –include node [get_lib_cells “*/*BUF_X64 or ULVT”]

#ICG timing driven clock aware placement for designs with critical ICG enable timing (occurs during initial_opto of place_opt) (splits ICGs are places them closer to cluster registers to reduce setup) -> Use if ICG cells are present in design

place_opt.flow.clock_aware_placement

set_app_options place_opt.flow.optimize_icgs true

#POWER BASED DESIGN

#Set power opt. options

set_app_options –list {opt.power.mode none|leakage|dynamic|total} #For setting in dynamic mode; you need to read SAIF (read_saif)file

#Set power opt. options for certain scenarios

set_scenario_status {func.ss_125c} –leakage_power true –dynamic_power –true

#List of all diff. types threshold cells & areas (LVT, HVT, RVT, ULVT)

report_threshold_voltage_group

#Set cells for Vt swapping

set_threshold_voltage_group_type –type low_vt “LvT ULvt”

(OR)

set_attribute –quiet [get_lib_cells */*LVT] threshold_voltage_group LVT

#Limit the use of LVT cells to limit leakage power, if it is a power based design

set_app_options –list {opt.power.mode leakage}

set_app_options –list {opt.power.leakage_type percentage_lvt}

set_max_lvth_percentage 10

#Enable AOCV Analysis (based on logic depth and distance) Less pessimistic than OCV Analysis

set_app_options –name time.aocvm_enable_analysis –value true

set_app_options –name time.aocvm_enable_distance_analysis –value true

read_ocvm –corners slow slow_derate_table

read_ocvm –corners fast fast_derate_table

#POCV -> Cumulative delay by adding delay distribution of each stage (std. deviation) -> less pessimistic than AOCV due to removal of data path pessimism in GBA

#1 POCV single coefficient file -> coefficient is characterized at particular input trans. And output load eg: 0.05 x 60ps

#2 Liberty Variation Format (LVF) -> func. Of input trans. and output load per timing arc eg: ocv_sigma_rise_transition (lvf_table) {

index1

index2

LUT…

}

set_app_options –name time.pocvm_enable_analysis –value true

set_app_options –name time.pocvm_enable_distance_analysis –value true

read_ocvm –corners slow slow_derate_table

read_ocvm –corners fast fast_derate_table

#Placement

place_opt

#The trail clock tree is maintained after place_opt

(OR) Perform Coarse Placement First

create_placement –timing_driven –congestion –congestion_effort

#then do place_opt –stages

#Post Placement checks

#

check_legality

#Analyze global route congestion

route_global –floorplan true –congestion_map_only true
#Congestion Analysis

Overflow/Supply

#Report Utilization

create_utilization_configuration –config –include -exclude

report_utilization #Default – std. cells area/site array area

route_global –congestion_map_only true –effort_level

report_qor –summary (WNS, TNS, NVE)

report_timing

report_constraints –all_violators

analyze_design_violations

#Incremental improving timing & area

refine_opt –endpoints –path_groups

#Report qor for each stage

Create_qor_snapshot –name <name> -display <name>

#Unplace std.cell

reset_placement

CTS

#ALL opts are done for all active scenarios

#For CTS make all scenarios active **(IMPORTANT)**

#PreCTS checks

check_design –checks pre_clock_tree_stage

report_net_fanout –high_fanout|threshold 50

report_net_fanout –threshold 60 [get_flat_cells –filter \ net_type==signal]

check_scan_chain

#Different stages of clock_opt command (optimization occurs on all active scenarios)

1. build_clock -> Builds timing driven GR clock trees and performs inter clock balancing to improve setup/hold
2. route_clock -> Rotues clock nets and updated latency
3. final_opto -> Optimizes the data path and clock logic simultaneously to further improve setup/hold

#Update clock latency before CTS (Not Required since clock_opt does it automatically)

set_scenario_status [all_scenarios] –active true

compute_clock_latency

#Control what cells are to used for fixing hold violations

set_lib_cell_purpose –include hold [get_lib_cells ”*/NBUFF*HVT */NBUFF*RVT”]

#Hold Fixing

set_app_options –list {clock_opt.hold.effort high}

set_app_options –list {opt.dft.clock_aware_scan true}

#Remove CRPR as it is too pessimistic

set_app_options –name time.remove_clock_reconvergence_pessimism –value true

report_crpr –from -to

#Classical CTS

set_app_options cts.compile.enable_global_route true

#Enable local skew ->During CCD these options are ON by default

set_app_options cts.compile.enable_local_skew true

set_app_options cts.optimize.enable_local_skew true #CTO

#CCD-CTS (Useful skew + Data path optimization)

#Target skew is irrelevant in CCD

#All clock ports must have a set_driving_cell or set_input_transition

#Uncertainity must also be updated before CT

#Enable CCD

set_app_options –name clock_opt.flow.enable_ccd –value true

#Set DRV values

set_max_transition <value> -clock_path [all_clocks] –scenario
default: 500ps

set_max_capacitance <value> -clock_path [all_clocks] –scenario
default: 600pF

report_clock_settings

#Set CTS cell selection

set_lib_cell_purpose –exclude cts [get_lib_cells]

set_lib_cell_purpose –include cts [get_lib_cells */clk_buff*]

#Resizing FF is also available in CCD

#Inter-Clock tree balancing is good for CCD ; w/o it tool will insert more buffers and run time is also longer

create_clock_balance_points-name group1 \

-objects [get_clocks “SDCLK SYSCLK”] (OR)

derive_clock_balacne_constraints –slack_less_than -0.3

report_clock_balance_groups

#To turn off skew adjustment for boundary registers (eg: input to reg paths) & ignore scan, reset ports for boundary identification, use the the follo. commads -> classical CTS will be performed on these registers

ccd.optimize_boundary_timing false

ccd.ignore_oprts_for_boundary_identifications

#Setting target skew, & latency

set_clock_tree_options –all –target_skew –target_latency –corners [all_corners]

report_clock_tree_options

#Allow movement of clock tree cells during clock_opt (eg: ICGs)

cts.compile.enable_cell_relocation (default: all cells)

#Applying NDR clock Rules

create_routing_rule <rule_name> -widths {M5 0.11 M6 0.14} \

-spacing {M5 0.48 M6 0.48} \

-cuts { {VIA3 {Vrect 1}} } #Use via names defined in .tf file

set_clock_routing_rules –rule <rule_name> –min_routing_layer M4 –max_routing_layer M5 –net_type sink|root|internal

report_routing_rule

#Clock cell spacing to avoid EM (formation of hot spots results in high current density)

set_clock_cell_spacing –lib_cells "*/clk_buff*" –x_spacing 2 –y_spacing 2

report_clock_cell_spacing

remove_clock_cell_spacing

#Post CTS Optimization

#Enable GRO (off by default)

set_app_options –name clock_opt.flow.enable_global_route_opt –value true

#Latency Check

Report_clock_timing –type latency

#Defining explicit ignore/exclude pin

set_clock_balance_points \

-consider_for _balancing false \

-balance_points [get_pins AND2X/A]

report_clock_balance_points

#Defining explicit sink pin

```
set_clock_balance_points  \
-consider_for _balancing true \
-balance_points [get_pins CLK/RAM_IP]
report_clock_balance_points
#Mode specific clock balance points
Foreach mode {m1 m2} {
current_mode $mode
set_clock_balance_points  \
-consider_for _balancing false \
-balance_points [get_pins AND2X/A]
report_clock_balance_points
}
report_clock_balance_points
#Setting insertion delay manually to clock pins of macros
set_clock_balance_points  \
-consider_for _balancing true \
-delay 0.12 –early \
-balance_points [get_pins CLK/RAM_IP]
report_clock_balance_points
#Reporting internal delay of clock pin
report_clock_qor –type latency –show_paths –to RAM_IP/clk
#To address degradation due to Crs Tlk and Coup Cap
synthesize_clock_trees –postroute
#Enable area recovery
```

set_app_options –name clock_opt.flow.enable_clock_power_recovery –value area|power|auto|none

#Post CTS checks

report_clock_qor (max global skew, insertion delay, DRC vio, histogram…)

report_clock_timing –type –modes -corners

report_clock_timing –type skew

#Min pulse width check

report_min_pulse_width –all_violators

#SIGNAL ROUTING

#Pre Routing Checks

check_routability (Blocked ports, out of boundary pins, min grid violations)

check_design –checks pre_route_stage

#Routing Stages

1. Global Route -> Assigns net to specific metal layers and does GR using Gcells. Uses zero width lines
2. Track Assignment -> Assigns each net to a specific track and creates actual metal shapes. Reduces no. of detours and vias
3. Detail Routing

#Routing

route_auto

route_detail –increamental true

route_opt # If required

(OR)

route_global

route_track

route_detail

#Report app options

report_app_options route.detail*

report_app_options route.common*

#Redundant via insertion (Replace single-cut with multi-cut vias or bar vias) -> Make sure no routing DRCs are there before using RVI

route.common.post_detail_route_redundant_via_insertion medium

route.detail.optimize_wire_via_effort_level high #OR use optimize_routes after route_auto

#Concurrent Antenna Fixing (jogging is enabled but default but diode insertion is not) 1# Jogging 2# Diode insertion

route_detail.antenna (default: ON)

route.detail.diode_libcell_names “DIODE1x”

route.detail.insert_diodes_during_routing true

#Timing Driven Routing

route.global.timing_driven true

route.global.timing_driven_effort_level

route.track.timing_driven true

#Enable Crs tlk prevention

route.track.crosstalk_driven true

route.detail.timing_driven true

#To fix shorts over macros

set_app_options –name route.detail_repair_shorts_over_macros_effort_level –value

route_detail –increamental true #Fix DRC violations

#Post Route checks

check_routes #DRC, shorts, opens, antenna & voltage area violations

check_lvs –checks all –open_reporting detailed #Open, shorts and floating nets

#Post route optimization

set_app_options route_opt.*

set_app_options route_opt.flow.enable_power #Leakage power opt.
set_app_options route_opt.flow.enable_ccd #CCD opt.

set_app_options route_opt.flow.enable_clock_power_recovery –value area|power|none #CCD power recovery

#Post route redundant via insertion

set_app_options route.common.post_detail_route_redundant_via_insertion medium

add_redundant_vias #Can be executed afer route_auto and also after route_opt

#Re-routing specific nets after routing is done (eco route)

Remove_routes –nets <net_name> -detail_route

route_eco –nets <net_name>

set_attribute –objects[get_nets <net_name/s] –name physical_status –value locked

#Via Prioritization (VL>VB>VS)

#Add via mapping options for redundant via insertion

MxN -> Horizontal(M)xVertical(N) directions.

add_via_mapping –transform all –from VL 1x1 –to VL 1x2 –weight 5

add_via_mapping –transform all –from VB 1x1 –to VL 1x2 –weight 5

add_via_mapping –transform all –from VS 1x1 –to VL 1x2 –weight 5

(OR) Use –from_icc_file <script.name> #define_zrt_redundant_vias

#SignOff (IC Validator)

#The current design is near timing closure and only small no. of DRCs are left

#Checks

signoff_check_drc #Specify the routing layers on which checks must be made

sign_check_design –short_with_metal_fill true #Launches IC Validator tool with a runset and creates an error data

signoff.fix_drc.init_drc_error_db <db_directory> # To specify the path of the db file generated by IC Validator

signoff_fix_drc #involkes ICC2 to fix violatons based on results from IC Validator and reruns IC Validator signoff DRC checks.

#Filler Cell Insertion

create_stdcell_fillers –lib_cells <filler_list>

connect_pg_net

#Metal Fill (Track based (automatic) or Pattern based fill)

#Unconstrained (Pre-Tape out) and Freeze Silicon (Post-Tape out) ECO ->VLSI PRO website

#Unconstrained ECO FLow

eco_netlist #Update Netlist with ECO changes & identifies changes b/w the two netlist versions

connect_pg_net

place_eco_cells- allow_move_other_cells #Place newly inserted ECO cells –displacement_threshold <distance>

displacement_threshold switch tries to place cells within the specified distance & if it cannot then it will not legalize those cells and reports the total no. of rejected cells in a collection.

route_eco #Route new nets

route_opt #Fix timing

#Freeze Silicon ECO flow

Insert and place spare cells

eco_netlist

#Manually update clock tree, scan chain, UPF

check_freeze_silicon #Check feasibility

map_freeze_silicon (M) / place_freeze_silicon (A) #Mapping newly created cells to spare cells

connect_pg_nets

connect_freeze_silicon_tie_cells #connect PG, connect tie-off pins to tie cells, if needed

route_eco #Update routing with the ECO changes

#Physically aware ECO with PT:

Input->.v, SDC, UPF, DEF, SPEF, Spacing rules. Steps for PT-ECO

1. Power Recovery #downsizing cells
2. DRC and timing fixing
3. Final leakage recovery

fix_eco_drc –open_site | -occupied_site | freeze_silicon

place_eco_cells -remove_filler_references <cell_names>

route.global.timing_driven false

route.detail.timing_driven false