Mixed signal ASIC : implementation and validation

Part 3 : assemble design and signoff checks

Bertrand GENNERET, Principal Application Engineer

7 Oct. 2022

Agenda

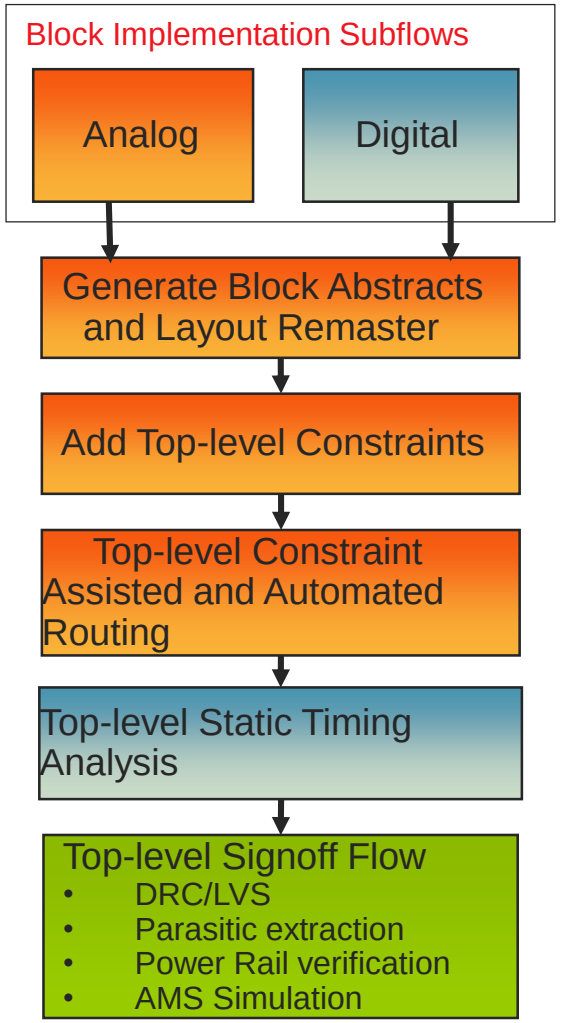
- Top level chip assemble
- Static Timing Analysis
- Crosstalk effect analysis
- Power analysis
  - Power consumption
  - Irdrop
  - Electromigration
- DRC / LVS
- Simulation
- Conclusion

Top level chip assemble

Analog on Top: implementation of top level with digital block

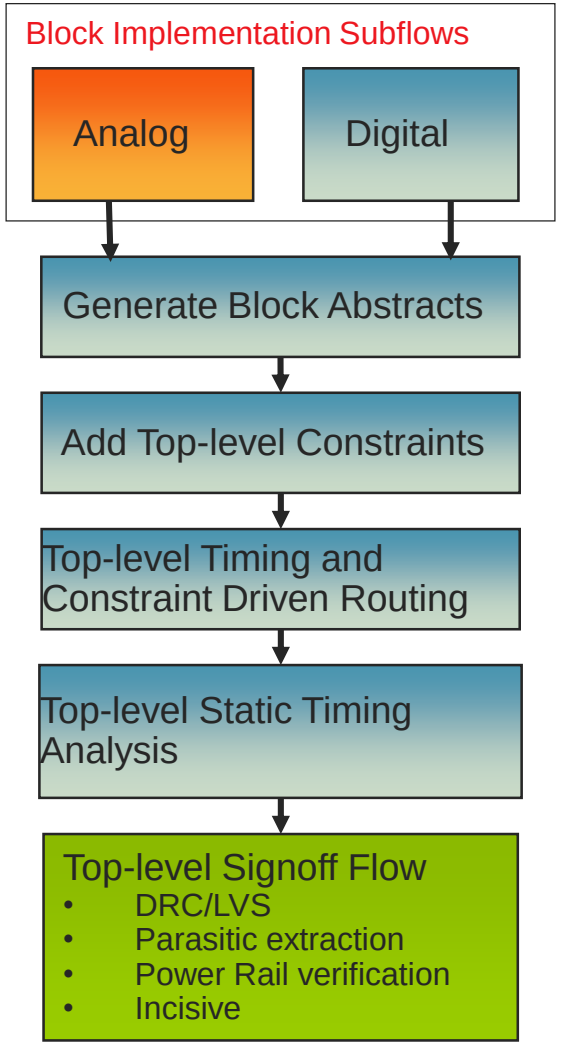
- For the implementation of top level, we will need to create an abstract for the digital block and replace it at the top level. The abstract is a footprint of the layout view and contains only the boundary, pins and blockage information of the block (refer to presentation, part 1).
- After the layout of the digital block is implemented (presentation, part 2) the next step is to create a more accurate abstract with Abstract generator with cover blockages in the block and adding antenna information to the abstract.
- Once the abstract is generated we do replace the initial softAbstract view of the digital block at the top level with the implemented abstract view in Virtuoso.
- User can then execute top level routing and DRC/LVS flows

Top level assembly flow



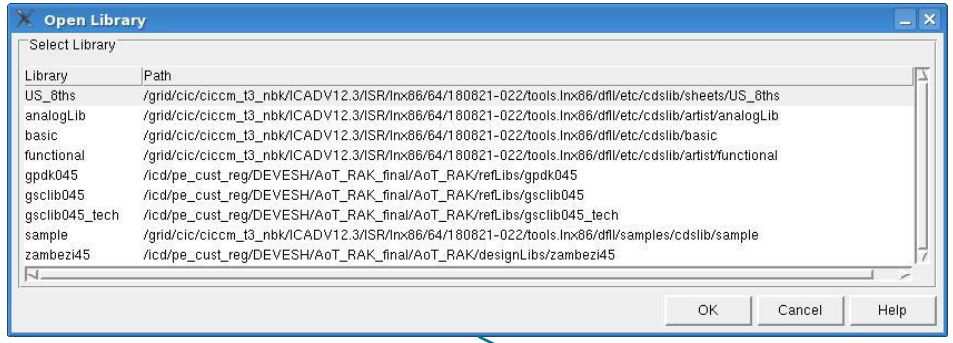
My blocks are done …
What's left ?

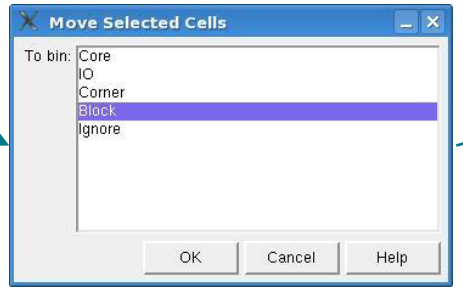
- Generate abstracts of the blocks
- Add routing constraints either in Virtuoso or Innovus
- Constraint driven VSR/CSR or timing driven NanoRoute
- Top-level STA with embedded AMS block FTM

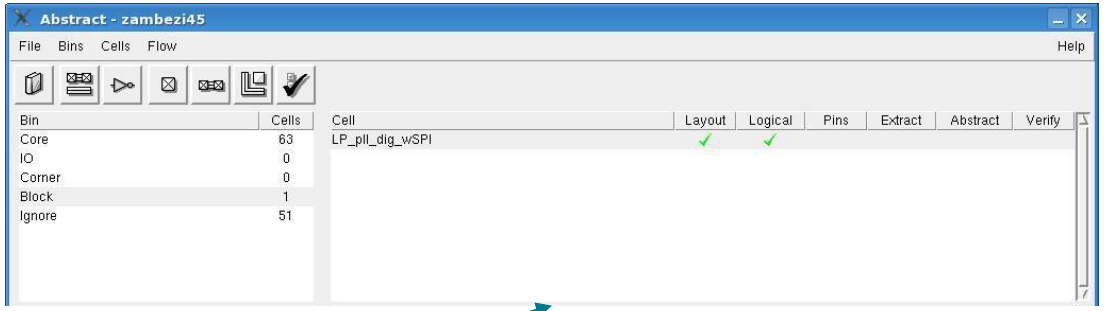


Analog on Top : detailed abstract creation

- Launch the abstract generator: abstract –log abstract.log
- Click File > Library > Open
- If necessary, select the core bin, and move the digital cell to bin Block

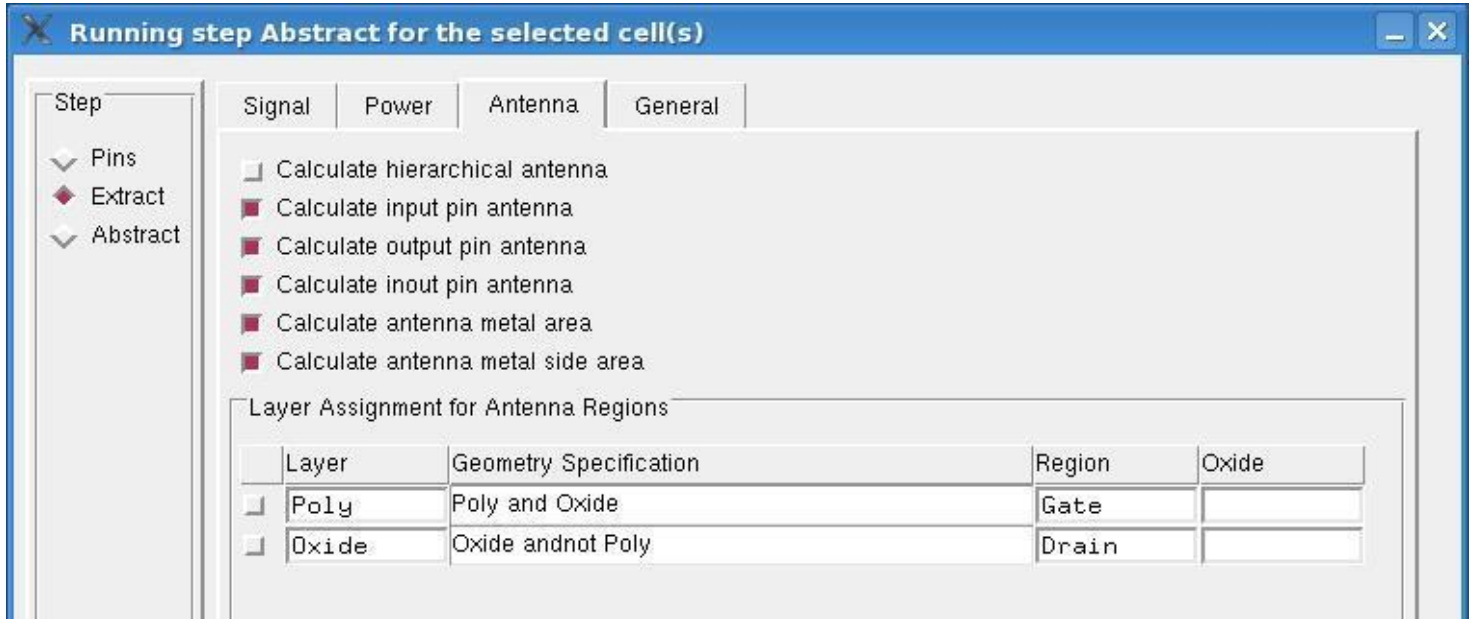






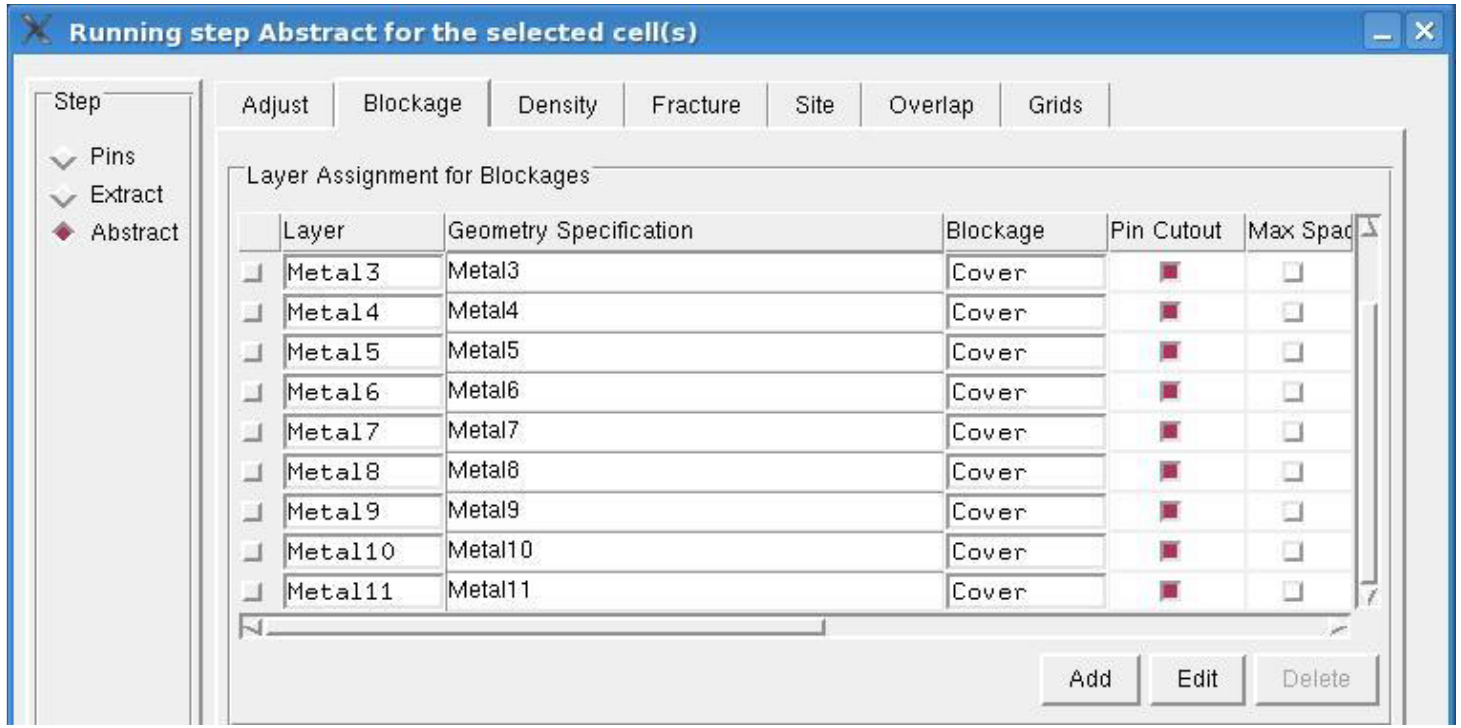
Extract antenna info

- Change to extract step and in the antenna tab you will see the options to calculate the antenna values.



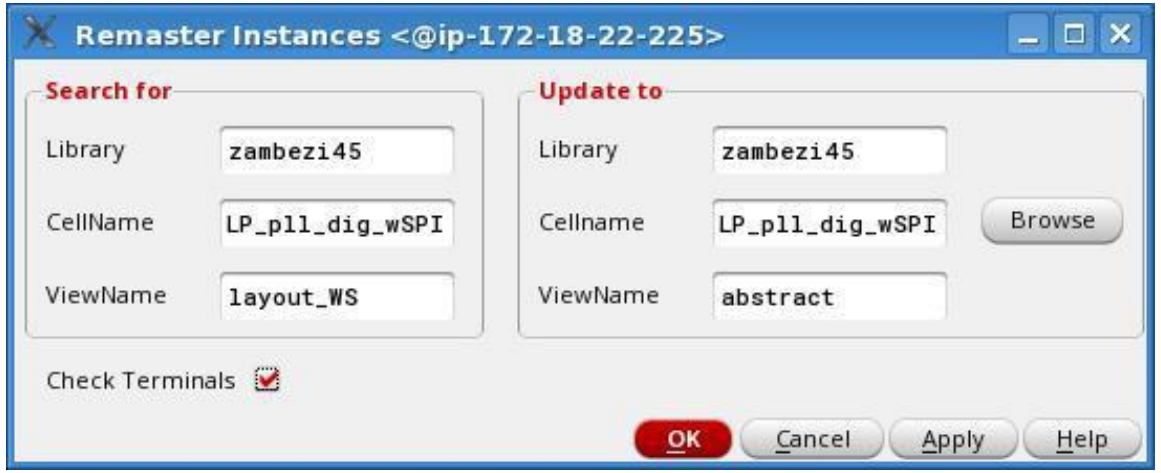
Blockage cover

- Change to the Abstract Step in the form and check the Blockage TAB. The option is to create COVER blockages for all layers and block any routing over the block when the top level is being routed.



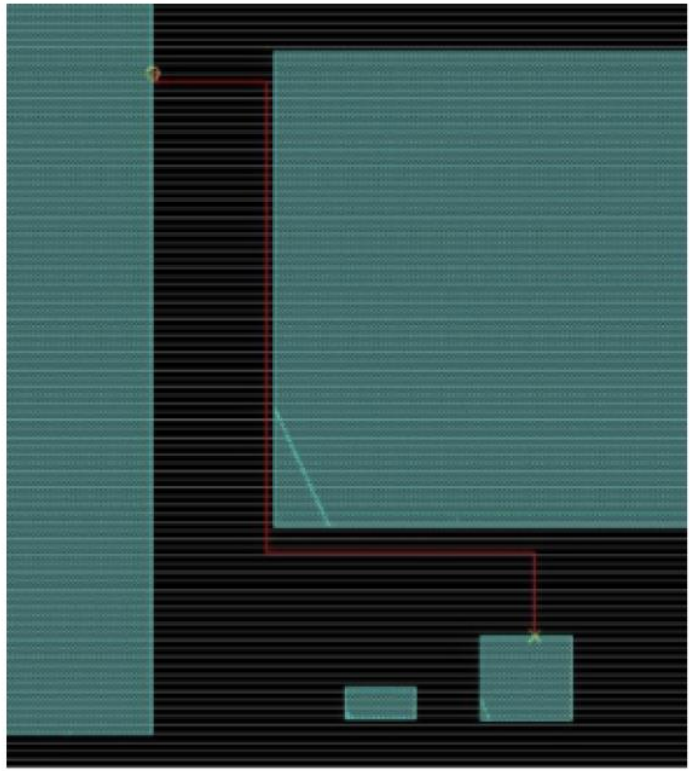
Use the abstract at top level

- Open the top level in Virtuoso XL
- Click on tools > remaster instances
- Replace the view name with the abstract created

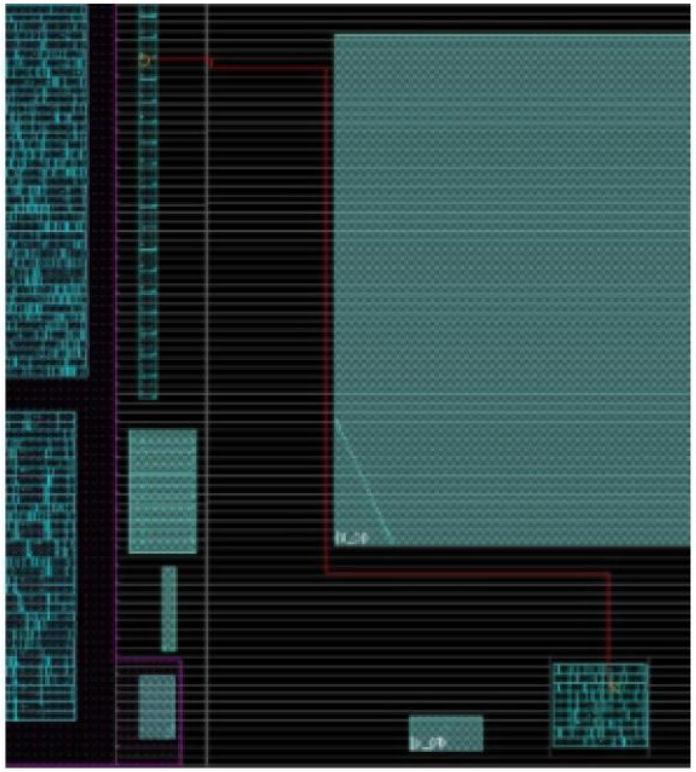


Why do we need to flatten the design for analysis?

- In many mixed-signal designs, the digital logic exists at various levels of the physical hierarchy. To perform an accurate analysis of such timing paths, the timer requires the digital logic in the lower levels of the hierarchy to be exposed for creation of the complete timing path. The digital tool should be able to trace the logical and the physical connectivity for the entire timing path.



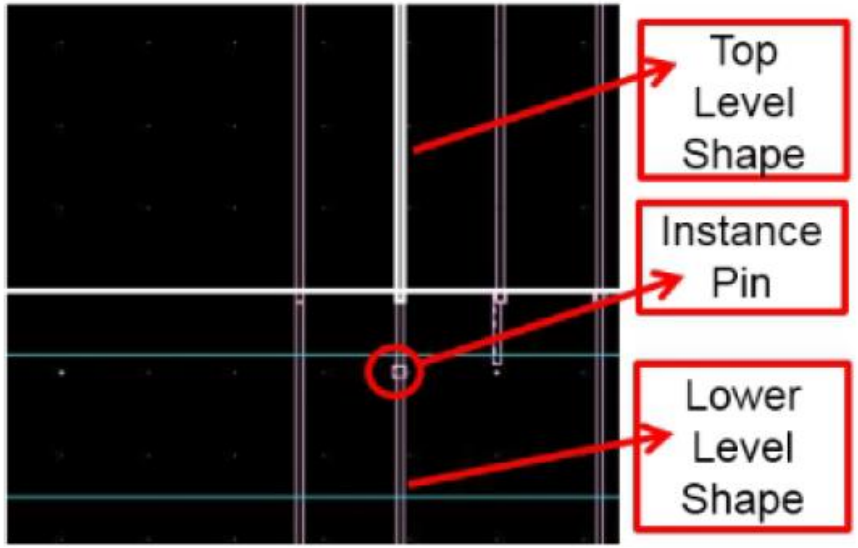
Complete path not visible after initial design load



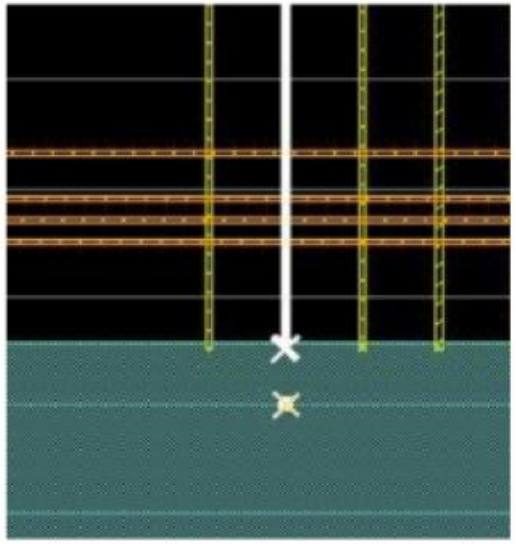
Complete path visible after assembleDesign

Requirements for Correct Connectivity Propagation

- The nets on the instance terminals should be properly connected for logical connectivity to be propagated. Mosaics should be avoided in the design because although Innovus creates the instance from the mosaic, assemble_design does not map the top-level nets to the lower-level mosaic instances.
- All cells should have shape pins on interconnect layers placed on the cell boundary.



Top level shape connected to instance pin by shape at lower level



Innovus sees this as an open

Simplified .lib VS flat design

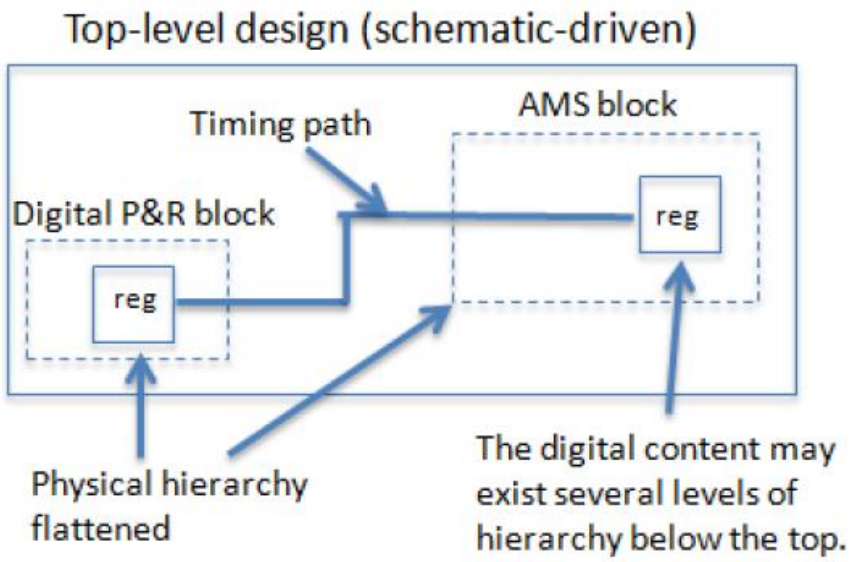
- Liberate AMS .lib can be generated for the mixed signal blocks such as
  - ADC
  - DAC
  - PLL
  - SERDES

- Flattening the design, to extract Verilog / spef for extraction is another option.
  - Avoid this solution if digital block is too depth in hierarchy

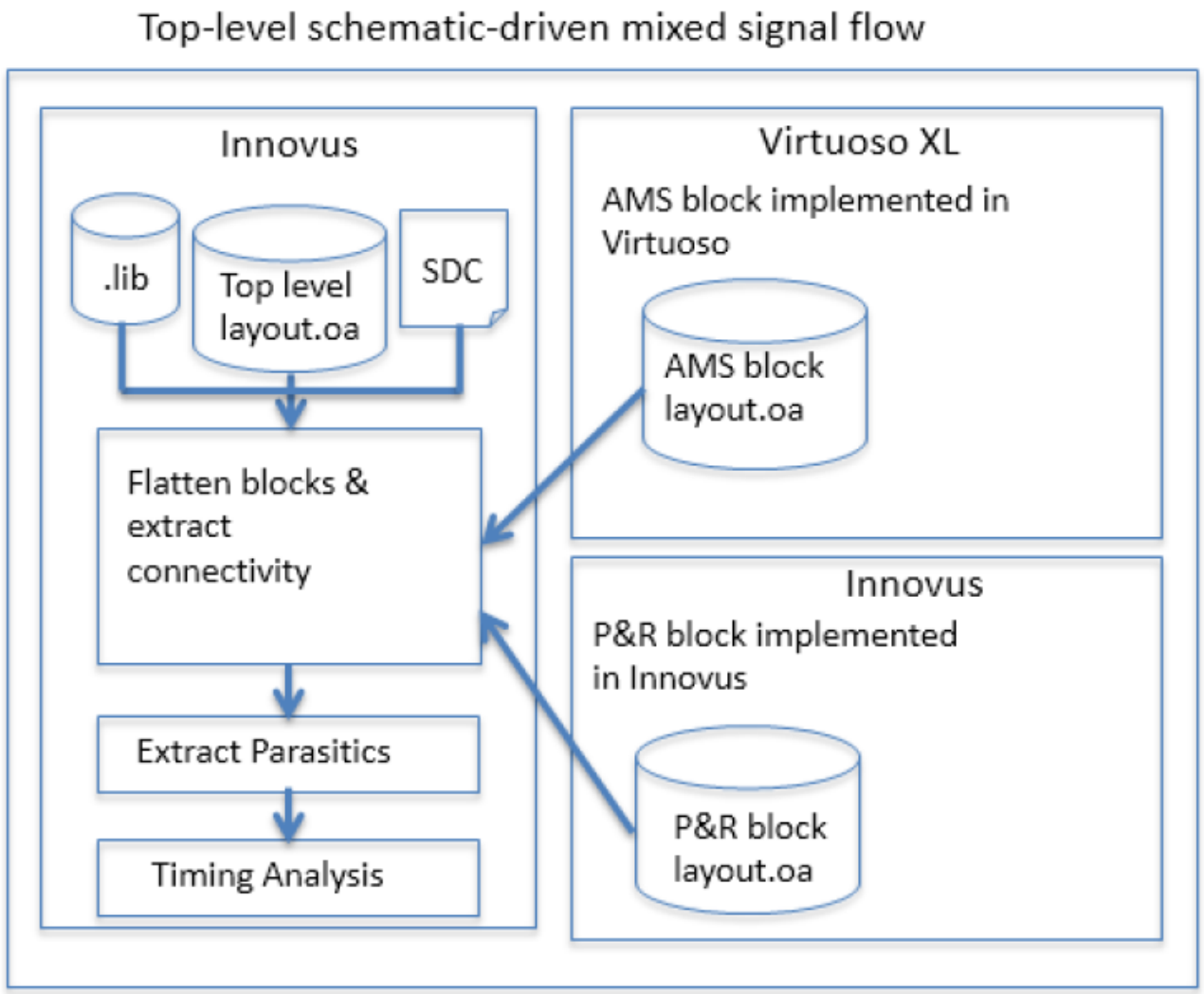
Static Timing Analysis

Running STA by Flattening the Design, schematic driven

- In this method, the Innovus command assemble_design is used to flatten the physical hierarchy to bring the instances and wires at the lower physical level to the top for parasitic extraction and timing analysis.
- Note that the physical hierarchy of the design is flattened only to enable static timing analysis. It does not alter the physical structure of the design in any way.



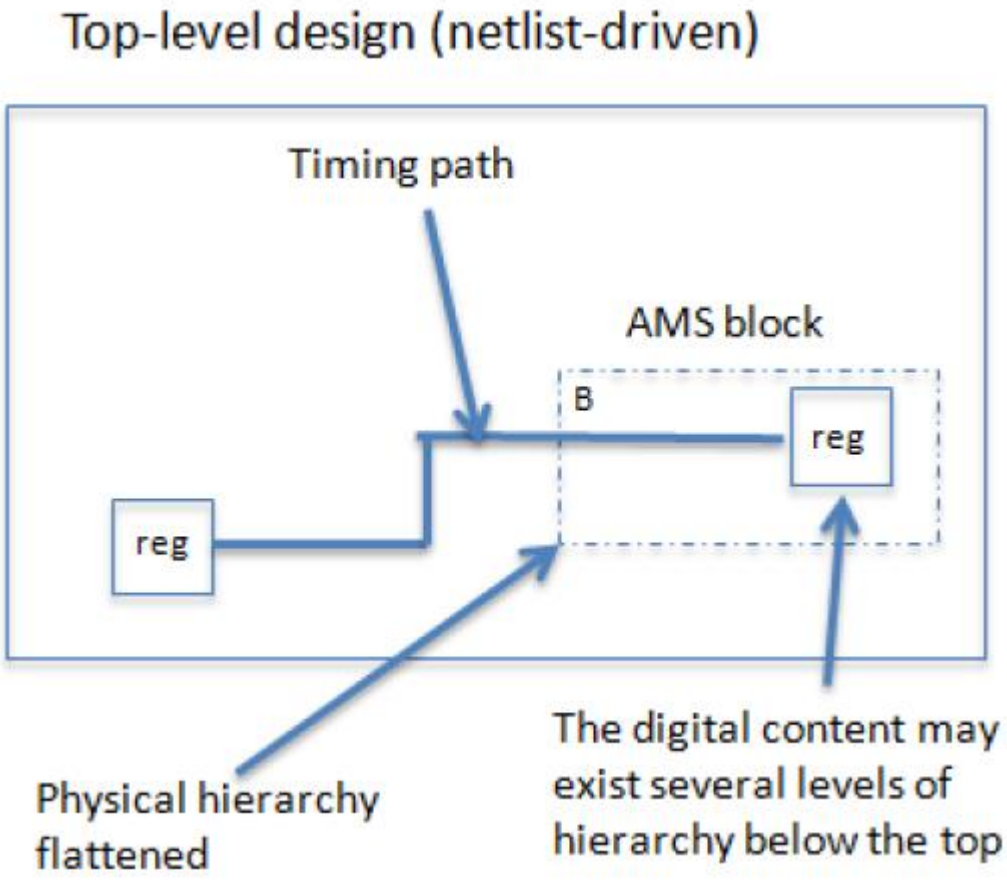
Running STA by Flattening the Design, schematic driven



Running STA by Flattening the Design, schematic driven

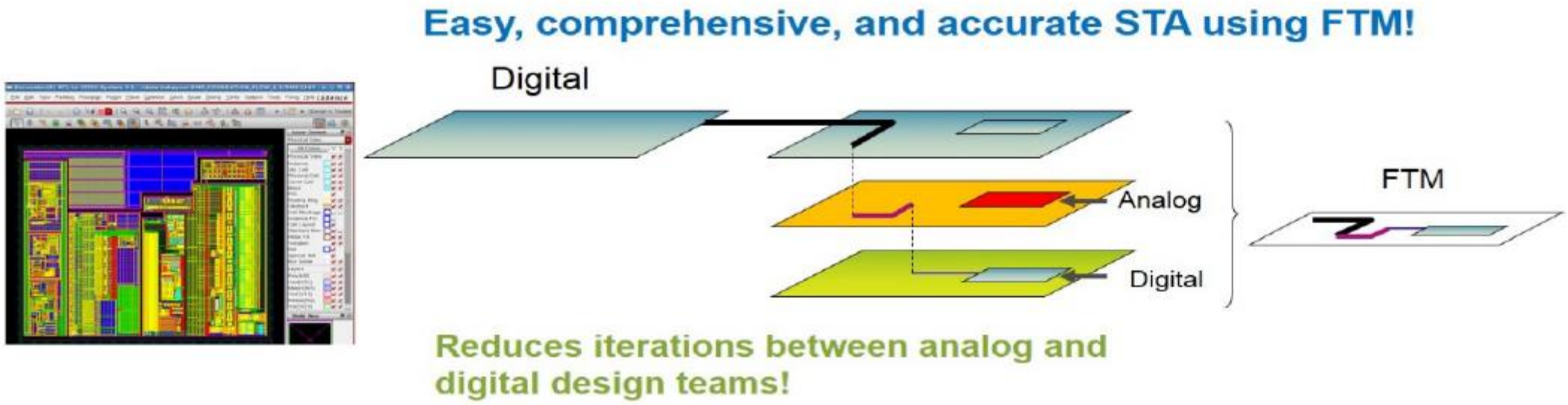
- If the top-level design is originally a schematic-driven design implemented by Virtuoso, the init_design command should be used to load the OpenAccess design.
- Commands to load the design:
  - set_db init_power_nets {VDD AVDD}
  - set_db init_ground_nets {GND AGND}
  - read_mmmc {scripts/viewDefinition.tcl}
  - read_physical -oa_ref_libs {gsclib045}
  - read_netlist -oa_cell_view {mylib top layout}
  - init_design

Running STA by Flattening the Design, netlist driven



Full Timing Model

- The Innovus Digital implementation system enables digital paths and logic within mixed sinal hierarchy to be extracted and included for top-level STA.



Basic FTM flow steps

- Load the AMS block as top in Innovus
  - read_db command can be used to retrieve the OpenAccess database
- Use assemble_design to physically flatten the digital logic to top AMS block
- Save the full AMS block Verilog netlist
- Run physical parasitic extraction and generate spef for each RC corner
- Create FTM using the createILMDataDir command for the AMS block
- Load the top level design and specify the FTM path to run the top-level STA

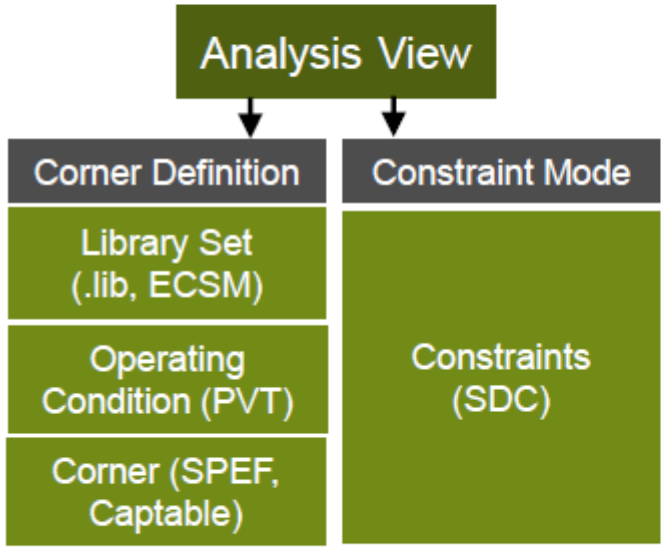
Setup for timing and power analysis

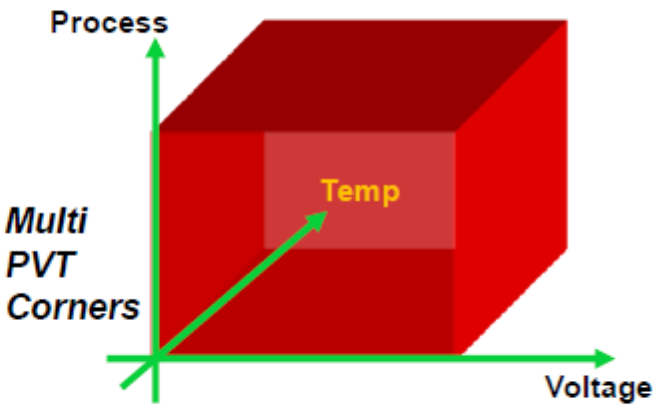
- When running such analysis the job is divided in several steps:
  - Look at the corner to be analysed (user specified) and do pruning to optimize runtime
  - Routing to get wire topology estimations
  - Extraction to get RC network
  - Call timing or power engine to compute results
  - Reporting

- Timing analysis is called regularly during the flow to get a quality of results status.

Multi Mode Multi Corner (MMMC) analysis

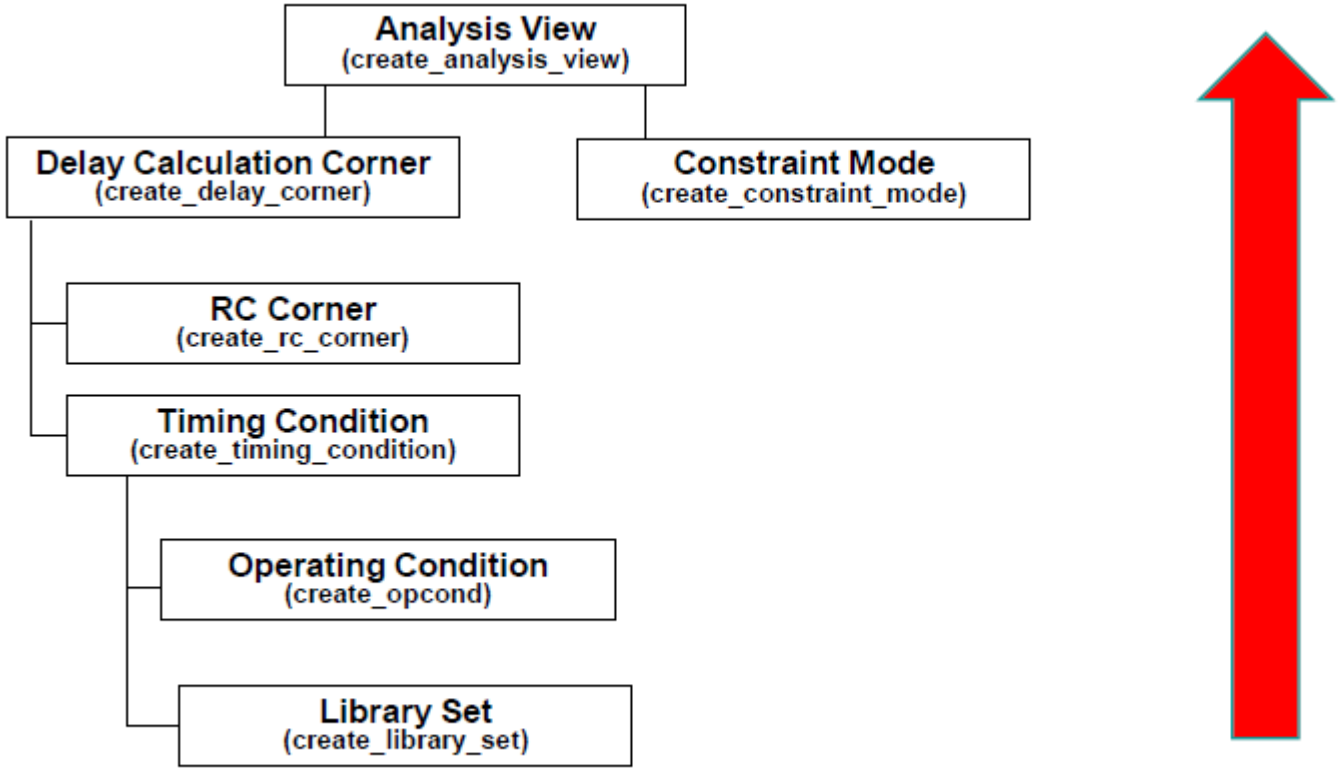
View 1	View 2	View 3	View 4
Functional Mode	Functional Mode	Functional Mode	Test Mode
FF Corner	SS Corner	TT Corner	SS Corner



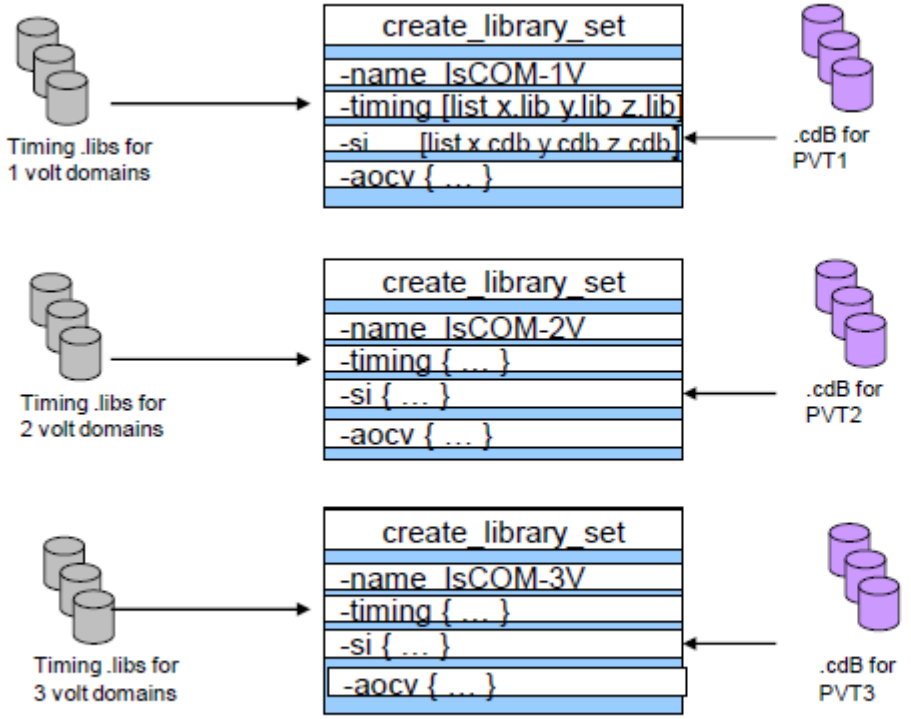


Concurrent analysis/optimization after views are set up

Setting up one analysis view



Creating a library set



- Library sets are created so that library references can be created once and conveniently referred to through the specification process.
- The same library set can be referenced multiple times by different delay calculation corners.
- Creates linkage between timing libraries and signal-integrity libraries (.cdb).

Creating a RC corner

- An RC corner object provides the software with all of the information necessary to extract and use the RCs for delay calculation.
- RC corner objects also control the attributes for running gate-level extraction sequentially on each RC corner.
- For each active RC corner in the design, the software extracts and stores a unique set of parasitics. You must use the RC corner scaling attributes when running the software in Multi-Mode Multi-Corner (MMMC) analysis mode.

To create an RC corner run the following:

```
create_rc_corner
```

For example, the following command creates an RC corner called rc-typ that uses the Quantus™ QRC tech file myTech_nc.qrctech, and derates the resistance values based on the temperature of 50° Celsius. In this example, the –temperature option is used to override the temperature specified in the QRC techfile:

```
create_rc_corner -name rc-typ -qrc_tech myTech_nc.qrctech -temperature 50
```

Timing conditions

- A timing condition is a set of libraries at a specific operating condition. Timing conditions are assigned to power domains or supply sets, and effectively describe the delay calculation requirements for the domains.
- A timing condition is associated with a library_set.
- A timing condition is optionally associated with an operating condition (if not specified uses the individual PVT in each library for that given library).
- To create a timing condition run the following:

```
create_timing_condition
```

Example

```
create_timing_condition -name tcWCCOM
-library_set lsCOM-1.0 -opcond_library slow_1.5.lib
-opcond WCCOM_1.5
```

Creating a delay calculation corner

- A delay calculation corner provides all of the information necessary to control delay calculation for a specific analysis view.
- Each corner contains information on the libraries to use, the operating conditions with which the libraries should be accessed, and the RC extraction parameters to use for calculating parasitic data.
- Delay corner objects can be shared by multiple top-level analysis views.
- Binds power domains to timing conditions with @ syntax.

To create a delay calculation corner, use the following command:

```
create_delay_corner
```

Example

```
create_delay_corner -name DC1 -rc_corner rcMax
-timing_condition {TC1 pd1@TC2 {pd3 pd4}@TC3}
```

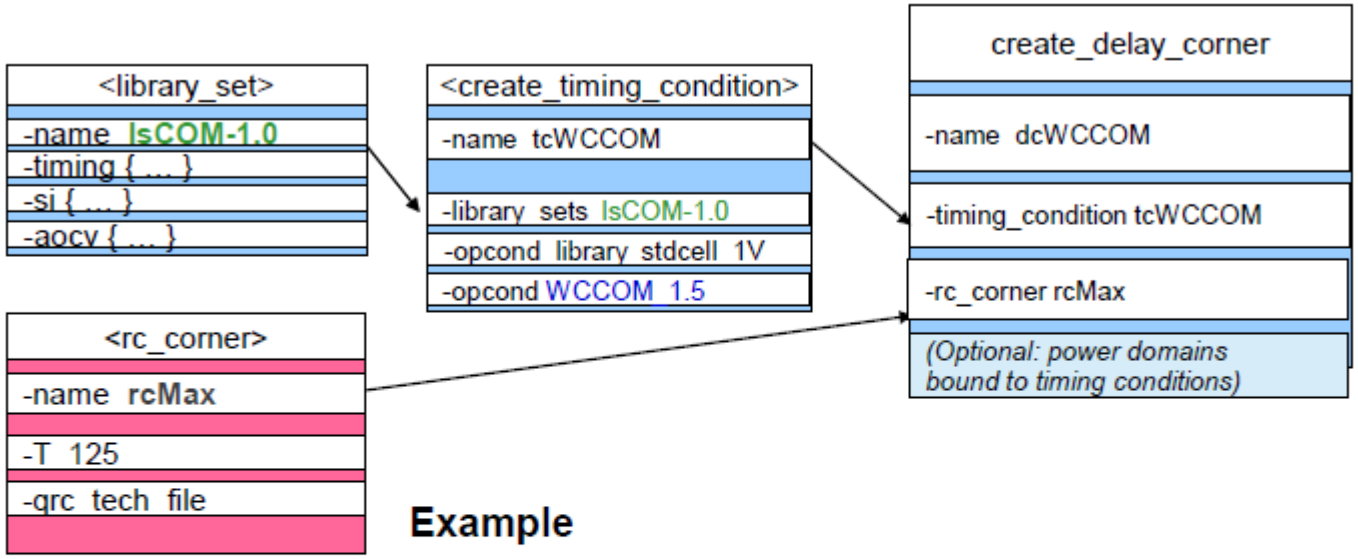
In the example above, TC1, the first element, is the default timing condition that applies to all power domains that are not covered by the *pd@TC* syntax. The default timing condition is mandatory.

You can specify –early_* and –late_* within a single delay calculation corner to control on-chip variation.

Example

```
create_delay_corner -name DC1 -timing_condition {TC1 PD2@TC2} -rc_corner QX
```

Delay calculation corners



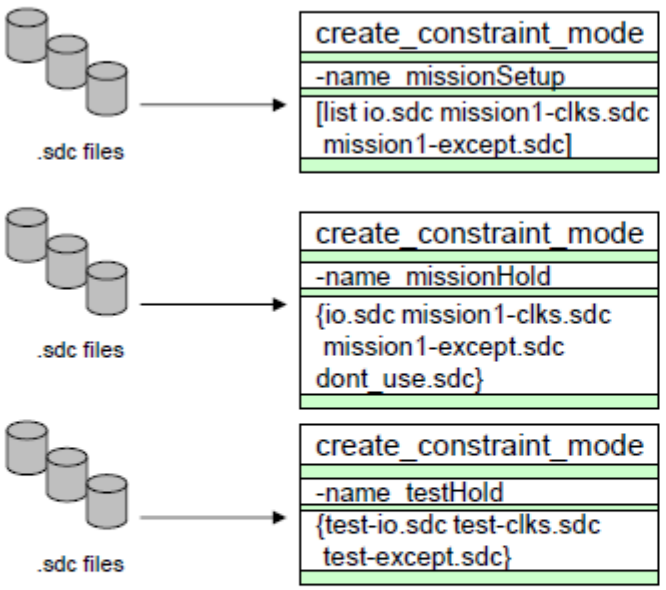
Example

To find out what the delays corners have been set to, enter:

```
get_delay_corner dcWCCOM -timing_condition tcWCCOM
get_delay_corner dcWCCOM -rc_corner rcMax
```

Specify the timing condition using the *–timing_condition* and extraction corner with the *–rc_corner* option.

Constraint Modes

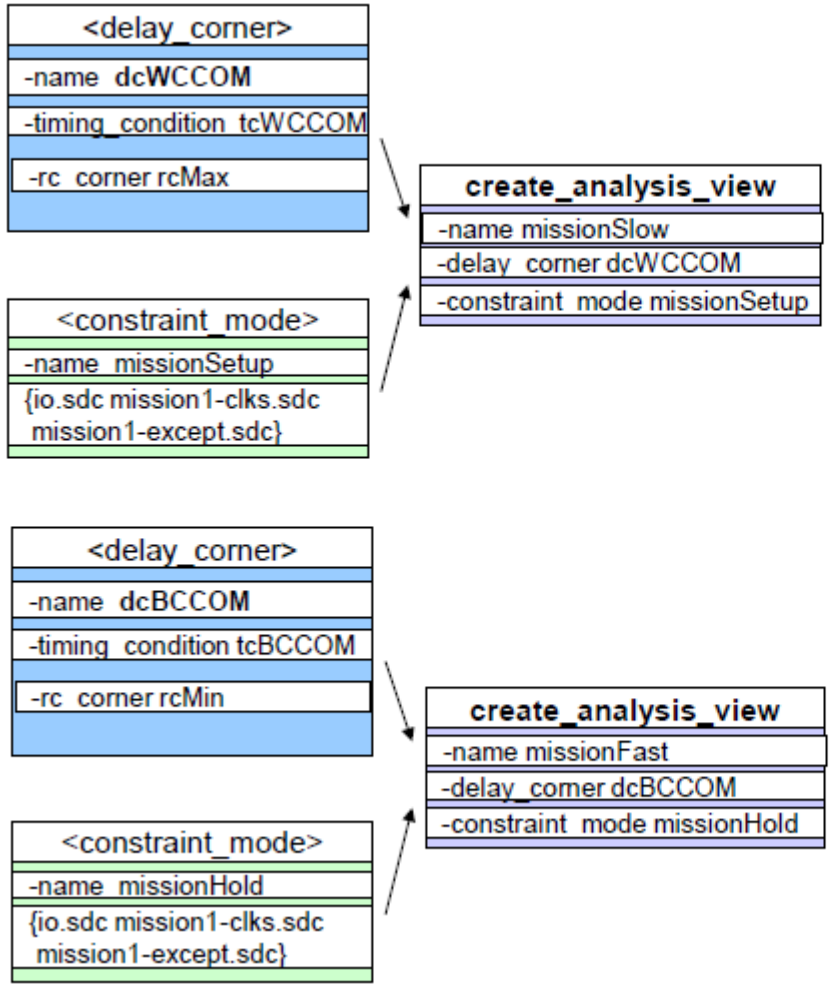


- The *create_constraint_mode* command is used to associate a Tcl list of *.sdc* files with a named mode.
- SDC files can be shared by many different modes.
- A mode defines one of possibly many different functional, test behaviors, or DVFS modes of a design.
- SDC files contain the clock specifications, conditionalizing constants, I/O timings, and path exceptions that make each mode unique.

Example

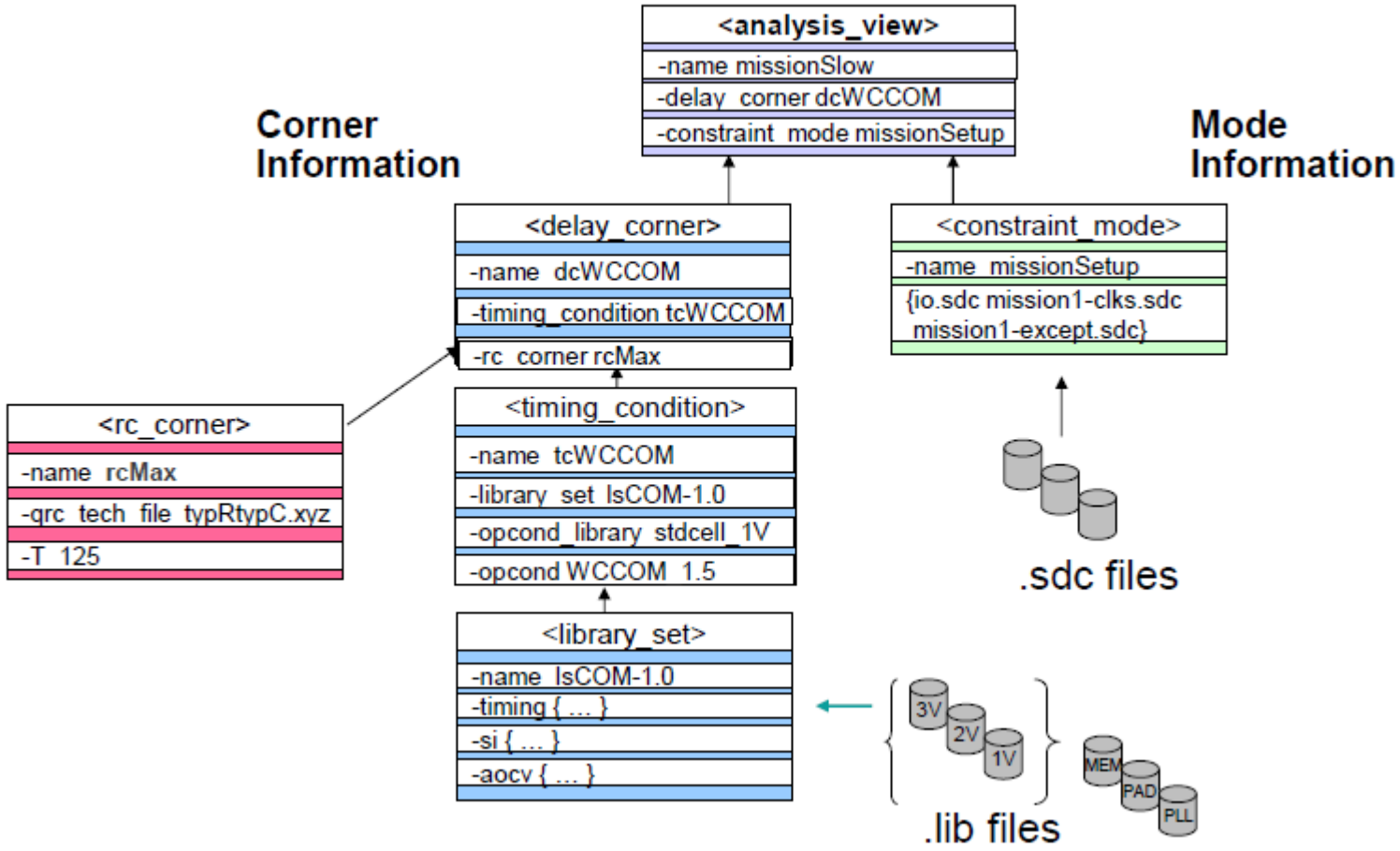
```
get_constraint_mode missionSetup -sdc_files
io.sdc mission1-clks.sdc mission1-except.sdc
update_constraint_mode -name missionSetup -sdc_files  "io.sdc
mission1-clks.sdc mission1-except.sdc fp.sdc"
get_constraint_mode missionSetup -sdc_files
io.sdc mission1-clks.sdc mission1-except.sdc fp.sdc
```

Analysis views



- The *create_analysis_view* command builds a top-level association of a delay calculation corner with a constraint mode.
- The set of active views defined to the system represent the different design variations to be timed and optimized.

MMMC definition summary



Setting active analysis views

After creating analysis views, you must set which views the software should use for setup and hold analysis and optimization including for leakage and dynamic power optimization.

- These "active" views represent the different design variations that are to be analyzed. Active views can be changed throughout the flow to use different subsets of views.
- Libraries and data are loaded into the system as specified by the active views.

To set active analysis views, use the following command:

```
set_analysis_view
```

The following command sets *missionSlow* and *mission2Slow* as the active views for setup analysis, and *missionFast* and *testFast* as the active views for hold analysis:

```
set_analysis_view -setup {missionSlow mission2Slow} \
-hold {missionFast testFast} -leakage {max_leakage_view} -dynamic
{max_dynamic_view}
```

Checking MMMC configuration

To report your current multi-mode multi-corner configuration, run the following commands:

```
report_analysis_views

all_analysis_views

# reports a list of all views

get_db analysis_views -if
  {.is_setup == "true"}

get_db analysis_views -if
  {.is_hold == "true"}

get_db analysis_views -if
  {.delay_corner.name ==
  "dcWCCOM"}

...
```

```
+ ALL Views
  |
  + Analysis View:  default_analysis_view_setup
    |
    + Delay Calc Corner: default_delay_corner_max
    |   |
    |   + timing_condition: default_mapping_tc_1
    |   |   |
    |   |   + library_sets: default_libset_max
    |   |       |
    |   |       + timing: /home/vinita/InnovusBlk_CUI_17_1/FPR/work/place.inn/libs/mmmc/rom_512x16A_slow_syn.lib
                          /home/vinita/InnovusBlk_CUI_17_1/FPR/work/place.inn/libs/mmmc/tpz973gwc-lite_slow.lib
                          /home/vinita/InnovusBlk_CUI_17_1/FPR/work/place.inn/libs/mmmc/ram_256x16A_slow_syn.lib
                          /home/vinita/InnovusBlk_CUI_17_1/FPR/work/place.inn/libs/mmmc/ram_128x16A_slow_syn.lib
                          /home/vinita/InnovusBlk_CUI_17_1/FPR/work/place.inn/libs/mmmc/slow.lib
                          /home/vinita/InnovusBlk_CUI_17_1/FPR/work/place.inn/libs/mmmc/pllclk_slow.lib
    |   |       |
    |   |       + si: /home/vinita/InnovusBlk_CUI_17_1/FPR/work/place.inn/libs/mmmc/slow.cdb
    |   |
    |   |
```

Customize the report to show *only* one of the following:

- The active setup or hold analysis views.
- All of the active views.
- All of the defined views in the design, including those that are currently inactive.

Specifying extraction mode

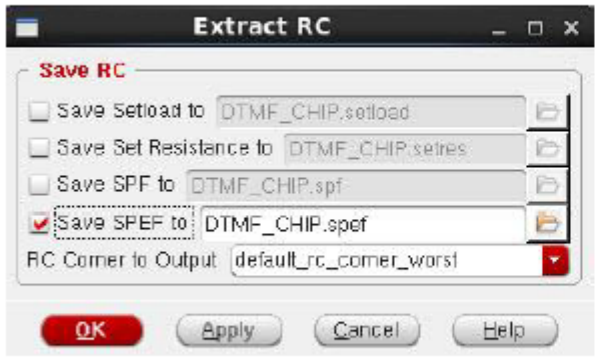
```
set_db extract_rc_effort_level {low | medium | high | signoff}
```

Where

- Low enables the detailed extraction engine.
- Medium enables the pre-route extraction engine for preRoute and tQuantus extraction engine for post_route extraction.
- High enables the integrated Quantus extraction engine.
- Signoff enables standalone Quantus extraction.

Running extraction

Timing – Extract RC



Command
`extract_rc`

To delete the extracted parasitics but to maintain the RC extraction modes that were set, run the command
`reset_parasitics`

- *Setload* format (lumped capacitance on nets)
- *Setres* format (resistance on nets)
- SPF saves the DSPF (detailed standard parasitic format) file
- SPEF (standard parasitic exchange format)

Performing timing analysis

The *time_design* command reports the results from each active analysis view, as well as an aggregated summary.

Syntax

```
time_design
```

By default, *time_design* creates only an aggregated summary.

Use the *–expanded_views* option to report timing of all views.

For multi-mode multi-corner analysis, the software creates a separate directory for each view.

For example, if your design has two analysis views, view1 and view2, the output reports are generated in the *./timingReports/view1* and *./timingReports/view2* directories.

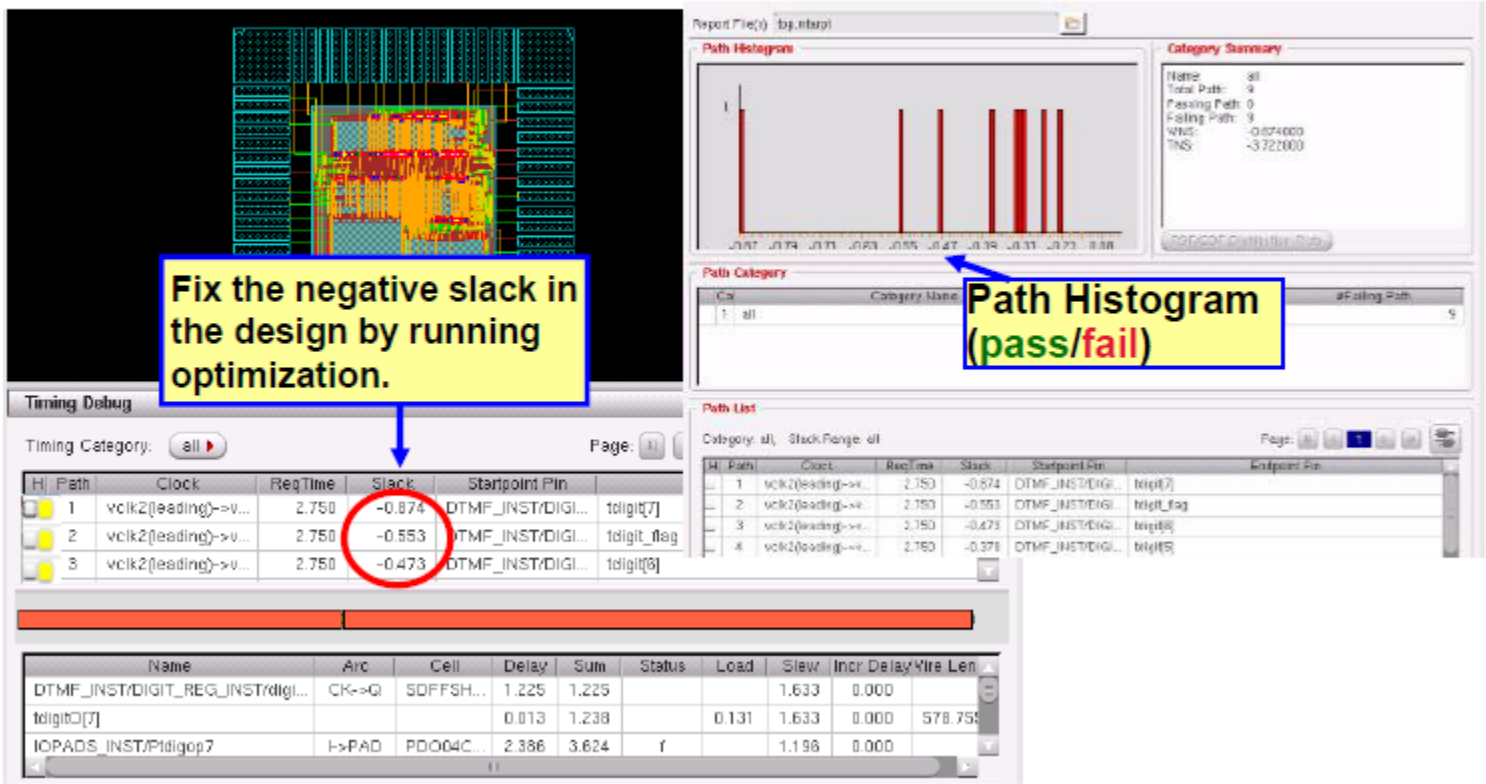
Command

```
time_design … -expanded_views
```

```
+--------------------+---------+---------+---------+
|     Setup mode     |   all   | reg2reg | default |
+--------------------+---------+---------+---------+
|           WNS (ns):| -0.545  | -0.545  |  2.895  |
|           TNS (ns):| -4.041  | -4.041  |  0.000  |
|    Violating Paths:|   22    |   22    |    0    |
|          All Paths:|   622   |   580   |   266   |
+--------------------+---------+---------+---------+
|AV_HL_FUNC_MAX_RC1  | -0.545  | -0.545  |  2.895  |
|                    | -4.041  | -4.041  |  0.000  |
|                    |   22    |   22    |    0    |
|                    |   622   |   580   |   266   |
+--------------------+---------+---------+---------+
|AV_HL_FUNC_MAX_RC2  |  0.019  |  0.019  |  3.815  |
|                    |  0.000  |  0.000  |  0.000  |
|                    |    0    |    0    |    0    |
|                    |   622   |   580   |   266   |
+--------------------+---------+---------+---------+
|AV_LO_FUNC_MAX_RC1  |  7.455  |  7.455  | 10.895  |
|                    |  0.000  |  0.000  |  0.000  |
|                    |    0    |    0    |    0    |
|                    |   622   |   580   |   266   |
+--------------------+---------+---------+---------+
```

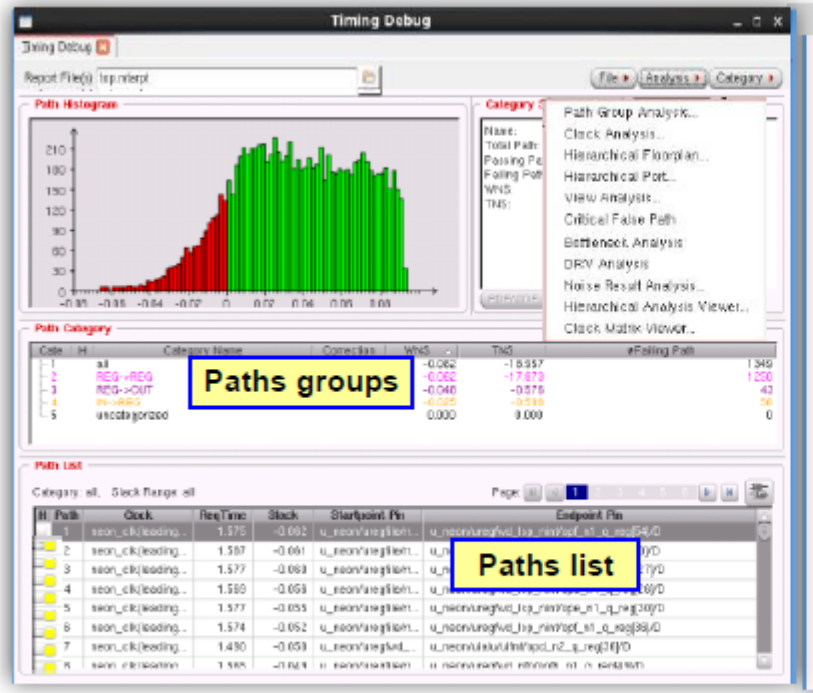
Timing debug utility

After running timing analysis, choose **Timing – Debug Timing** to help debug the causes of timing violations in your design.

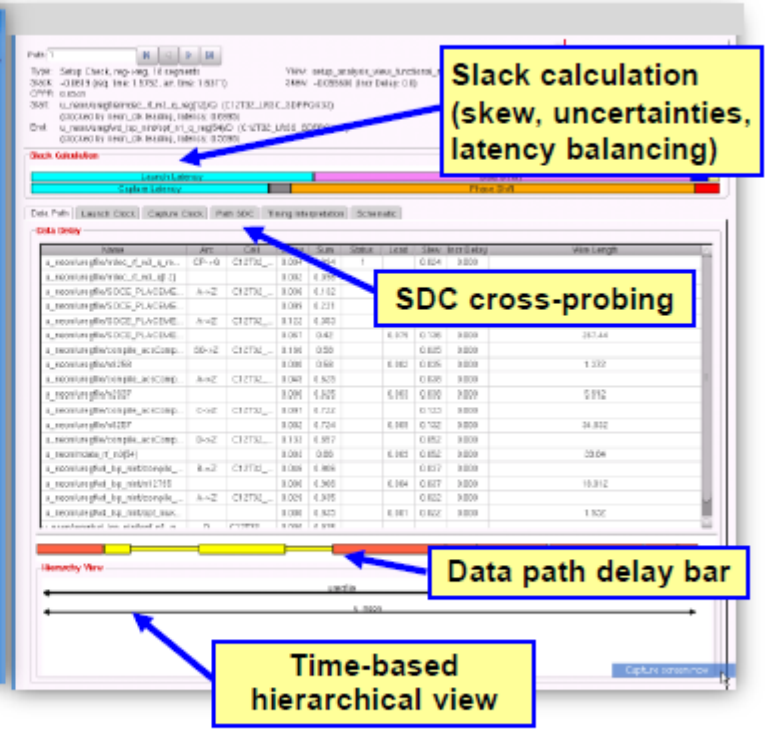


Timing debug utility

Global Path Histogram



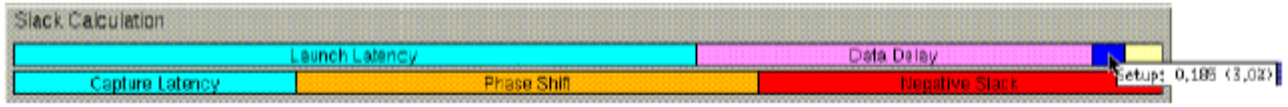
Detailed Path Viewer



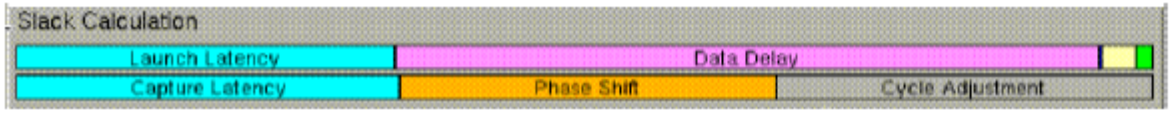
Reduces thousands of failing paths to a small number of unique problems to solve.

Examples of debugging with Timing Path Analyzer

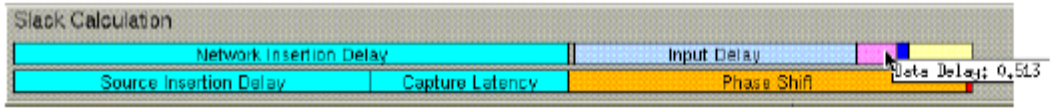
Launch and capture latency components are not aligned. Therefore, there can be large clock latency mismatch in this path.



The cycle adjustment bar in the required time indicates the presence of multicycle path.

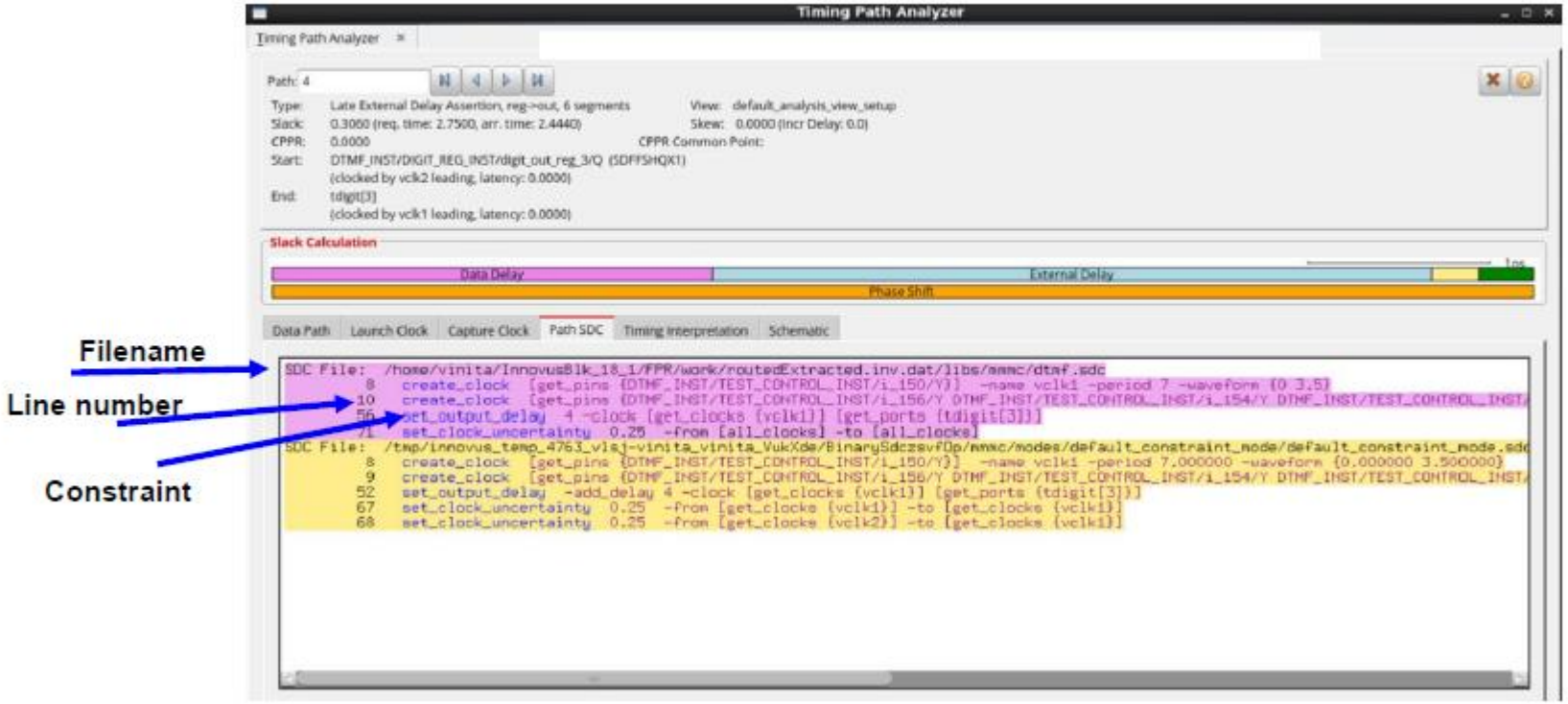


Large input delay in an I/O path is represented by the light-blue bar in the arrival time.



Timing Path Analyzer: Path sdc

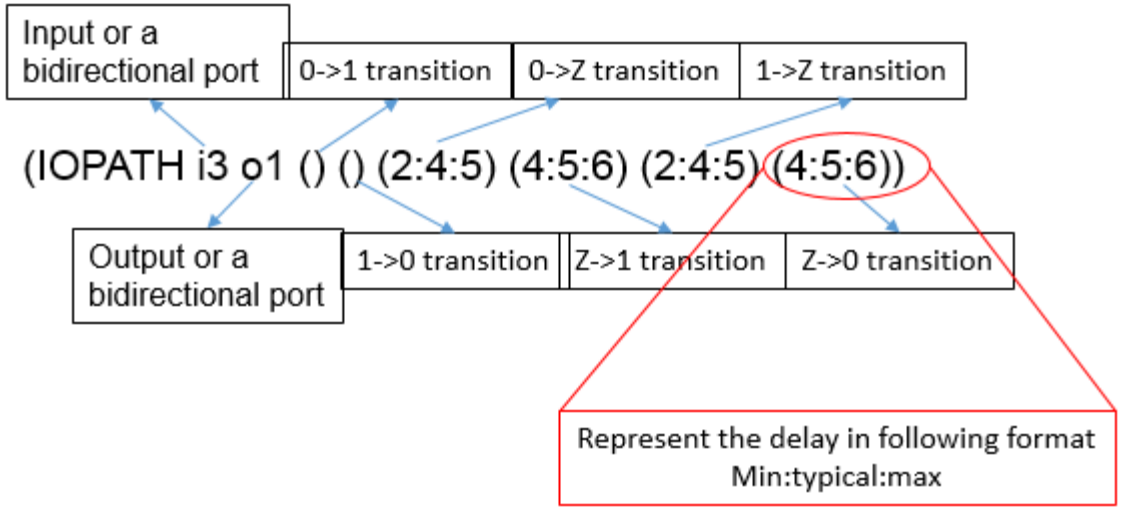
The Path SDC tab shows all the SDC constraints that match the topology of the current path. Use this software to debug typical constraint issues.



Export sdf file

- Sdf stands for Standard Delay Format

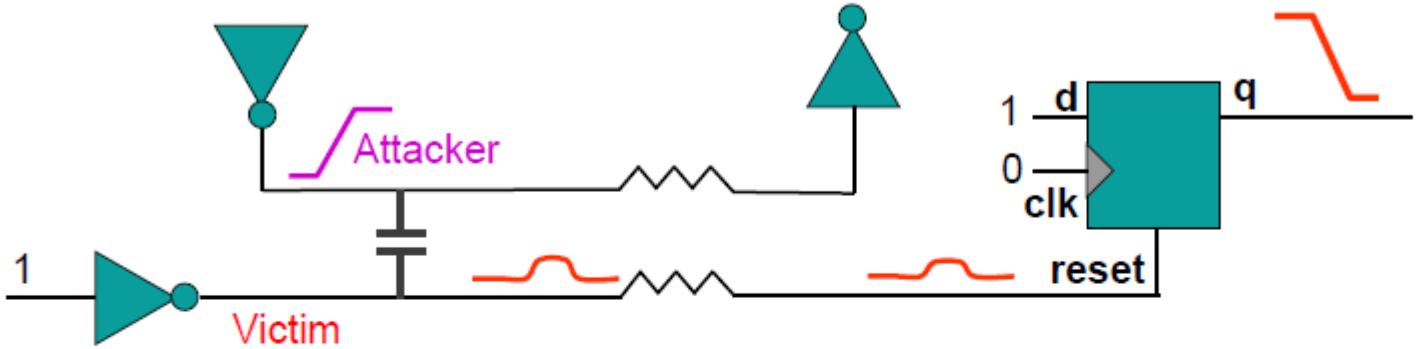
- You can export it from Innovus with the command:
  - write_sdf -edges check_edge -version 3.0 -precision 3 -target_application verilog



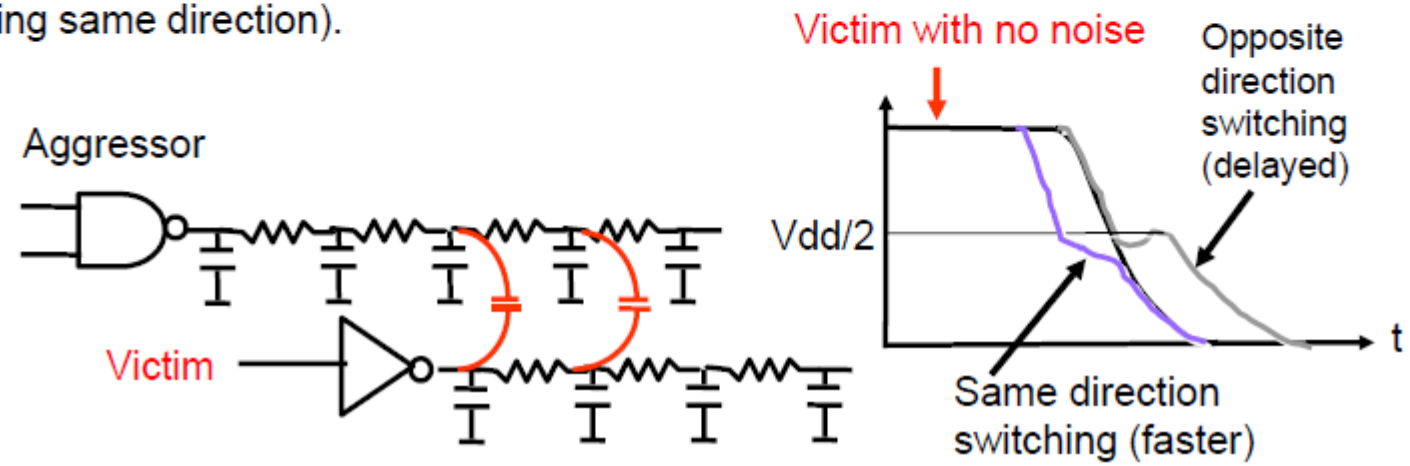
Crosstalk effect analysis

Impact of noise

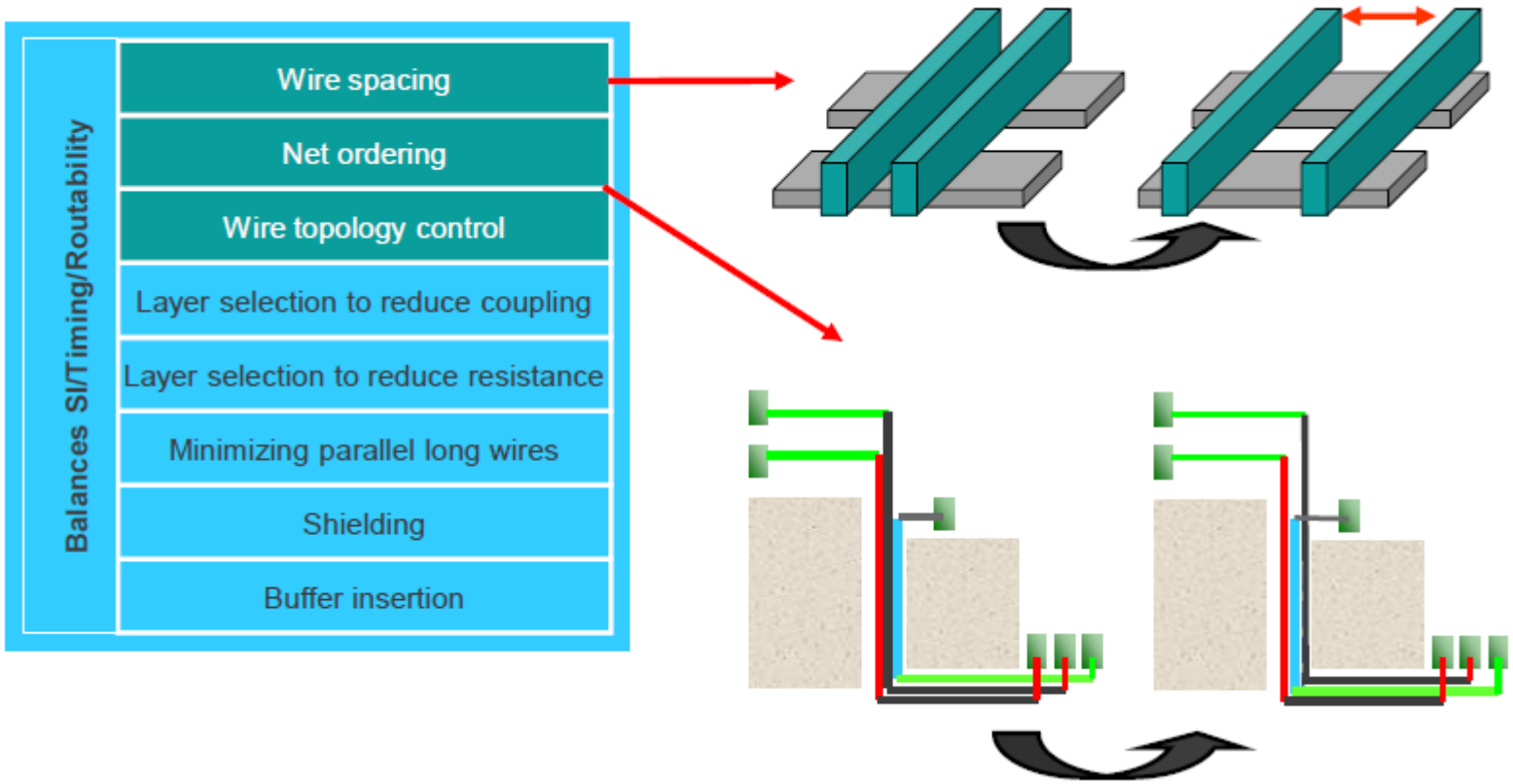
A glitch noise that exceeds a threshold can cause functional failures.



Crosstalk noise can create timing problems involving setup (switching opposite direction) and hold (switching same direction).

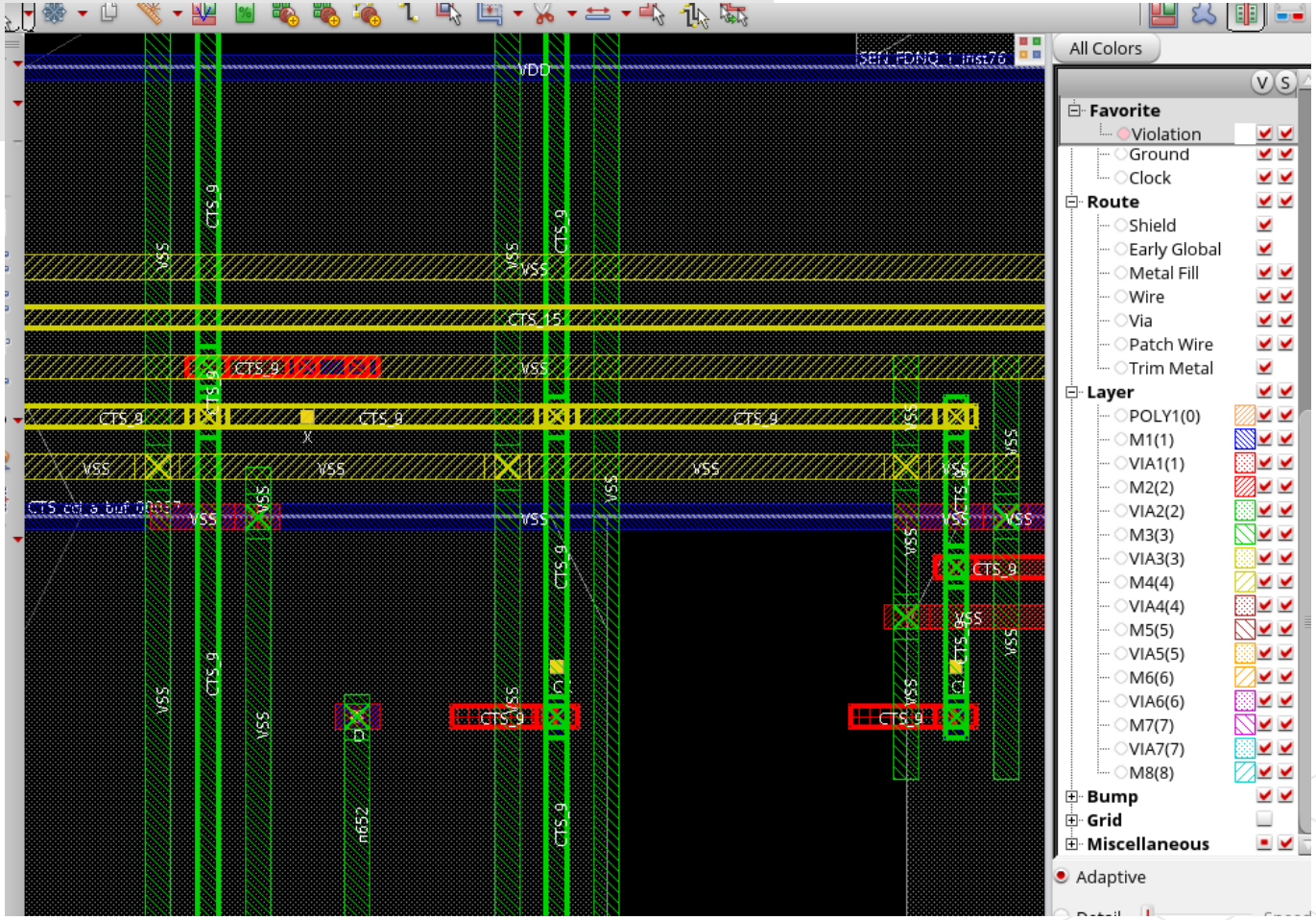


Crosstalk avoidance



Shielding of clock nets

```
create_route_type -name clock_route -top_preferred_layer M5 -bottom_preferred_layer M3 -shield_net VSS
set_db cts_route_type_top clock_route
set_db cts_route_type_trunk clock_route
set_db cts_route_type_leaf clock_route

ccopt_design
```



Fixing crosstalk Post-route

Choose **ECO – Optimize Design.** Select **Post-Route, Include SI**, and click **SI Options**.

Command

```
opt_design -post_route
```

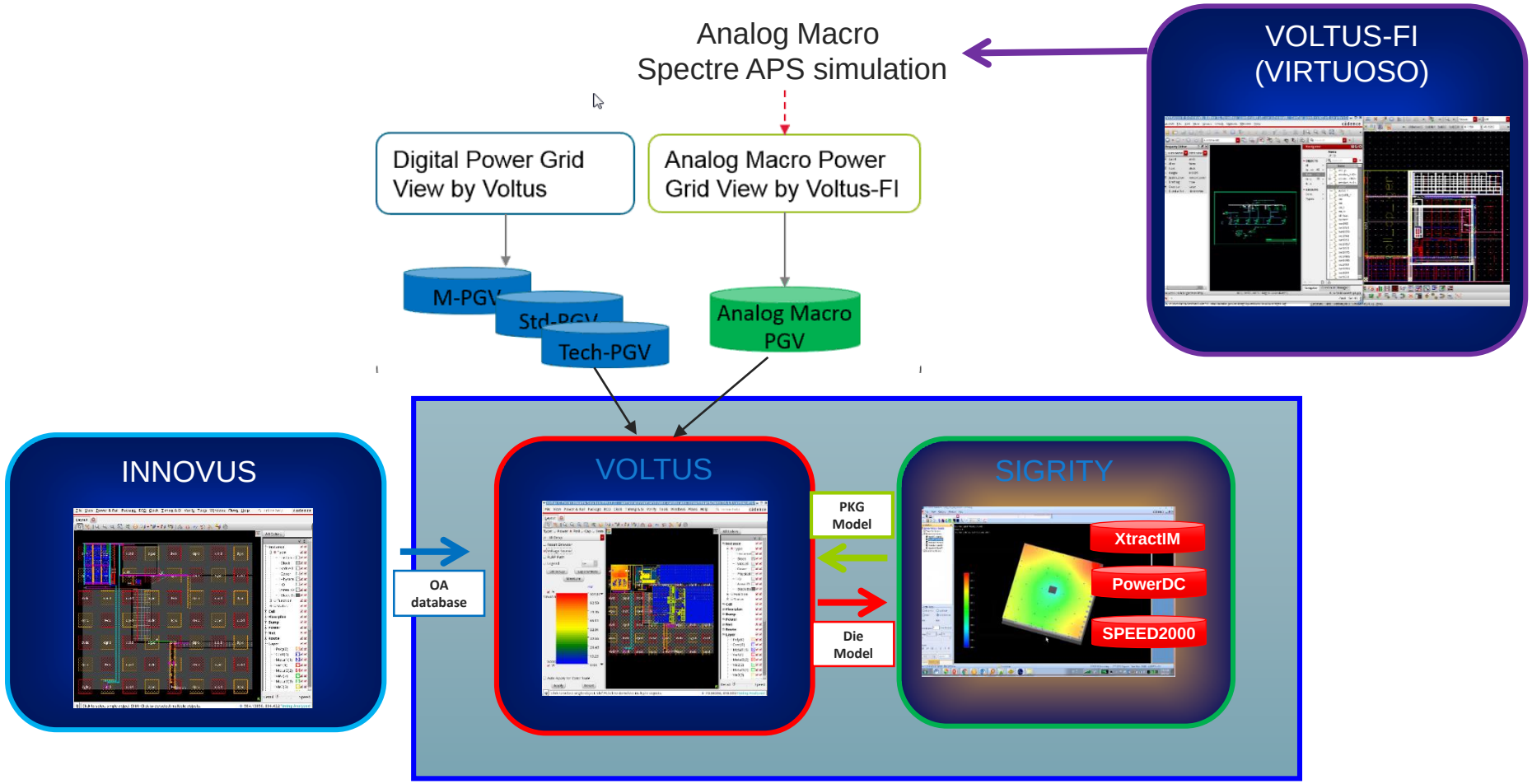
By default, delay due to Signal Integrity are computed, and glitch fixing is activated if using –postroute.

To compute only base delay you can setup:
set_db delaycal_enable_si false

To turn on Signal Integrity delay but turn off glitch optimization you can use:
set_db opt_post_route_fix_glitch false

Power analysis

Power analysis of Mixed Signal design (Digital on Top)



Verifying power ground shorts

It checks for power and ground shorts between two geometries belonging to different nets.

The command performs power and ground short check between the following:

- PG and PG nets
- PG and signal nets
- PG and other special nets

Tcl Usage

```
check_pg_shorts -out_file <>
```

```
@[DEV]voltus 24> check_pg_shorts -out_file pg
 *** Starting VERIFY PG SHORT(MEM: 1079.7) ***

  VERIFY PG SHORT ...... Initializing
  VERIFY PG SHORT ...... Deleting Existing Violations
  VERIFY PG SHORT ...... Creating Sub-Areas
                  ...... bin size: 3840
  VERIFY PG SHORT ...... SubArea : 1 of 1
  VERIFY PG SHORT ...... Short:  0 Viols.
  Verification Complete : 0 Short Viols.

**********End: VERIFY PG SHORT**********
 *** verify PG short (CPU: 0:00:00.6  MEM: 94.6M)
```

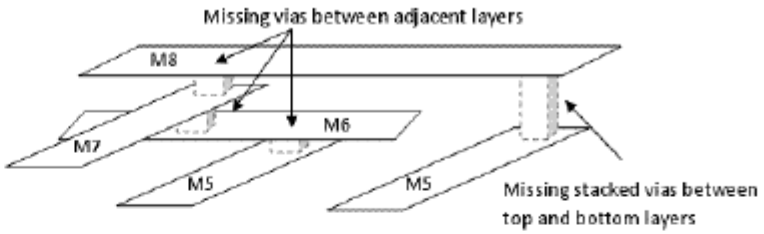
Log file snippet

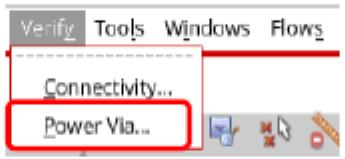
Violation Report

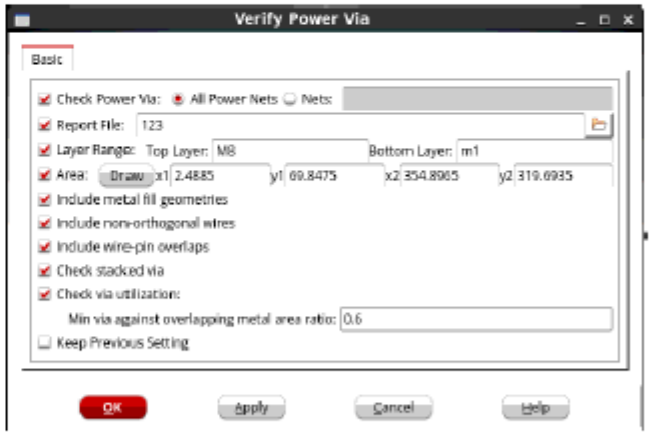
```
#############################################################
#  Generated by:      Cadence Voltus IC Power Integrity Solution
#  OS:                Linux x86_64(Host ID noi-sakshin)
#  Generated on:      Thu Mar 15 16:46:40 2018
#  Design:            super_filter
#  Command:           check_pg_shorts -out_file pg
#############################################################


Begin Summary ...
  Verification Complete : 0 Short Viols.
End Summary
```

Verifying power via

- Checks for metal geometry overlap on PG nets
- Ignores metal fill geometries by default
- Generates text report with location of missing via on PG net
- Highlights violations in the layout canvas
- Supports stacked via check using the *-stacked_via* option which checks for missing vias between all non-adjacent as well as adjacent layers.
- Performs missing power via analysis in the user-defined region
- Checks missing power vias on wire and pin overlaps using the *-check_wire_pin_overlap* option
- Checks via density utilization in metal overlaps







Tcl Command

```
check_power_vias -layer_range {M8 M1} -area {4.805 1.2785} {353.46 256.396} \
-check_fill -non_orthogonal_check -check_wire_pin_overlap -stacked_via
```

Calculating Power consumption Data

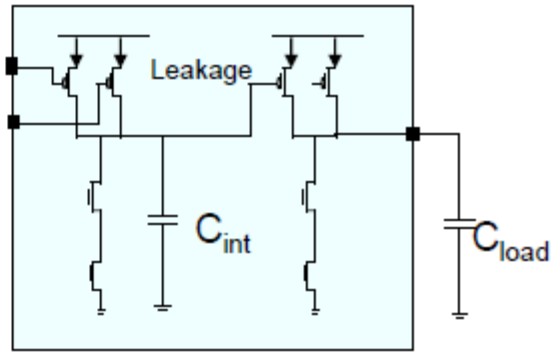
Switching power

- The power consumed by charging and discharging of load capacitance.
- Formula: $P_{avg} = ½ * C_{load} * V^2 * F * A$

  where C_{Load} is the output capacitive loading, V is the voltage, F is frequency, and A is the average switching activity either from VCD or computed.

Internal power

- The power consumed in charging and discharging of interconnect and device capacitances internal to a cell:
  - Input-pin-related internal power
  - Arc-based internal power
  - Crossbar current



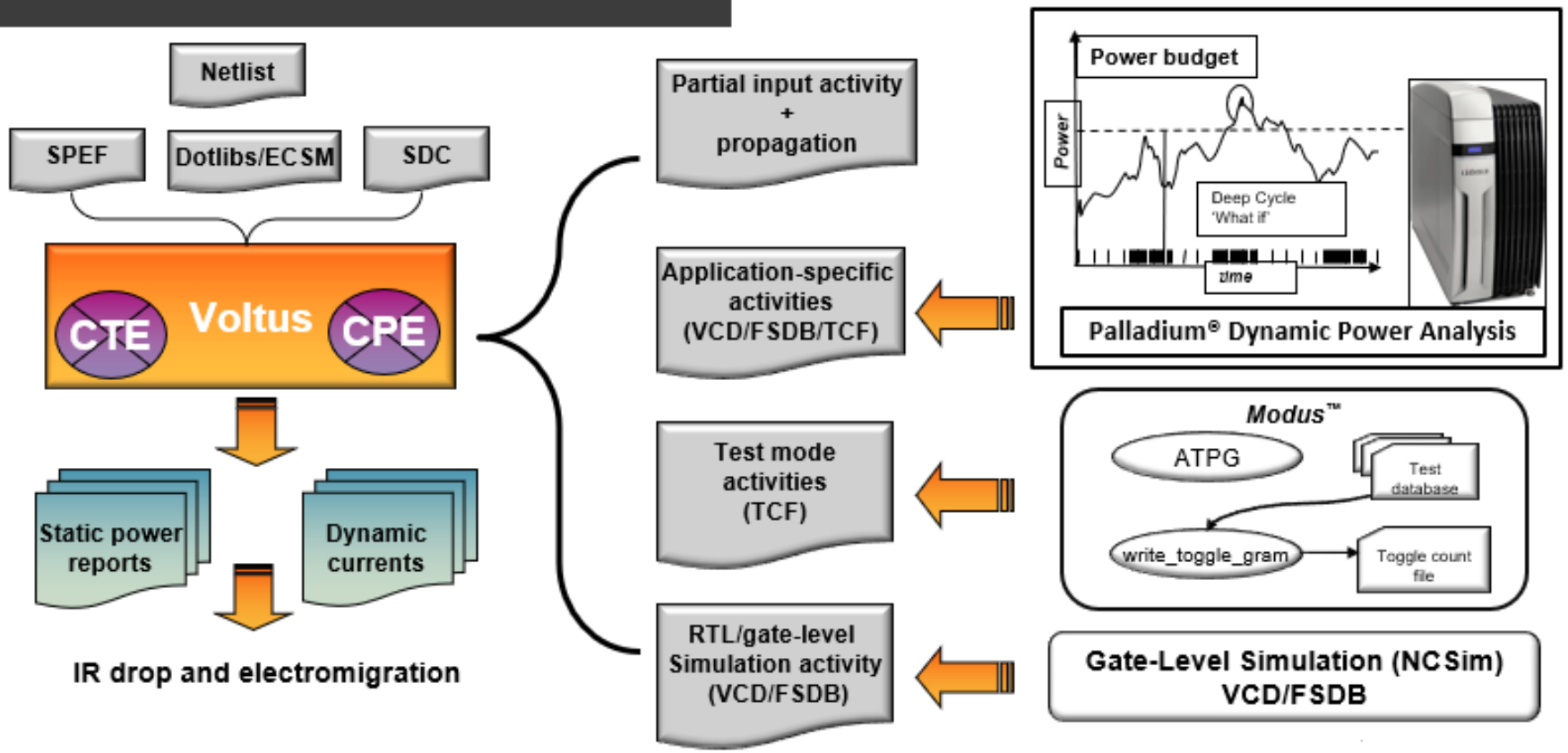
Leakage power

- The power consumed by devices when they are not switching
- Supports state-dependent leakage power calculation based on *.lib* file
- Requires state-dependent leakage library characterization
- Supports K factor (for process, temperature, and voltage) in the *.lib* file
- High-leakage power for 130nm; significant issue for 90nm and below

Activity setup for power calculation

How do you get realistic power estimates for your design?

Solution: Voltus Common Power Engine (CPE) provides accurate activity information which enables accurate static and dynamic power estimation.



Tcl command sample: static (average) power calculation

```
read_db innovus.db

read_spef ../innovus_cui/test_top_cbest.spef -rc_corner cbest
read_spef ../innovus_cui/test_top_cworst.spef -rc_corner cworst

set_analysis_view -setup {func_worstlib_cworst} -hold {func_worstlib_cworst}

set_default_switching_activity -reset
set_default_switching_activity -input_activity 0.2 -sequential_activity 0.2

#read_activity_file -format VCD -start 10300ns -end 10500ns funcmode3.vcd

set_db power_method static
set_db power_report_statistics true
set_db power_report_missing_nets true
set_db power_write_db true
set_db power_write_static_currents true

set_power_output_dir static_power_cworst
report_power -out_file static_power_cworst/power.rpt
```

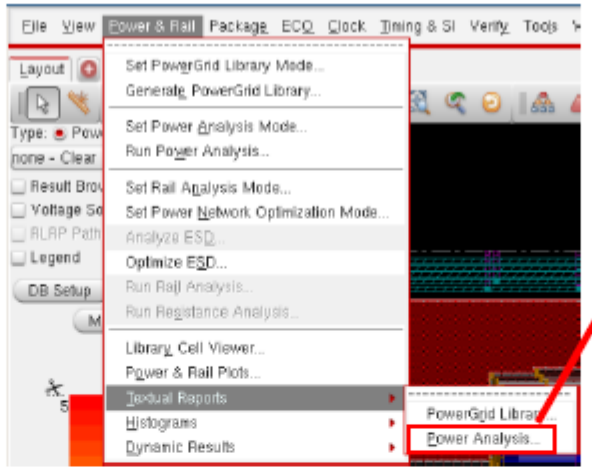
Generate db to be able to display power maps

Generate current files (.ptiavg) to be used later for irdrop

Power reports

After the static-power database is generated, you can generate incremental static power reports based on:

- Clock domain
- Power domain
- Hierarchy level
- Instance and cell
- Cell type
- Net based
- MMMC view based



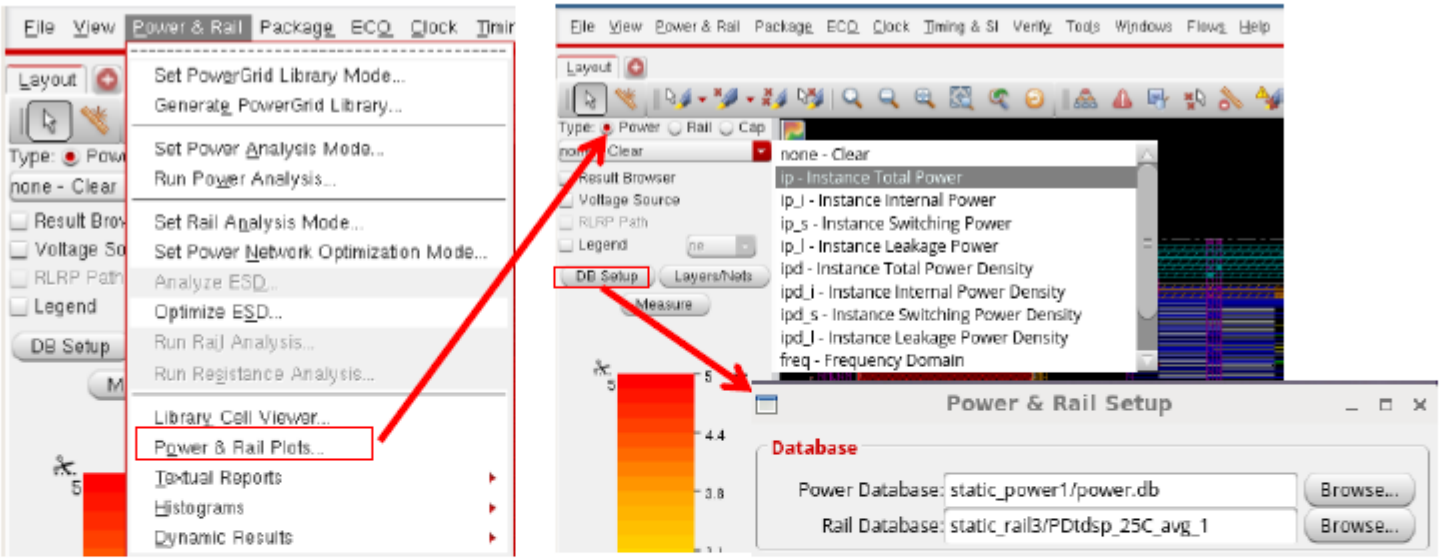

```
-----------------------------------------------------------------------------------
Group                          Internal   Switching    Leakage      Total  Percentage
                               Power      Power        Power        Power  (%)
-----------------------------------------------------------------------------------
Sequential                       2.365      0.2023       0.155      2.722       56.8
Macro                                0        0.5           0        0.5      10.43
IO                                   0          0           0          0          0
Combinational                   0.6282     0.3563     0.06186      1.046      21.83
Clock (Combinational)           0.3023     0.1952     0.01079     0.5083      10.61
Clock (Sequential)             0.01212   0.002453   0.001092    0.01566     0.3268
-----------------------------------------------------------------------------------
Total                            3.307      1.256      0.2288      4.792        100
-----------------------------------------------------------------------------------

-----------------------------------------------------------------------------------
*       Power Distribution Summary:
*               Highest Average Power:                    u0 (pll):          0.5
*               Highest Leakage Power: ring/clk__FENCE_MSV_Fence_I_I0 (CLKBUFX20):  0.000367
*               Total Cap:      3.87233e-11 F
*               Total instances in design:  2674
*               Total instances in design with no power:     0
*           Total instances in design with no activity:     0
*               Total Fillers and Decap:     0
-----------------------------------------------------------------------------------
```

Text Report

Tcl Command

```
report_power -out_file clk.rpt -clock_network pll_clk -cell_type all -sort total
```

Viewing power results maps



If the power database is not specified, the static power stored in memory is used to plot the data.

TCL Command

```
read_power_db -in_file power.db
gui_set_power_rail_display -plot <ip | ip_i | ip_s | ip_l | freq | td |
slack |load> \  -enable_result_browser true
```

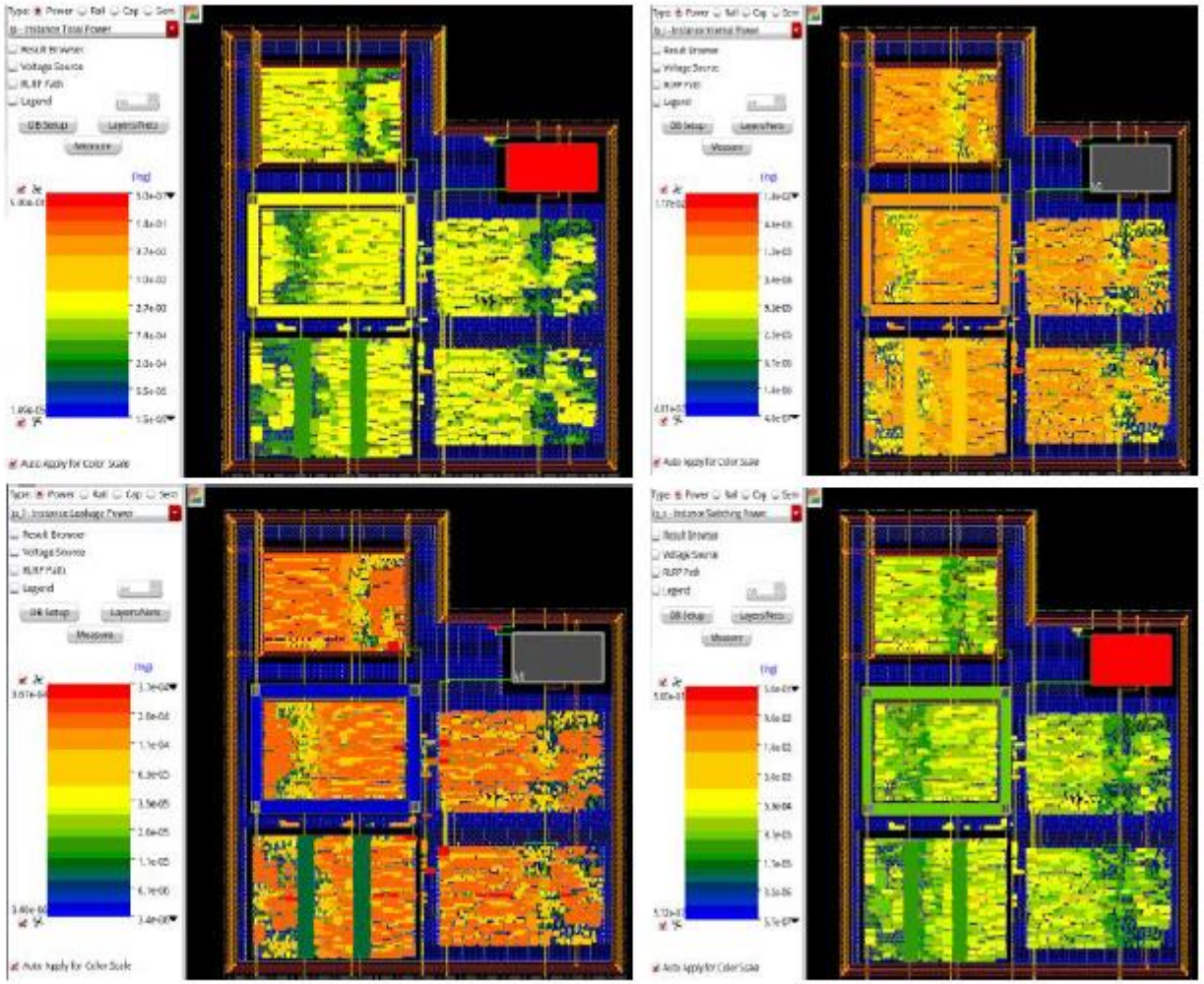
Available power maps

Static power can be read in one of two ways:

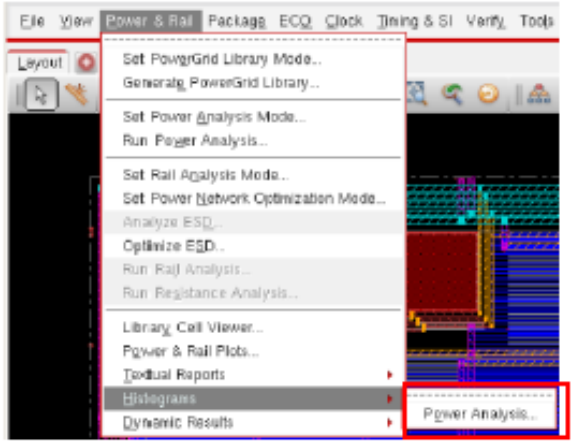
- From calculated power stored in memory during the session
- From *power.db* generated during static power calculation

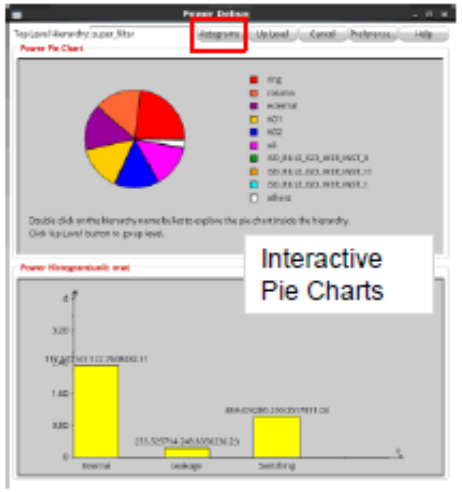
You can display the following plots:

- Instance total power and density
- Instance internal power and density
- Instance switching power and density
- Instance leakage power and density
- Frequency domain
- Slack
- Transition density
- Loading capacitance



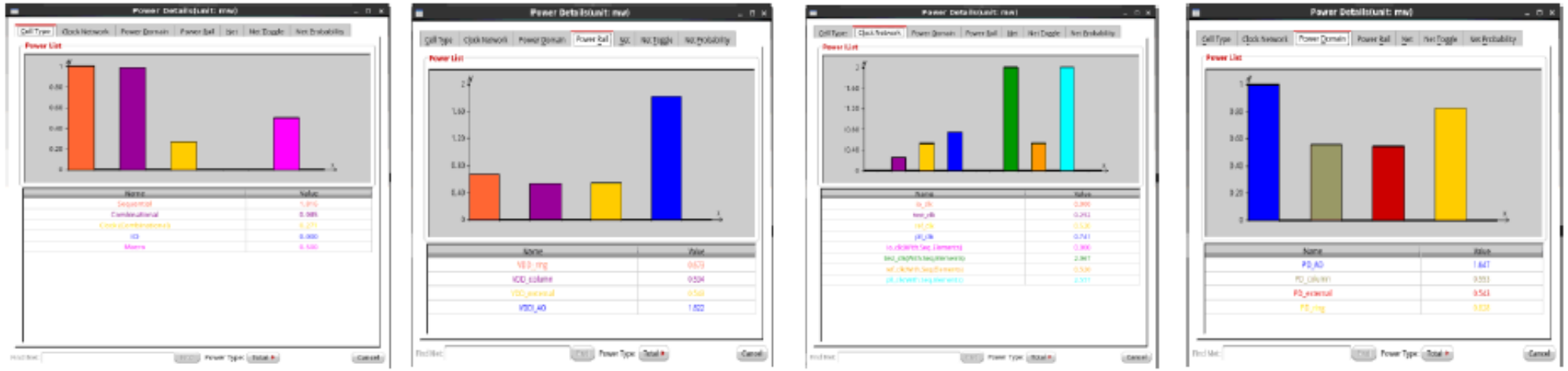
Histograms





Histogram by:

- Cell types
- Power domain
- Power nets
- Clock domain
- Net toggle
- Net probability



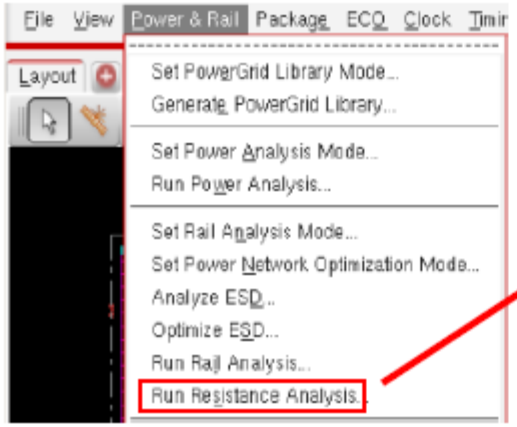
Debugging instance power

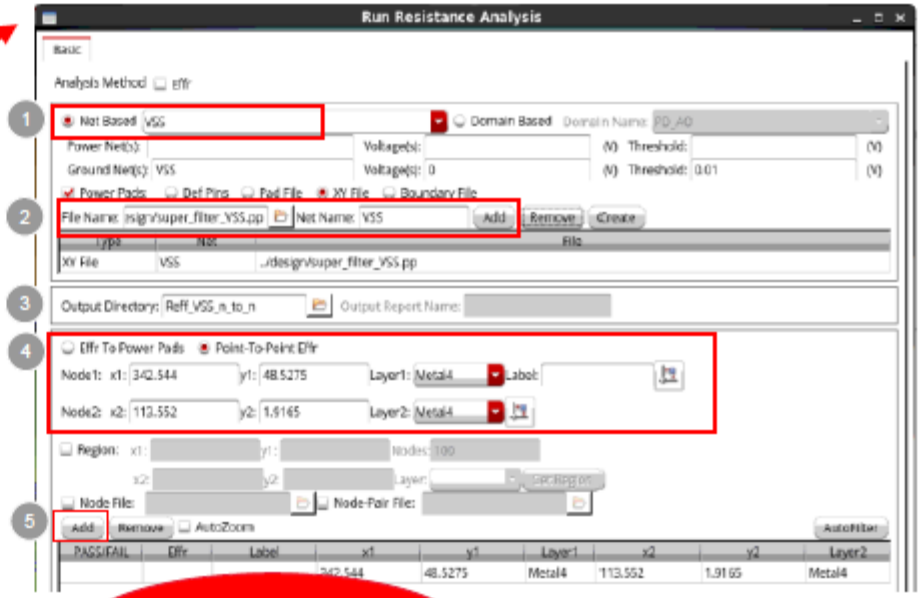
Instance: TDSP_CORE_INST0/MPY_32_INST/M16X16_INST/mul_8_14/g5477

```
report_inst_power TDSP_CORE_INST0/MPY_32_INST/M16X16_INST/mul_8_14/g5477 -out_file pwr.rpt
```

```
Instance: TDSP_CORE_INST0/MPY_32_INST/M16X16_INST/mul_8_14/g5477
Cell: ADDFXLMTR
Liberty file: /home/sakshin/voltus/voltus_labs/LIBS/LIB/scmetro_cmos10lp_rvt_ff_1p1v_m40c.lib
Internal power: 0.00175445mW
Switching power: 0.00062103mW
Leakage power: 0.00000013mW;
Total power: 0.00237560mW;

Direction Voltage(V) Duty      Density      Cap(pf)  Rise slew(ns) Fall slew(ns) Power(mW)  Net                                                    Pin
   Input 1.10000002 0.23330000 2.500000e+08 0.00125000 0.04039999 0.03630000          0  TDSP_CORE_INST0/MPY_32_INST/M16X16_INST/mul_8_14/n_671  A
   Input 1.10000002 0.49840000 1.966091e+08 0.00121900 0.04460000 0.04030000          0  TDSP_CORE_INST0/MPY_32_INST/M16X16_INST/mul_8_14/n_640  B
   Input 1.10000002 0.25999999 1.804418e+08 0.00144200 0.04290000 0.03810000          0  TDSP_CORE_INST0/MPY_32_INST/M16X16_INST/mul_8_14/n_619  CI
  Output 1.10000002 0.49959999 2.500000e+08 0.00285600 0.18449998 0.12949999 0.00043197  TDSP_CORE_INST0/MPY_32_INST/ab_result[12]  S
  Output 1.10000002 0.24609999 2.500000e+08 0.00125000 0.04929999 0.04210000 0.00018906  TDSP_CORE_INST0/MPY_32_INST/M16X16_INST/mul_8_14/n_673  CO

Leakage power
When  Duty  Power
 (((A) & (B)) & (CI))      0.0302319      0.0302319*1.14889e-10
 (((A) & (B)) & (!(CI)))      0.0860448      0.0860448*1.24213e-10
 (((A) & (!(B))) & (CI))      0.0304261      0.0304261*1.17852e-10
 (((A) & (!(B))) & (!(CI)))      0.0865972      0.0865972*1.10255e-10
```

Resistance analysis





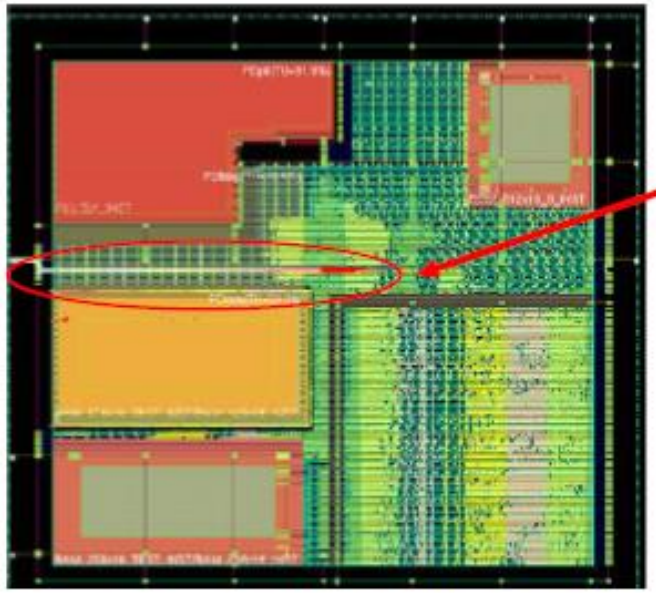

```
# Layer Representation: LEF
#      Coordinate Unit: um
#      Resistance Unit: Ohm

NET: VSS

# Threshold: N/A

# Effective resistance for given pair of nodes
# PASS/FAIL        REFF  LABEL          X1       Y1 LAYER1          X2       Y2 LAYER2
  -              15.0519  -          328.715   51.040 Metal1       113.552    1.917 Metal4
```

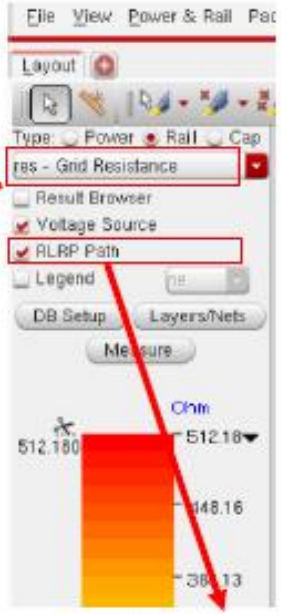
Effective analysis report

Tcl Command

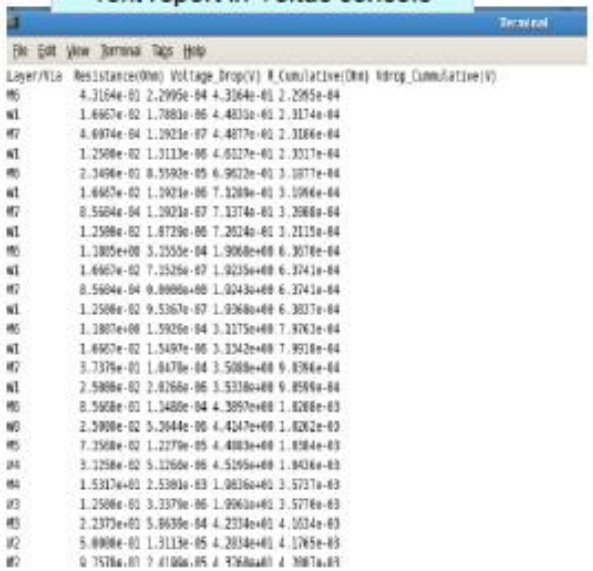
```
report_resistance \
  -net VSS -output_dir    ./Reff_VSS_n_to_n \
  -node_pair_list {{386.0615 110.619 Metal1 536.115 309.011 Metal 4 }}
```

Least-Resistance Path analysis



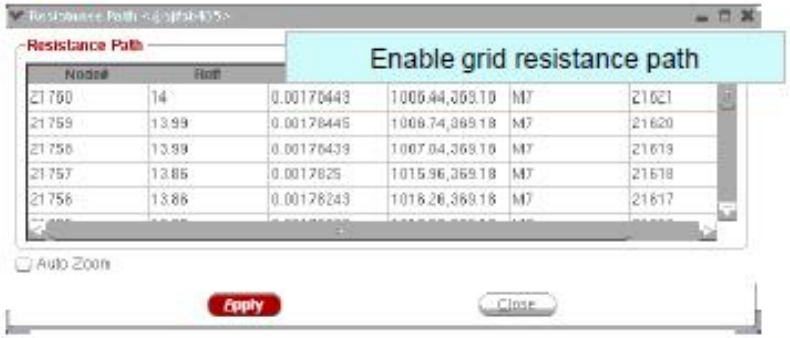


Text report in Voltus console

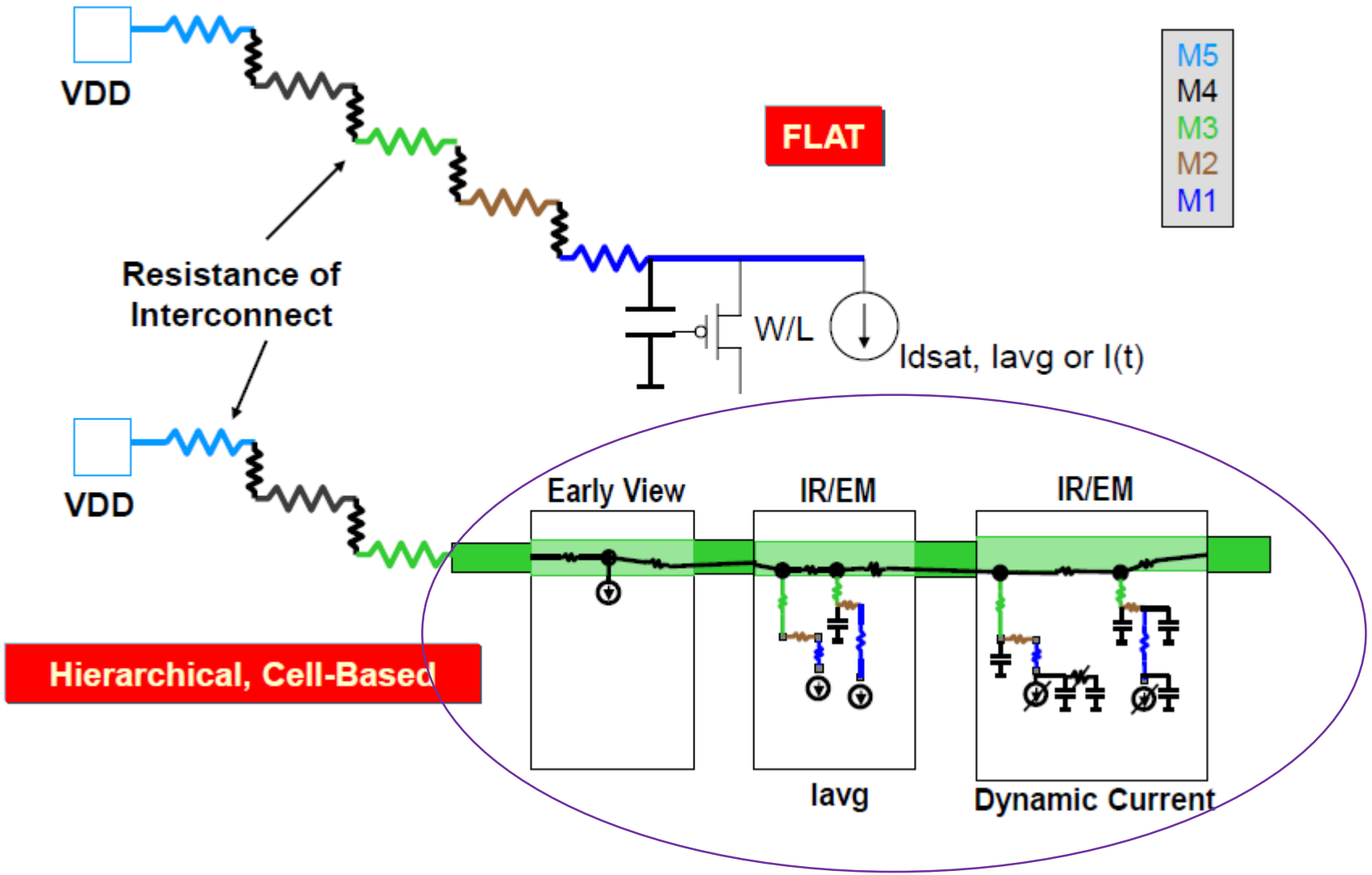


- Similar to *rlrp* analysis, resistance analysis can help to identify weak power grid
- Node-based and grid-based resistance plots
- Interactive trace of least-resistance path in form
- Detailed text report for all segments in LRP

Enable grid resistance path



Ir drop analysis



Power Grid Views (.cl)

Step1: generate power grid view (PGV = .cl file)

```
set_pg_library_mode \
-cell_type stdcells -extraction_tech_file RCgen.tch \
-lef_layer_map lefdef.map \
-power_pins { VDD 1.08 VDDO 1.08 VDDG 1.08 }  \
-ground_pins { VSS GND VSSG } \
-cells_file built_cells_list -temperature -25 \
-spice_models models/v1d1_usage.l \
-spice_corners { ss_lib } -spice_subckts { typical_max_m40.spice } \
-current_distribution propagation \
-power_switch_parameters {{HEAD8DMTR VDDG VDD {} {} {}}}

write_pg_library -out_dir stdcell_pgv
```

```
==========================================================================
Cell                PowerNets       Capacitance      PowerGrid Views
--------------------------------------------------------------------------------------------------------------|

BUFGX2MTR
                    VSSG(0.0000)     5.9e-16          EARLY(1 taps) EM(1 taps) IR(1 taps)
                    VSS(0.0000)     0.0e+00          EARLY(1 taps) EM(1 taps) IR(1 taps)
                    VDDG(1.0000)     7.7e-16          EARLY(1 taps) EM(1 taps) IR(1 taps)
                    VDD(1.0000)     0.0e+00          EARLY(1 taps) EM(1 taps) IR(1 taps)
 Cell Type = STDCELL                                    Cell_Status: PASS
--------------------------------------------------------------------------------------------------------------|
AND2X1MTR
                    VSS(0.0000)     6.9e-16          EARLY(1 taps) EM(1 taps) IR(1 taps)
                    VDD(1.0000)     6.9e-16          EARLY(1 taps) EM(1 taps) IR(1 taps)
 Cell Type = STDCELL                                    Cell_Status: PASS
--------------------------------------------------------------------------------------------------------------|
```

stdcells.report

Step2: Rail analysis

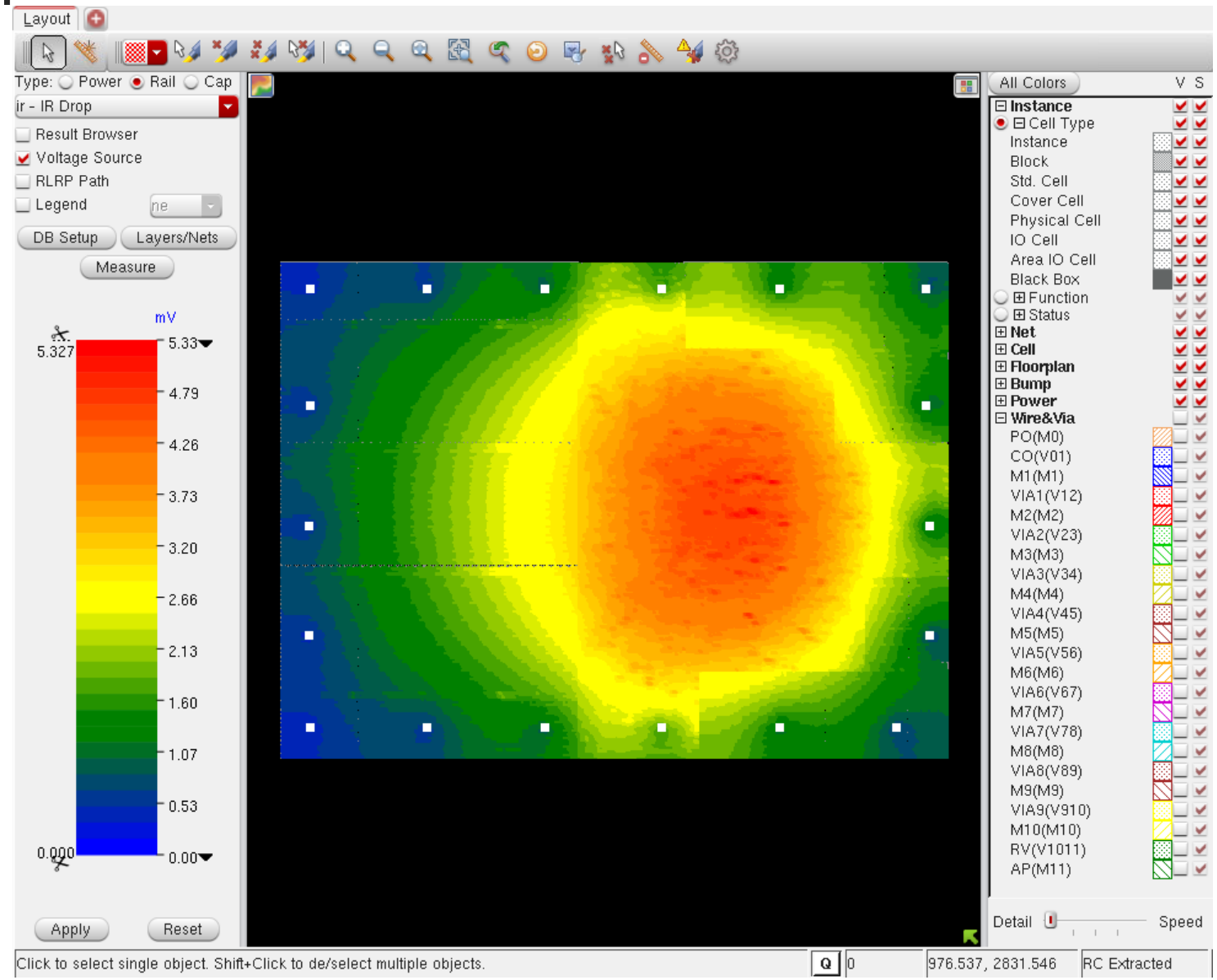
```
set_rail_analysis_config \
    -method static -accuracy hd \
    -power_grid_libraries {accurate_stdcells.cl TSDN65LPA1024X32M8F.cl} \
    -analysis_view AV_wc_on -voltage_source_search_distance 50        \
    -report_via_current_direction false                                  \
    -temperature 125

set_pg_nets -net VDD -voltage 1.08 -threshold 0.972
set_pg_nets -net VSS -voltage 0.0 -threshold 0.108

set_power_data -reset
set_power_data -format current -scale 1 {static_VDD.ptiavg static_VSS.ptiavg}
set_power_pads -reset
set_power_pads -net VDD -format xy -file ../DATA/vstorm/dma_mac-VDD.pp
set_power_pads -net VSS -format xy -file ../DATA/vstorm/dma_mac-VSS.pp

report_rail -type domain -output_dir ./AVallOn_PDCore PDcore
```

VDD.pp - /sprt/scrat

File Edit Search Prefere

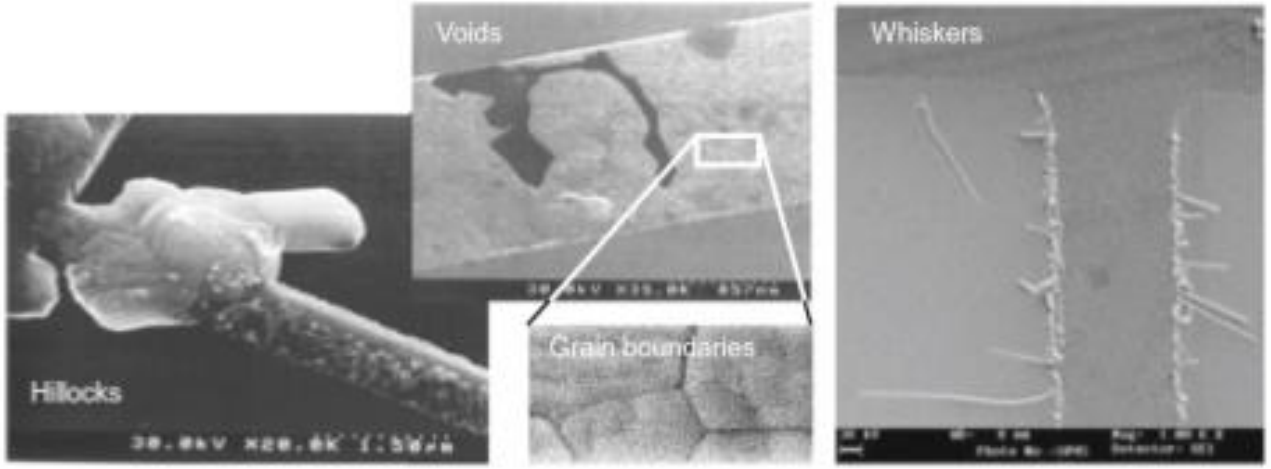
```
*vsrc_name   x    y    layer_name
VDDvsrc1 276.680 553.720 M2
VDDvsrc2 276.680 553.717 M1
VDDvsrc3 285.840 553.720 M2
VDDvsrc4 285.840 553.717 M1
VDDvsrc5 295.000 553.717 M2
VDDvsrc6 295.000 553.717 M1
VDDvsrc7 304.160 553.720 M2
VDDvsrc8 304.160 553.717 M1
VDDvsrc9 313.320 553.720 M2
VDDvsrc10 313.320 553.717 M1
VDDvsrc11 298.600 672.545 M1
VDDvsrc12 298.600 672.545 M2
VDDvsrc13 295.000 671.700 M3
VDDvsrc14 295.000 671.700 M4
VDDvsrc15 295.000 671.700 M5
VDDvsrc16 295.000 671.700 M7
VDDvsrc17 295.000 665.580 M3
VDDvsrc18 295.000 557.570 M3
```

Ir drop map



Electromigration

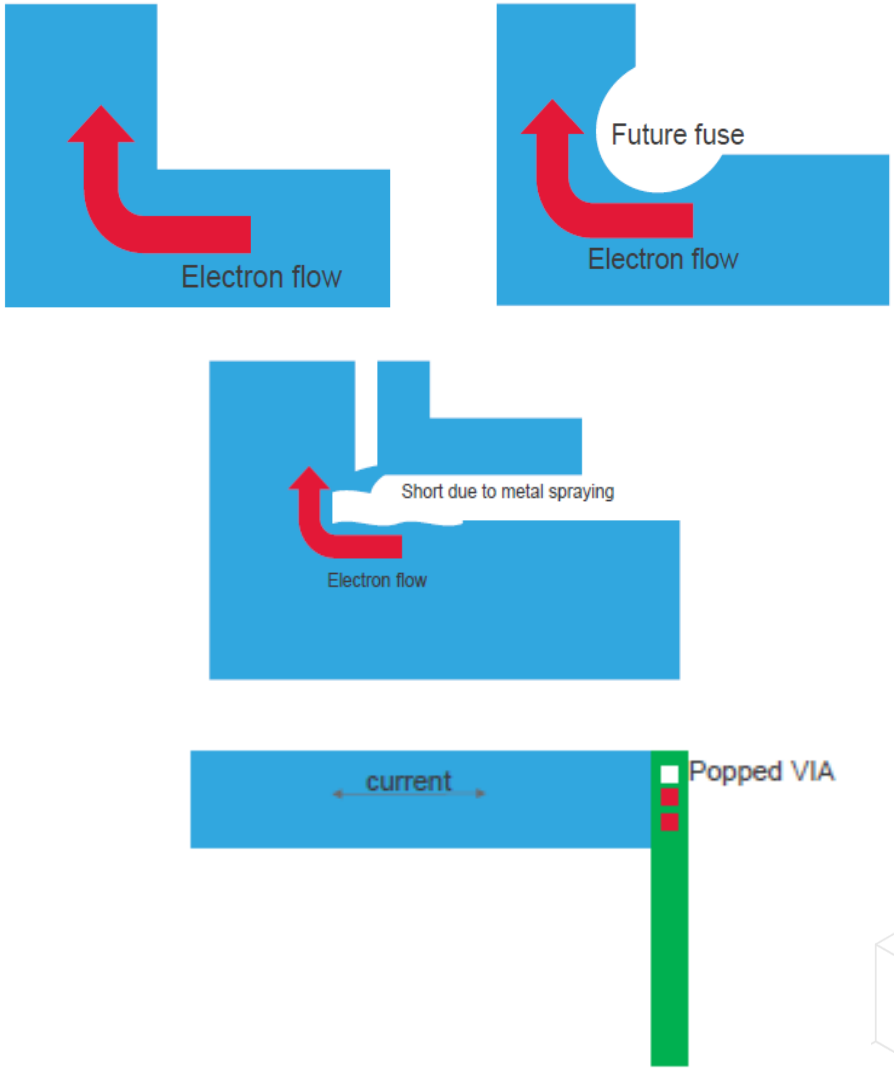
- If enough electrons collide with a metal lattice ion over a period of time, this can move in the direction of the electron flow, causing damage to the conductor, e.g. **voids** or **hillocks**



- With the scaling down of technology dimensions, power has not been scaled down proportionally, leading to high currents flowing through smaller and smaller interconnections, thus eventually causing the current densities to exceed the maximum allowed

Consequences of Electromigration

- The following failure mechanisms can occur:
  - If enough atoms move, the wire breaks and becomes an **open circuit**
  - If enough atoms move to the same location, a **short** to an adjacent metal wire is created
  - Joule Heating causing the via expansion in the dielectric layer can eventually end up in the popping of the via



Electromigration rules

Metal wiring level / interlevel connection	Via landing on Bottom metal	I_{max} (mA)
M1		0.95 x (w x 0.9 – 0.002)
Mx (1x metal)		0.92 x (w x 0.9 – 0.002)
My (2x metal)		1.8 x (w x 0.9 – 0.013)
Vx (drawn size 0.05 x 0.05 um2)	Landing on M1/Mx	0.04 per via
Vx rectangular via (drawn size 0.05 x 0.13 um2)	Landing on M1/Mx	2 x 0.04 per via

Temperature	105C	110C	115C	120C	125C
Rating factor of I_{max}	1.4	1.0	0.9	0.7	0.4

How to load EM Rules in Voltus?

Below are the different methods sorted by priority

- Use of qrcTechFile with embedded EM rules given by foundry
- Use of ICT EM rules
- Use of iRCX EM rules
- Use of Voltus native format EM rules

Check and update qrcTechFile

- Techgen (under Quantus™ QRC) is used to update the *qrcTechFile* with the EM model information.

- To update the *qrcTechFile* with the EM model information, use these commands in Quantus:
  - To dump ICT file from *qrcTechFile*:

```
Techgen -process_out <qrcTechFile>
```

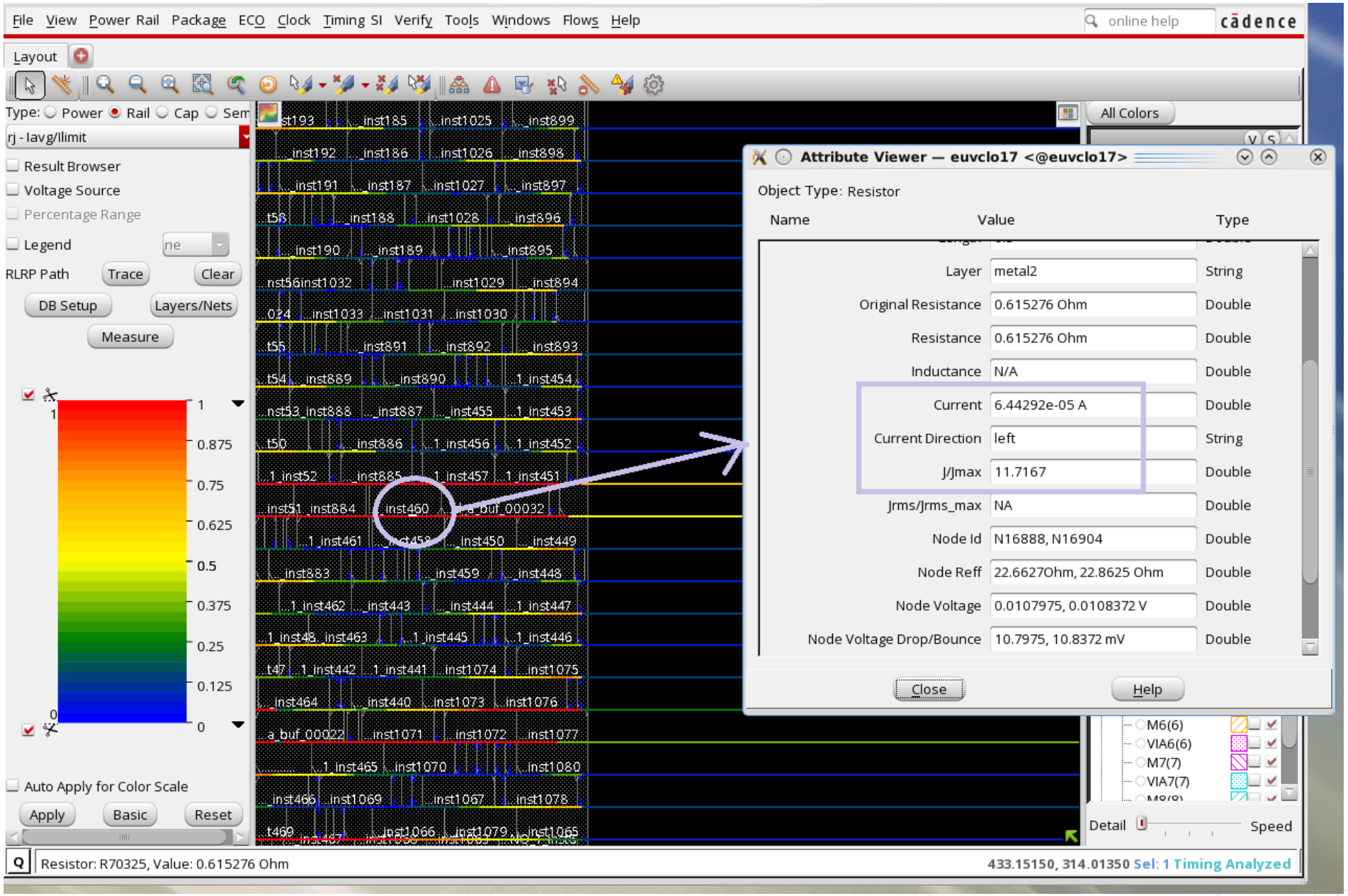
  - To dump info on encrypted qrcTechFile:

```
Techgen -tech_info <qrcTechFile>
```

  - To include EM rules from *.ICT* into *qrcTechFile*:

```
Techgen -update_em <qrcTechFile> \
ict_file_with_EM models
```

EM map

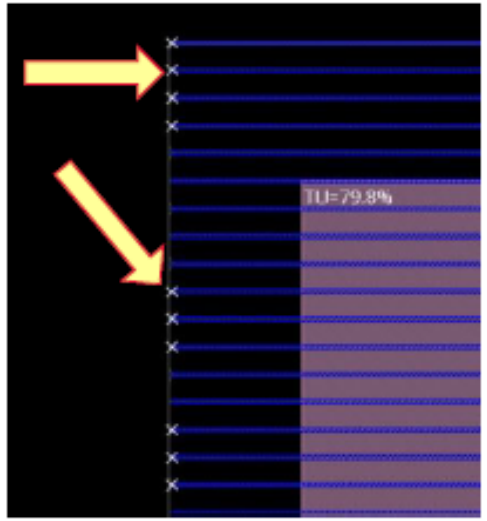


DRC / LVS

Early checks with Innovus: Verify connectivity

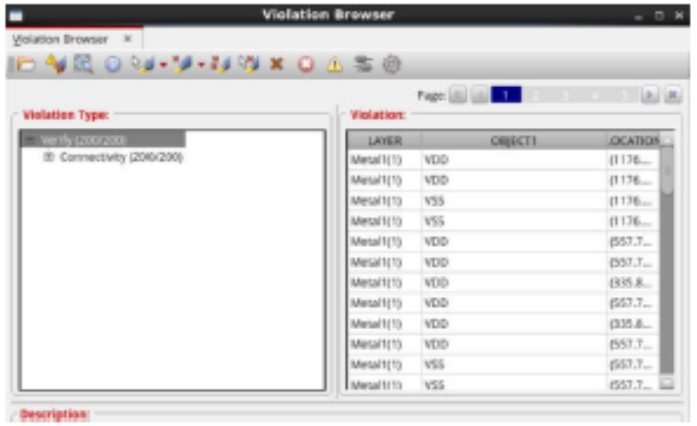
Choose **Check – Check Connectivity** to report open nets, antennas, loops, and partial routing, for all nets or specified nets in your design.

Example
Violation markers generated for open/unconnected rails.



Command
`check_connectivity`

Choose **Tools – Violation Browser** to view details of the violations and to zoom to selected violations.



Early checks with Innovus: verify DRC rules

Set the check_* attributes and run the check_drc command to verify DRC rules.

Syntax

```
set_db check_drc_ndr_spacing {true | false}
set_db check_drc_check_only {all | regular | special}

check_drc [-help] [-area {x1 y1 x2 y2}] [-view_window]
[-check_ndr_spacing] [-check_only {all | regular | special | cell}]
[-layer_range <string>] [-limit <integer>] [-out_file <string>]
```

Signoff Verifications with Pegasus

Pegasus features:

- DRC (Design Rule Check)
- XOR (Sometimes referred to as LVL - Layout Versus Layout)
- FastXOR (Fast XOR)
- ERC (Electrical Rule Check)
- PERC (Programmable ERC)
- LVS (Layout Versus Schematic)
- SVS (Schematic Versus Schematic)
- Device Signature Generator
- Sign-Off Metal Fill
- Layer Viewer

Running PVS from the command line

```
pvs -[flow] {options} rules_file1 ... rules_fileN
```

Example

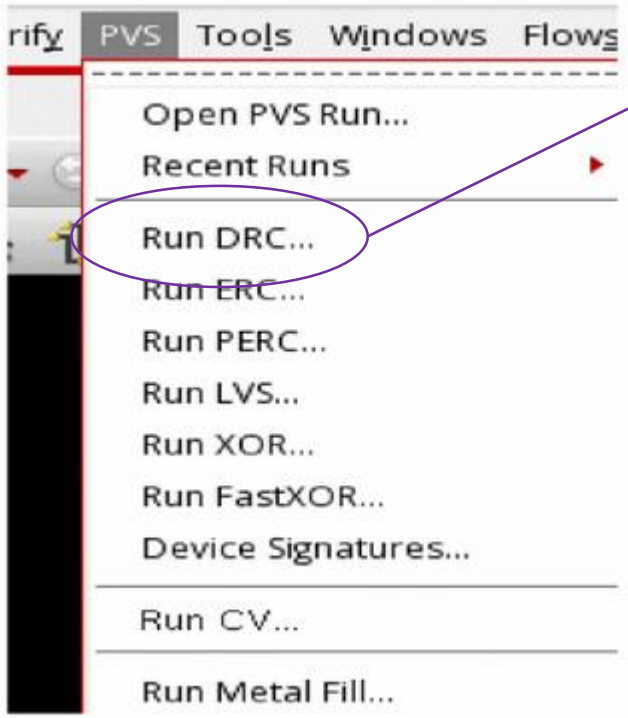
```
pvs -drc 65nmdrc.rul
pvs -lvs -ui_data -qrc_data 65nmlvs.rul
pvs -lvs -qrc_data 65nmlvs.rul optional.rul
pvs -lvs -gds test.gds -top_cell test -source_verilog top.v \
  -source_cdl stdcells.cdl \
  -source_top_cell test_sch 65nmlvs.rul \
  pvl_edtext.rul
```

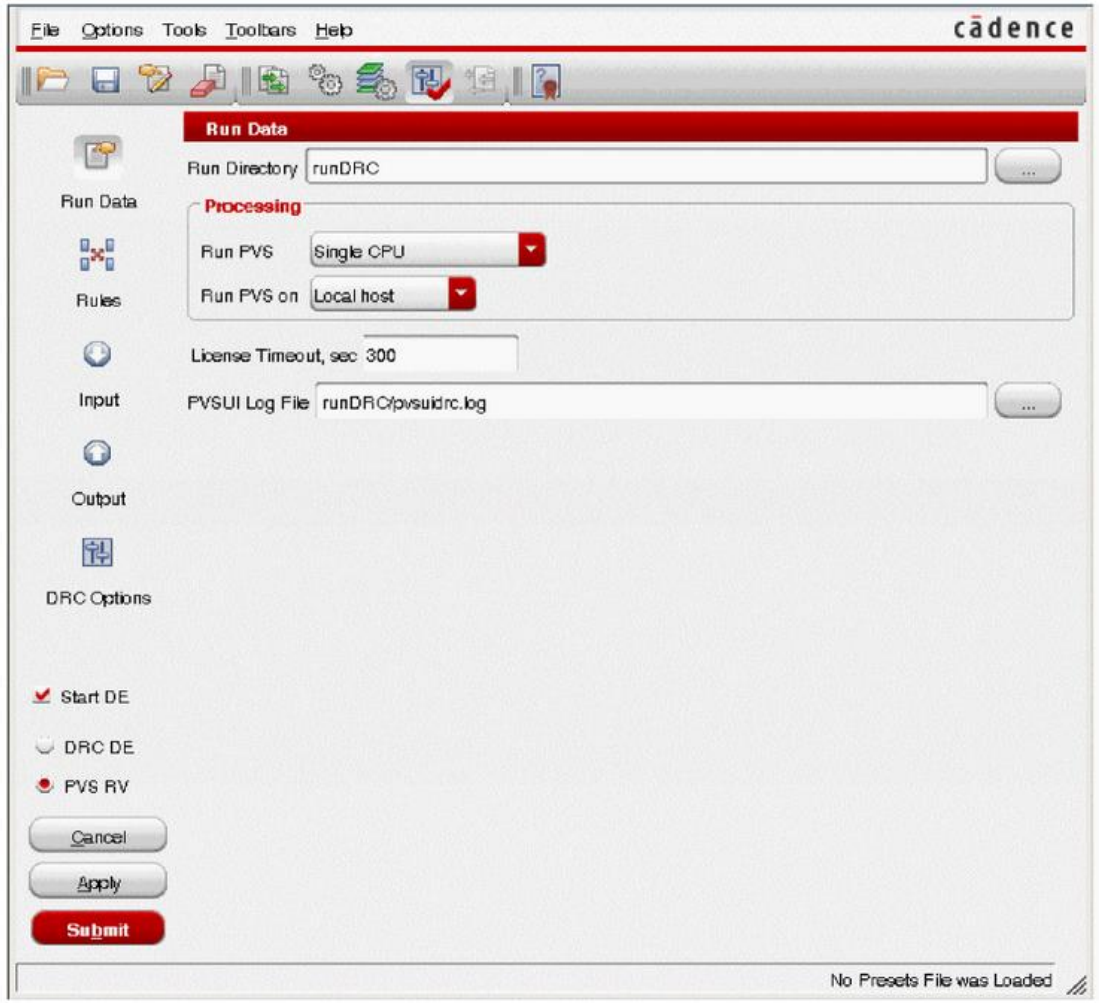
DRC rule file

```
++++++++++drc.rule+++++++++++++++

layout_primary "ecad";              ;; top layout cell
layout_path "./ecad.gds";           ;; gds file
layout_format GDSII;                ;; layout format

results_db -drc drc.ascii;          ;; contains drc output results

layer_def POLY 3;       ;;defines POLY as layer name for layer no 3
layer_def CONT 5;       ;;defines CONT  as layer name for layer no 5
layer_def MET 6;        ;;defines MET as layer name for layer no 5

;; Rule check POLY_CONT_SPACING

rule "POLY_CONT_SPACING" {
    caption check POLY to contact spacing < 3;
    exte POLY CONT -lt 3 -abut lt 90 -single_point -output region;
}

+++++++++++++++++++++++++++++++
```

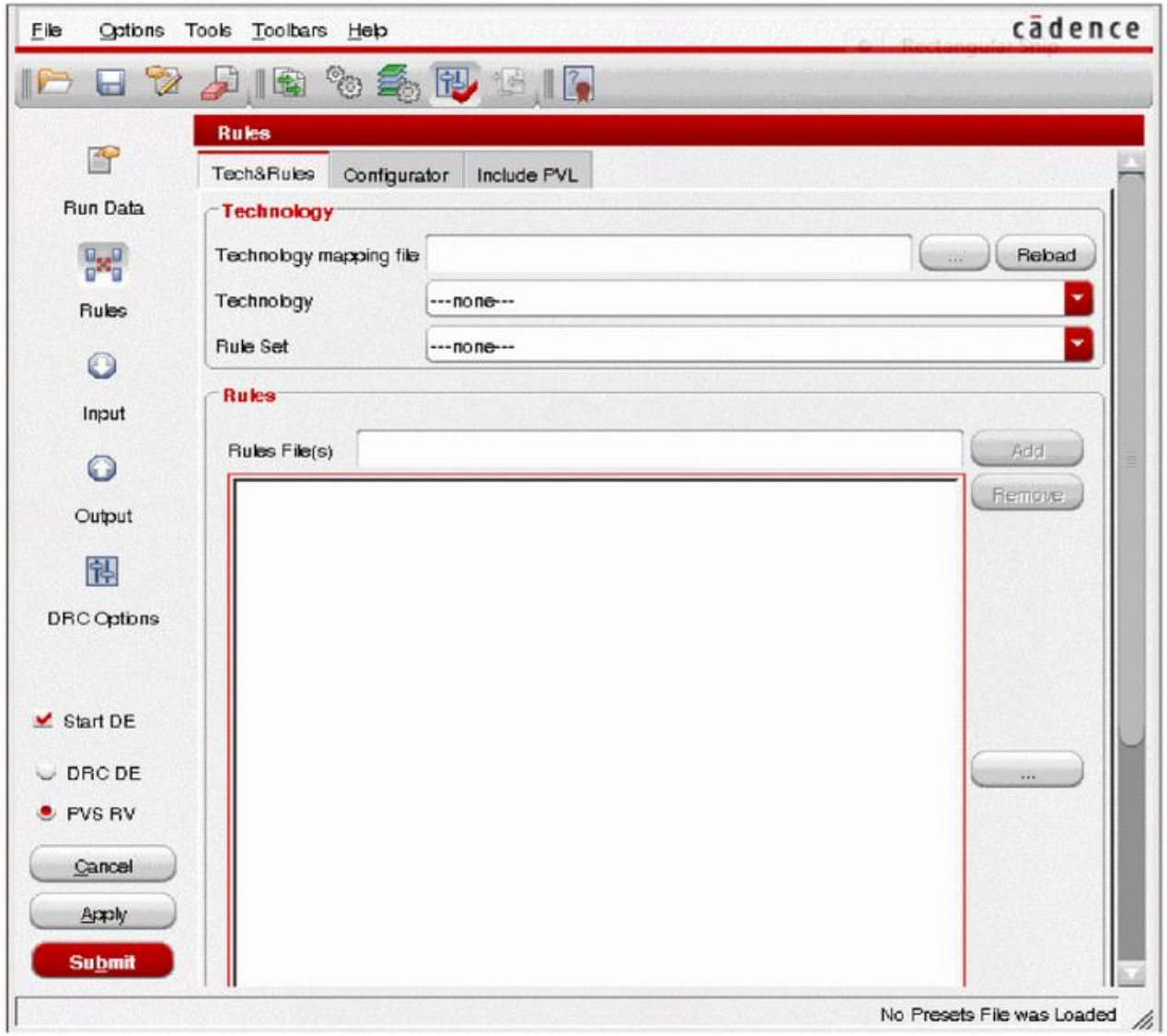
Running PVS from Innovus (DRC)

Innovus PVS menu

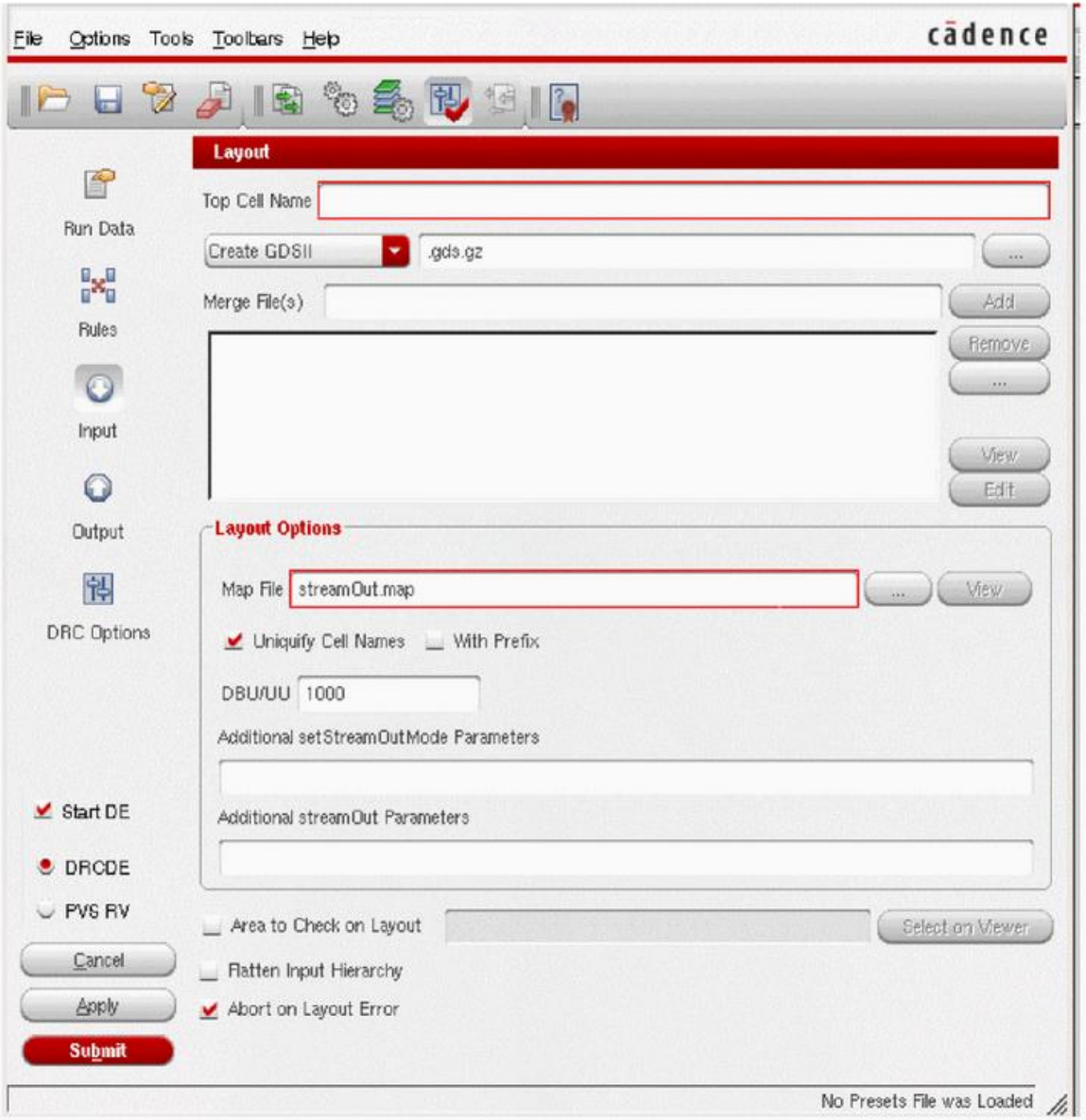




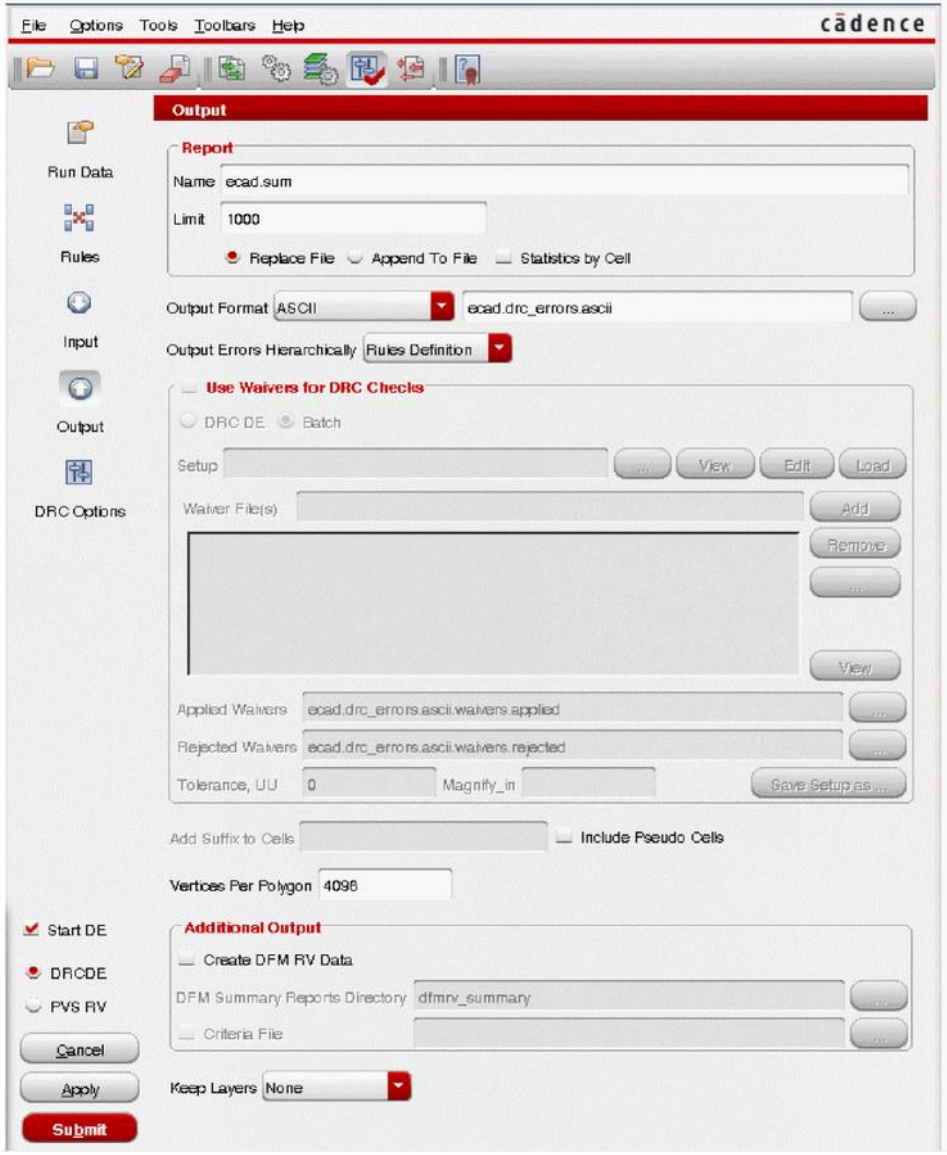
Running PVS from Innovus: DRC Rules



Running PVS from Innovus: DRC Inputs

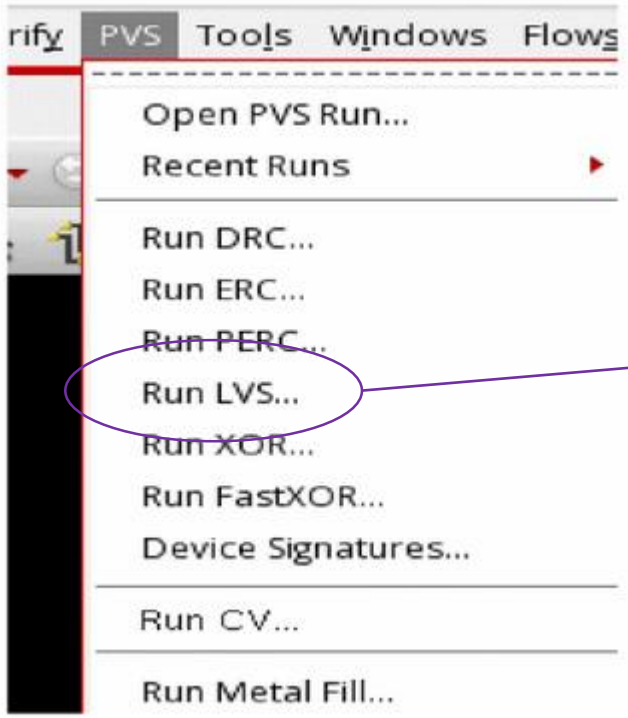


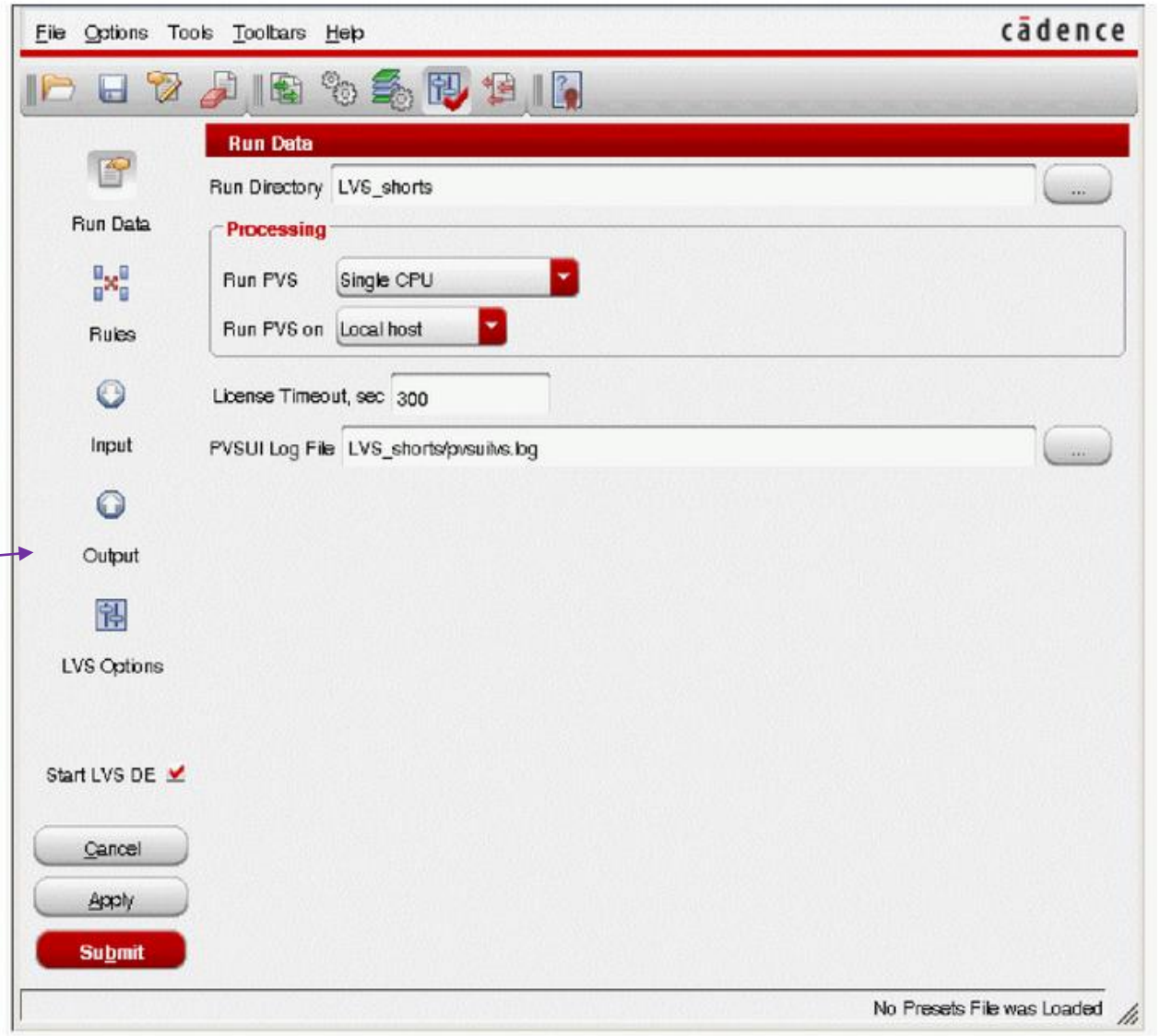
Running PVS from Innovus: DRC Outputs



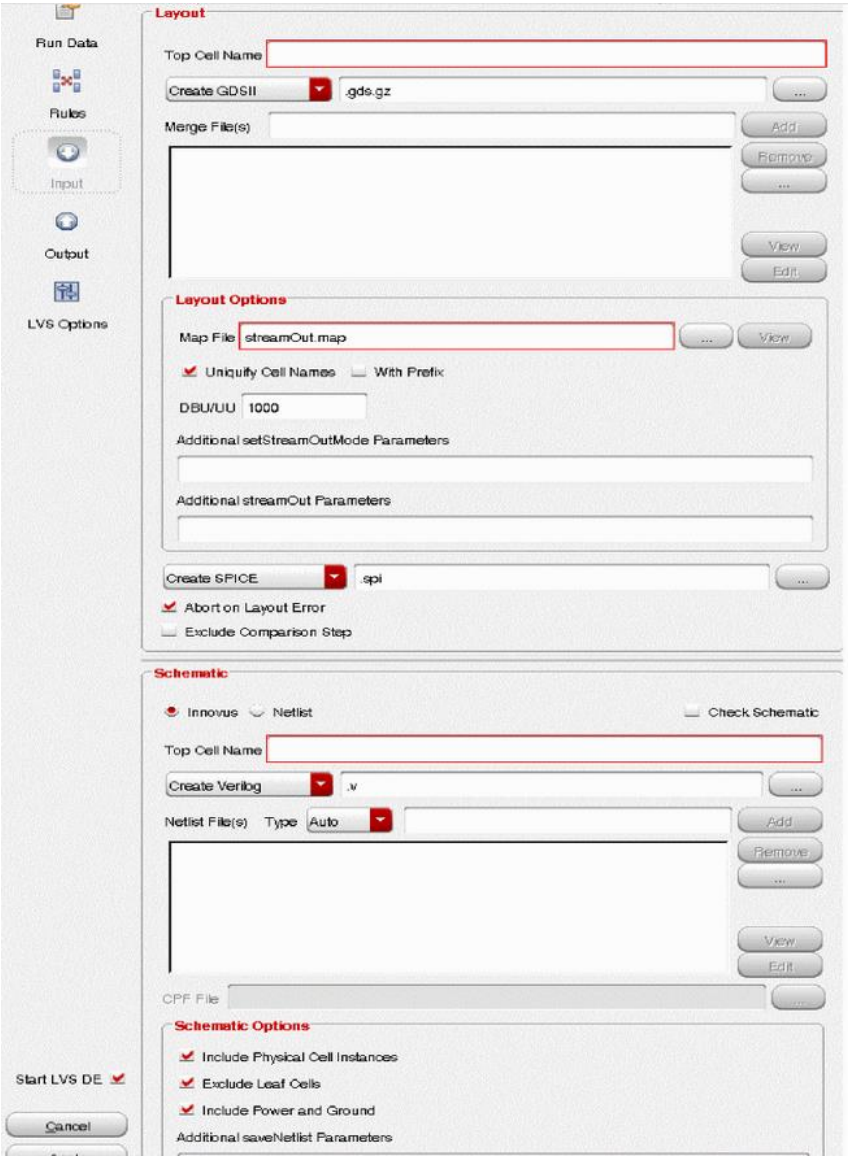
Running PVS from Innovus (LVS)

Innovus PVS menu

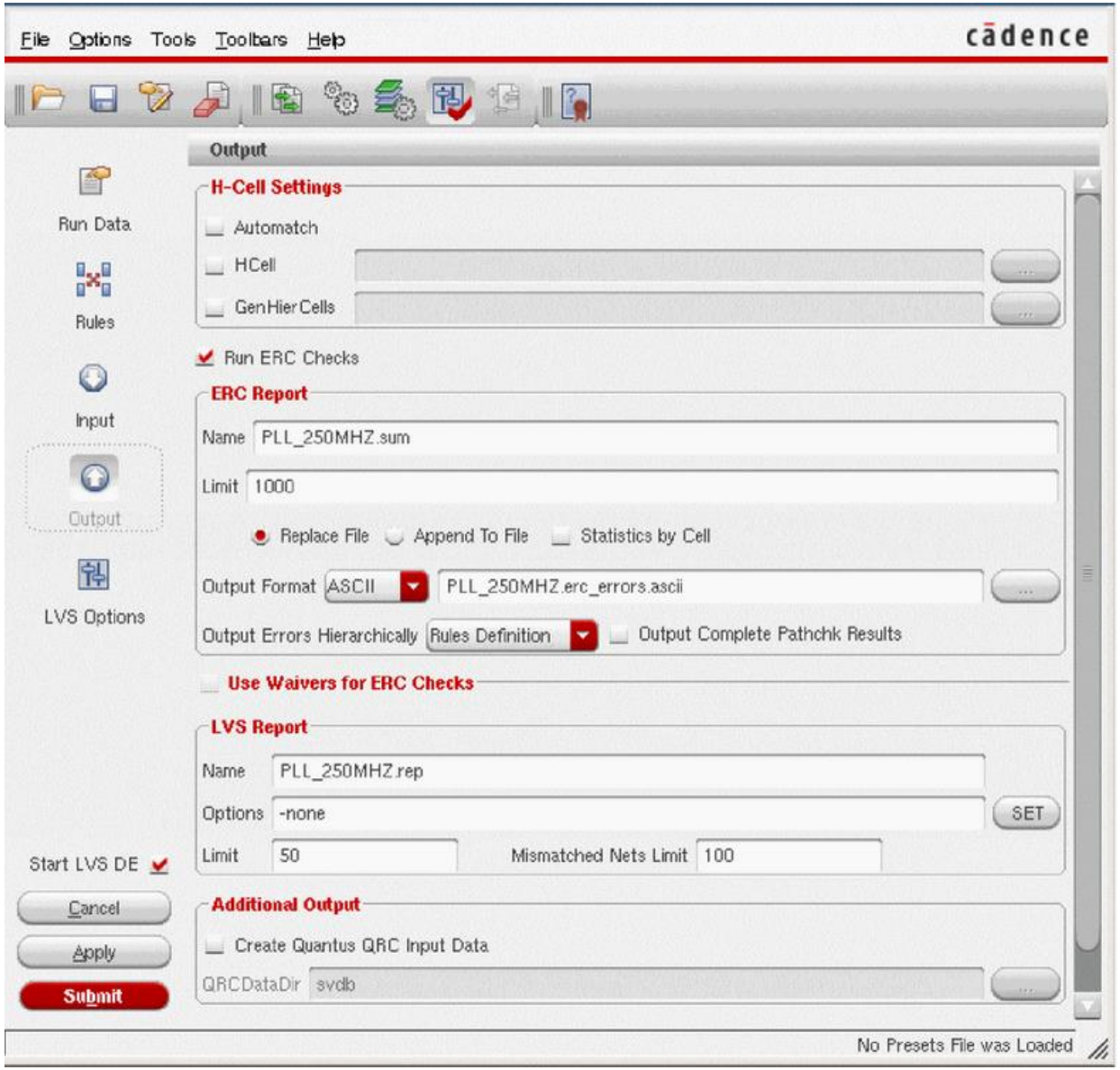




Running Pegasus from Innovus: LVS Inputs



Running Pegasus from Innovus: LVS outputs



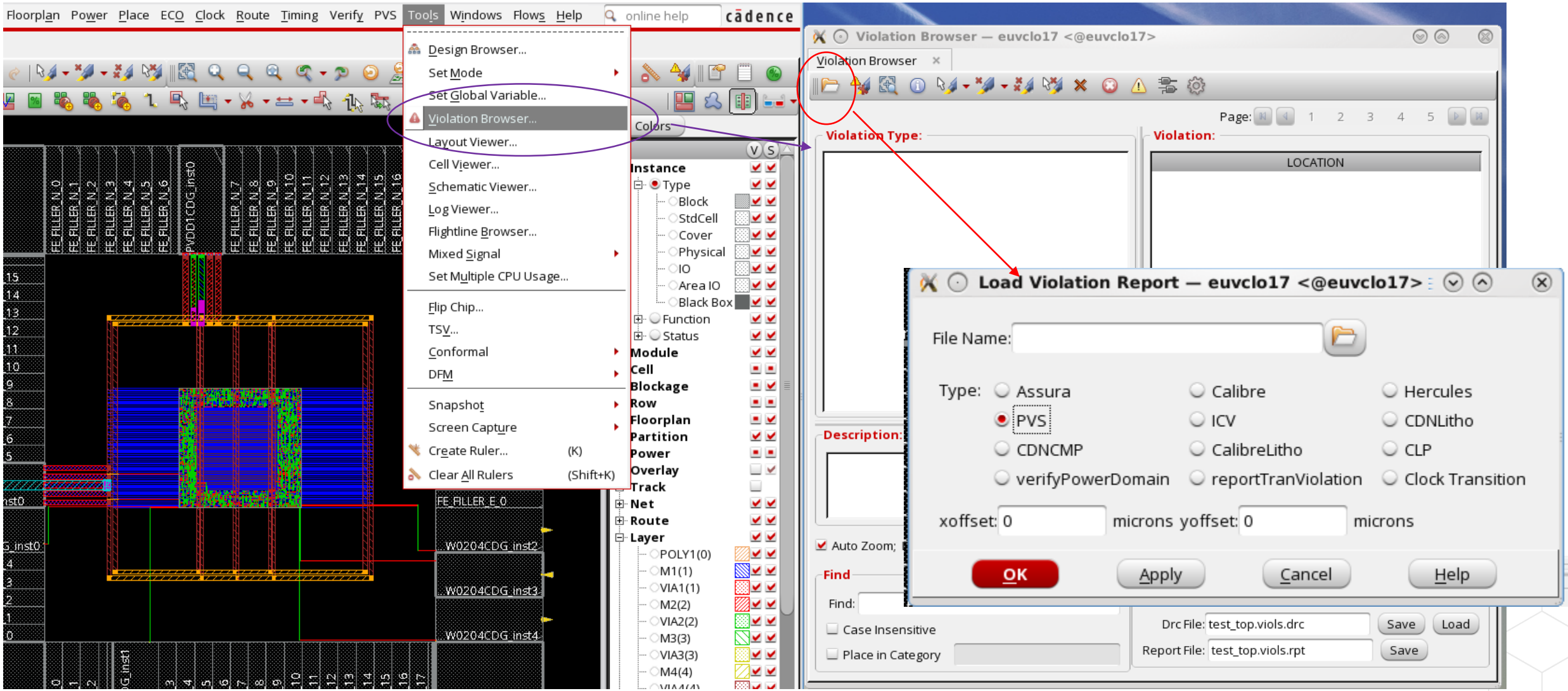
Main report window

This section shows log of the current job. Errors and warnings are automatically highlighted in this field:

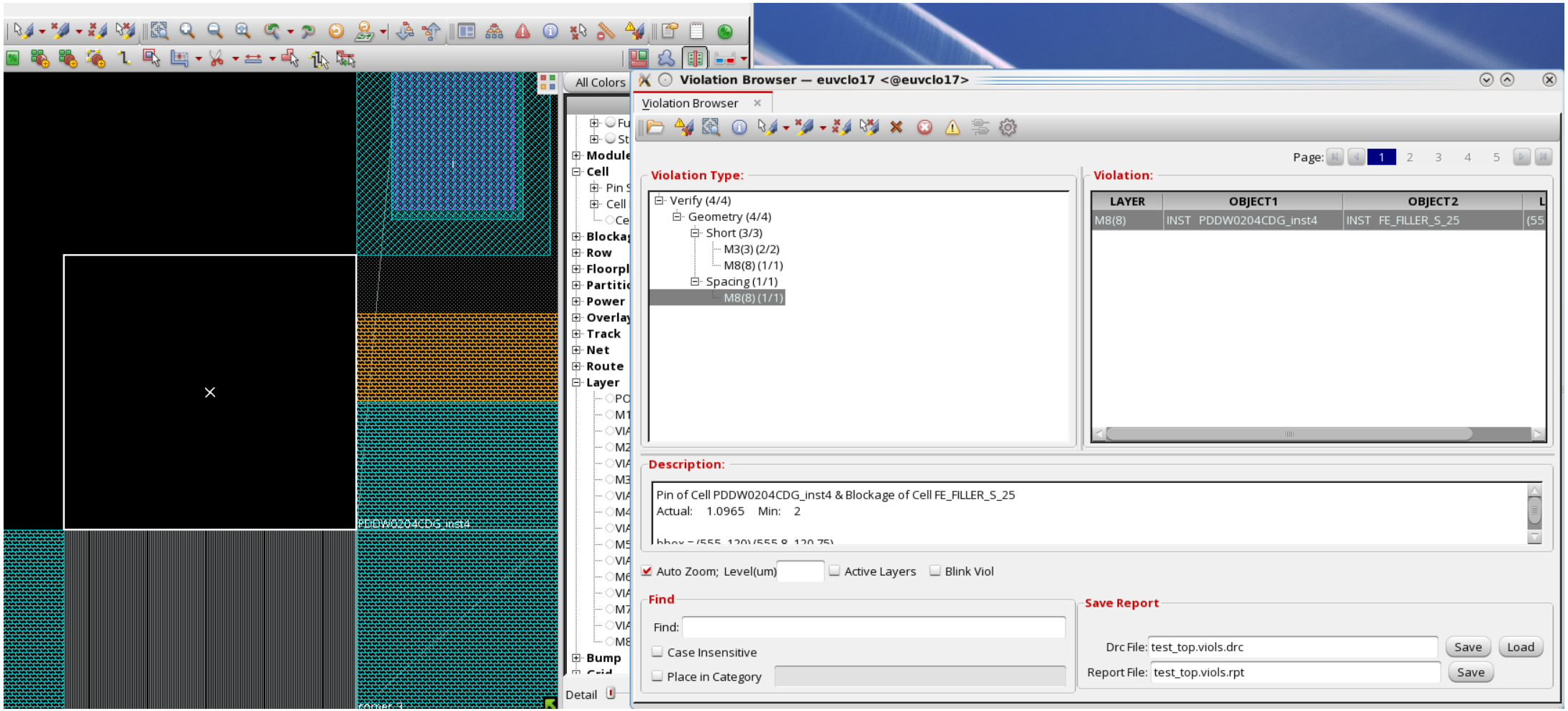
```
****** Start processing file 'lvs.rul.__pre__' ******
****** End processing file 'lvs.rul.__pre__' ******
LVS_REPORT_FILE "lvsrep";
[WARN]: LVS_REPORT_FILE is skipped, it is set in control file.
LVS_REPORT_MAX 1000;
[WARN]: LVS_REPORT_MAX is skipped, it is set in control file.
MASK_RESULTS_DB -none;
MASK_SVDB_DIR svdb -query -cci;
FLOW_DATA ui_data;
LVS_FIND_SHORTS yes;
```

```
 finished committing cdl netlists (0s)
ERROR (NVN-15200): schematic_primary "ecad" - cell 'ecad' does not exist in the schematic side.
Check the rule file and rerun it.

Generating the LVS report...

Command "pvsnvn  lvs.rul.rsf" was terminated with signal 11

[ERROR] pvsnvn terminated abnormally.
```

Load results in Innovus



Loads results in Innovus

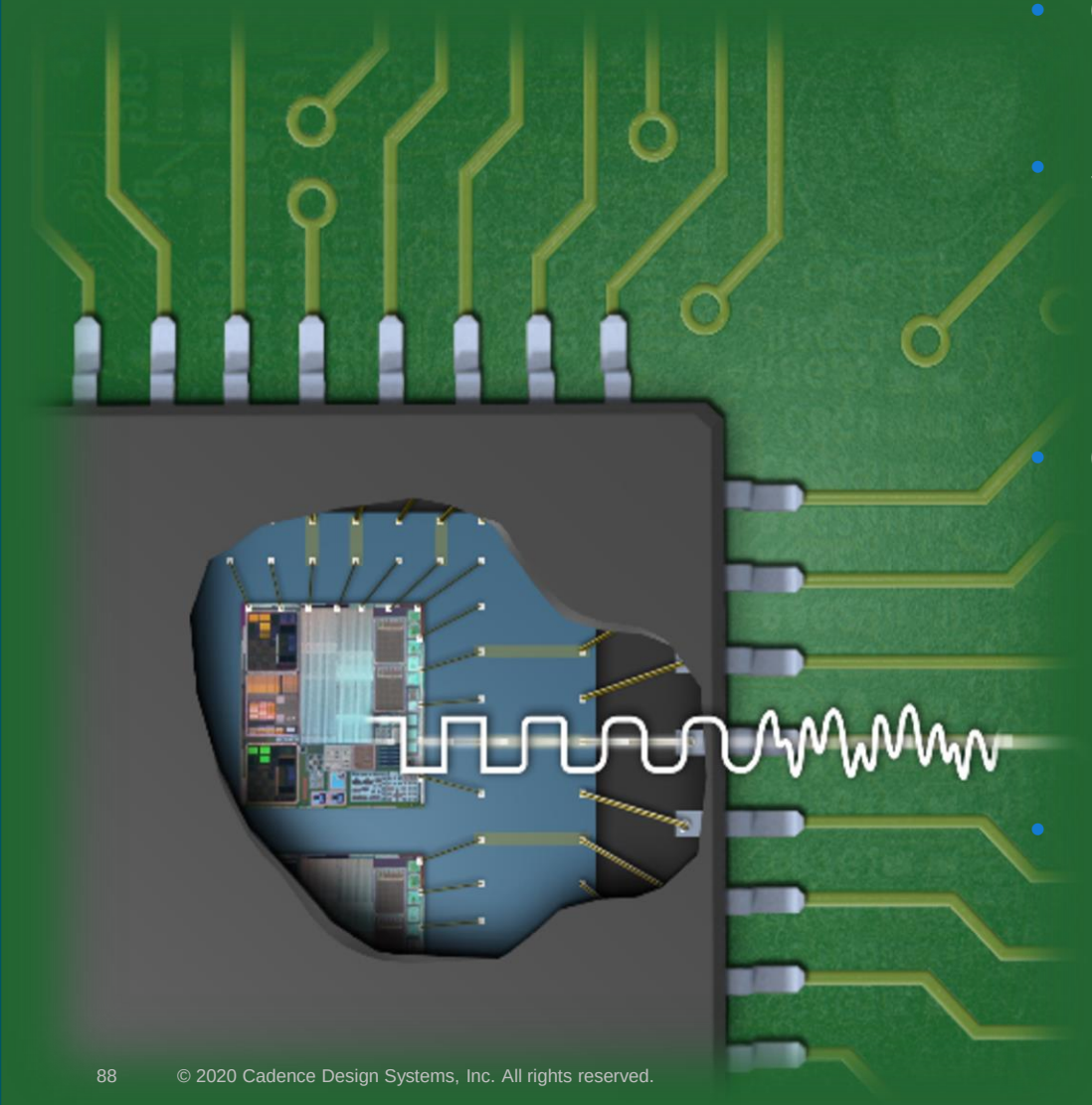


Simulation

Prerequisites for final simulation

- The final schematic has been laid out and has passed DRC and LVS checks and the parasitics for the block have been extracted.
- The RC extraction contains not only the designed devices and their connectivity but also a network of parasitic devices (Rs, Cs, Ls) that model the physical routing between the devices.
- Since the parasitic simulations can be time-consuming, these analyses are limited here to specific and necessary parameters, testbenches, and process corners.

Cadence Mixed-Signal Solution : AMS Designer



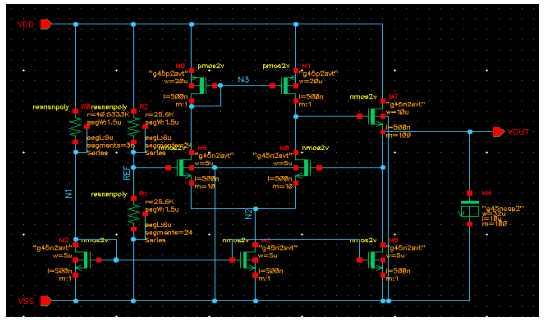
- Combine the power of the Spectre® and Xcelium™ simulators, now with AMS-Flex for easier version control
- Supports extensive set of languages that can be mixed and matched according to need
  - Spice, Verilog + A/MS, VHDL +AMS, SystemVerilog, Verilog AMS & SV RNM, C++ representations
- Common AMS simulator within *domain specific* environments
  - Use with Virtuoso® ADE Assembler for simulation management and distribution with an analog point of view
  - Use with Cadence SimVision™ Debug tools for full range of digital analysis
- Low power simulation with IEEE 1801

Easy Use Model (Virtuoso GUI)

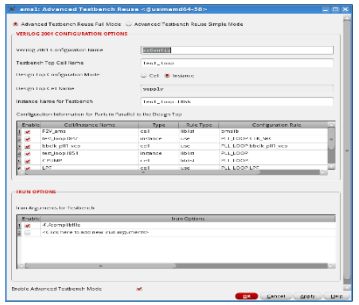
Starting from Scratch in 5 steps

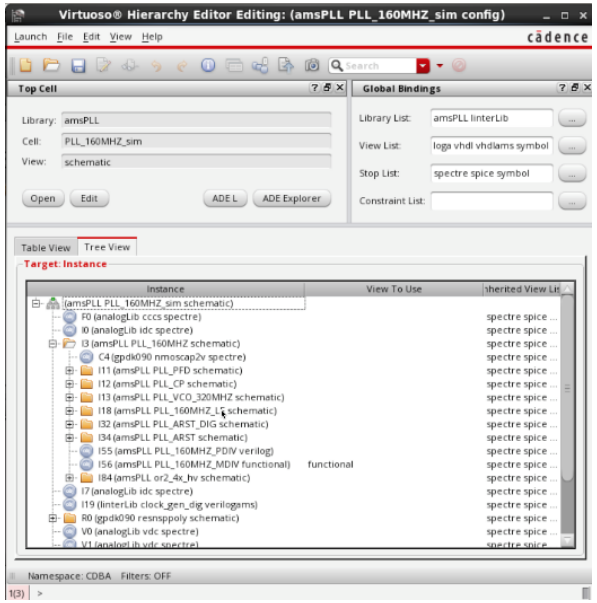
Schematic Creation → Maestro Creation → Partitioning Configuration → ADE Setup → Interactive Debugging

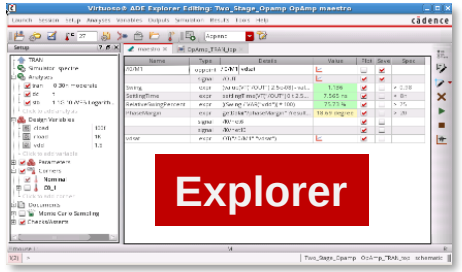
- Virtuoso Schematic Editor

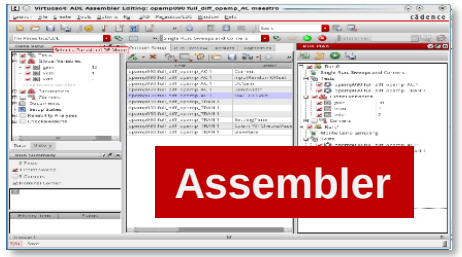


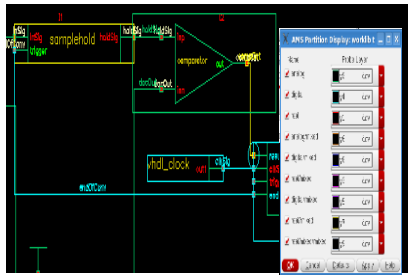
- Language text Importing into Virtuoso
- ADE Advanced TestBench (valuable for digital super top or UVM-MS)

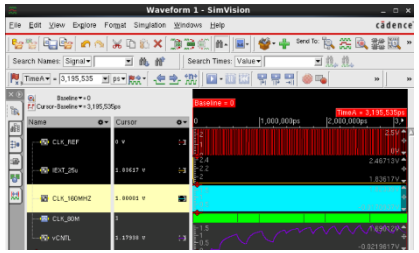






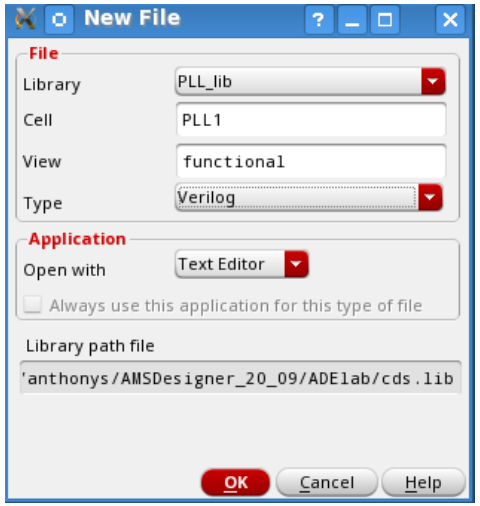


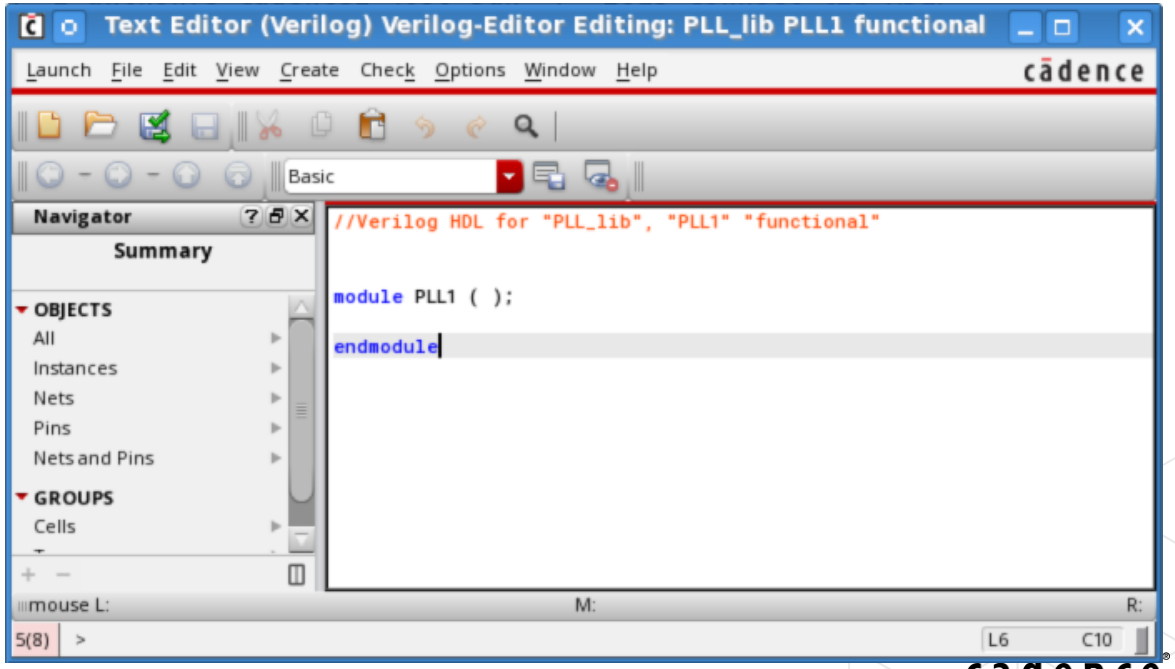




Importing Verilog into Virtuoso Using the New Cellview Form

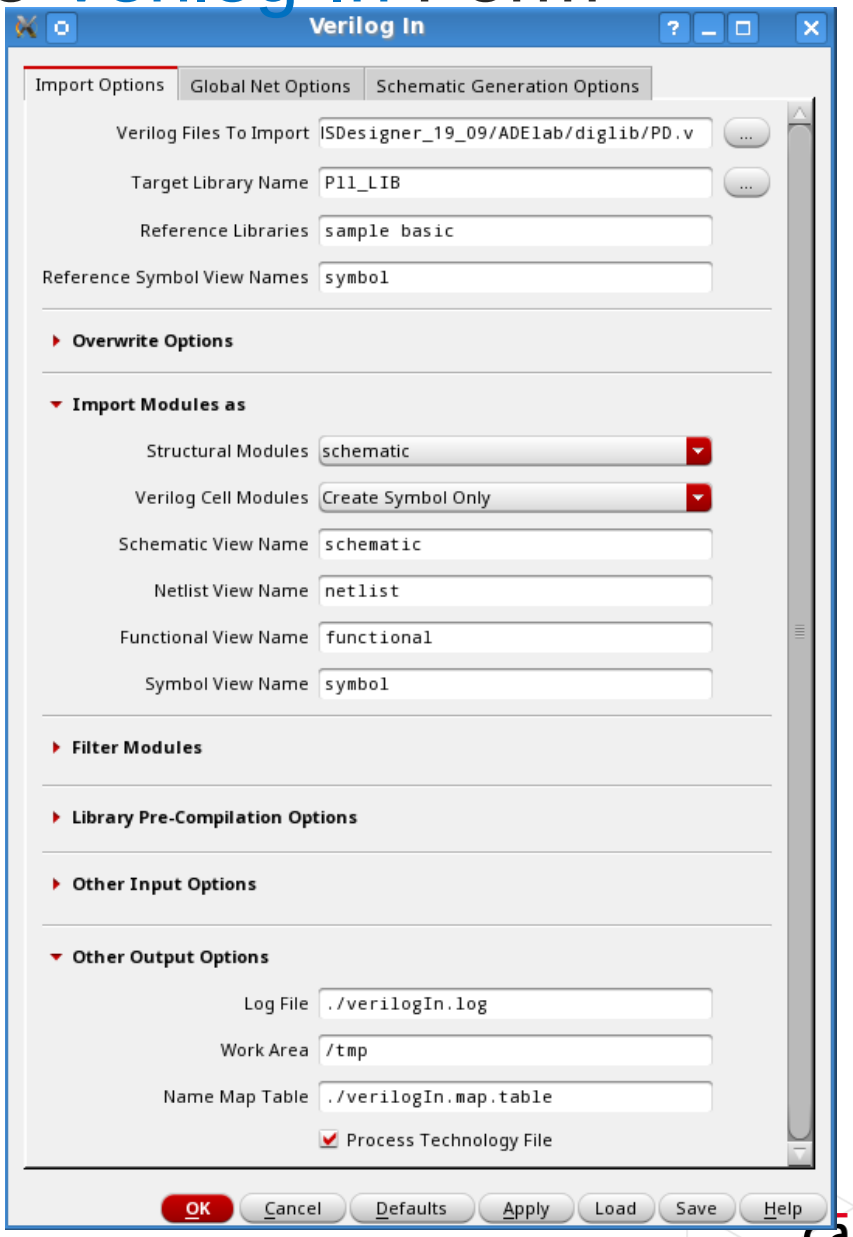
1. Create a new cellview in the Library Manager for the chosen design library.
   - You can set the *View* name to any name.
   - Choose Verilog as the *Type*.
   - Choose Text Editor as the *Application* to *Open with*.
2. When you click OK on this form, a Verilog text editor comes up. Copy-paste your Verilog text file here.
3. Once edits are done, save and quit this text file, which triggers compilation.
   - If a symbol doesn't exist, you will be prompted to create one. If it does exist, then port consistency is verified.
4. Once compiled, the file is copied into the DFII library structure and visible in the Library Manager as a **`functional`** (default name) view. You are also be notified of any errors in the CIW.





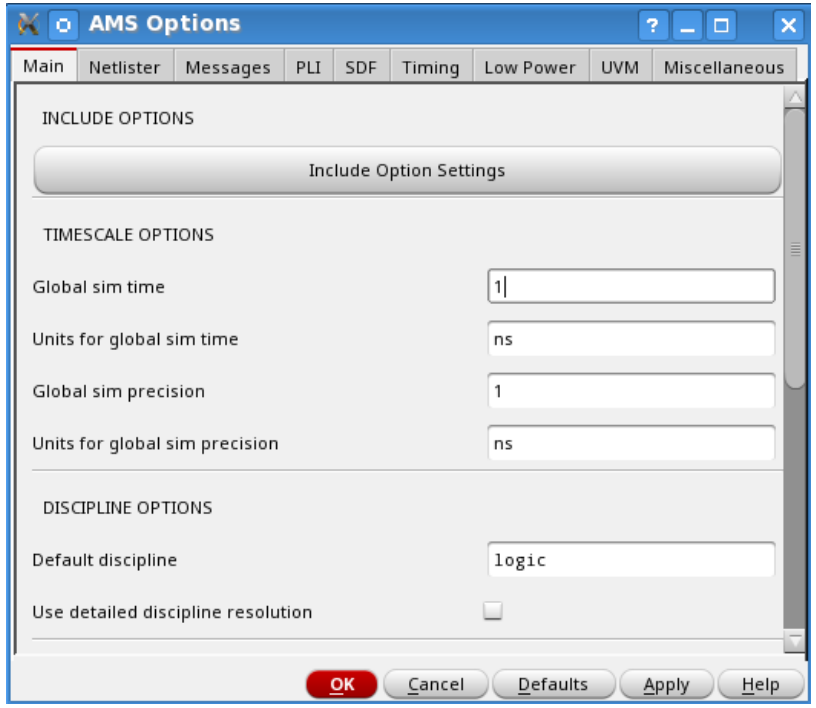
Importing Verilog into Virtuoso Using the Verilog-In Form

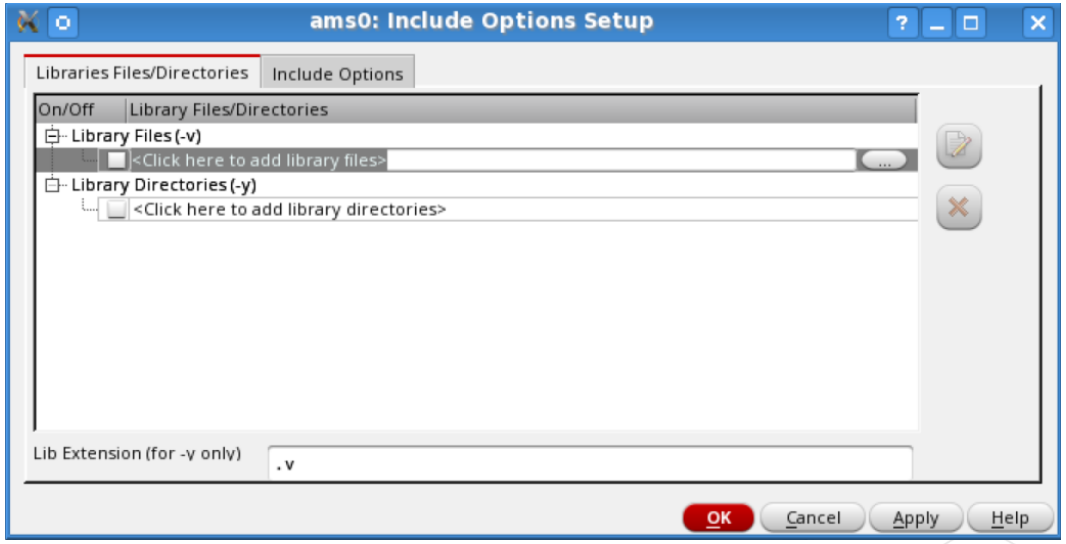
1. Go to the Command Interpreter Window (CIW) in Virtuoso and choose File – Import – Verilog to open the Verilog In form.
2. Browse and add your Verilog file to the Verilog Files to Import section.
3. Set the Target Library Name, and add Reference Libraries if needed.
4. Specify the Functional View Name for the Functional blocks.
5. Optionally, in the Global Net Options tab on the top of this form, enter the global net names for power/gnd used during schematic creation.
6. Click OK to trigger compilation, you can review a log file if you choose to.
7. View the new cell view that is created in the Library Manager.



Referencing Text Files Outside DFII: Using –v/–y

1. From the ADE, go to Simulation – Options – AMS Simulator to open the AMS Options form.
2. Click the Include Option Settings button to open the Include Options Setup form.
3. In this form, you can browse to and add the required files or library.





Referencing Text Files Outside DFII:
Using Library Compilation Options (–makelib/-reflib)

Use the –makelib option to precompile design units in the specified files into a reference library.

Using the –makelib Option:

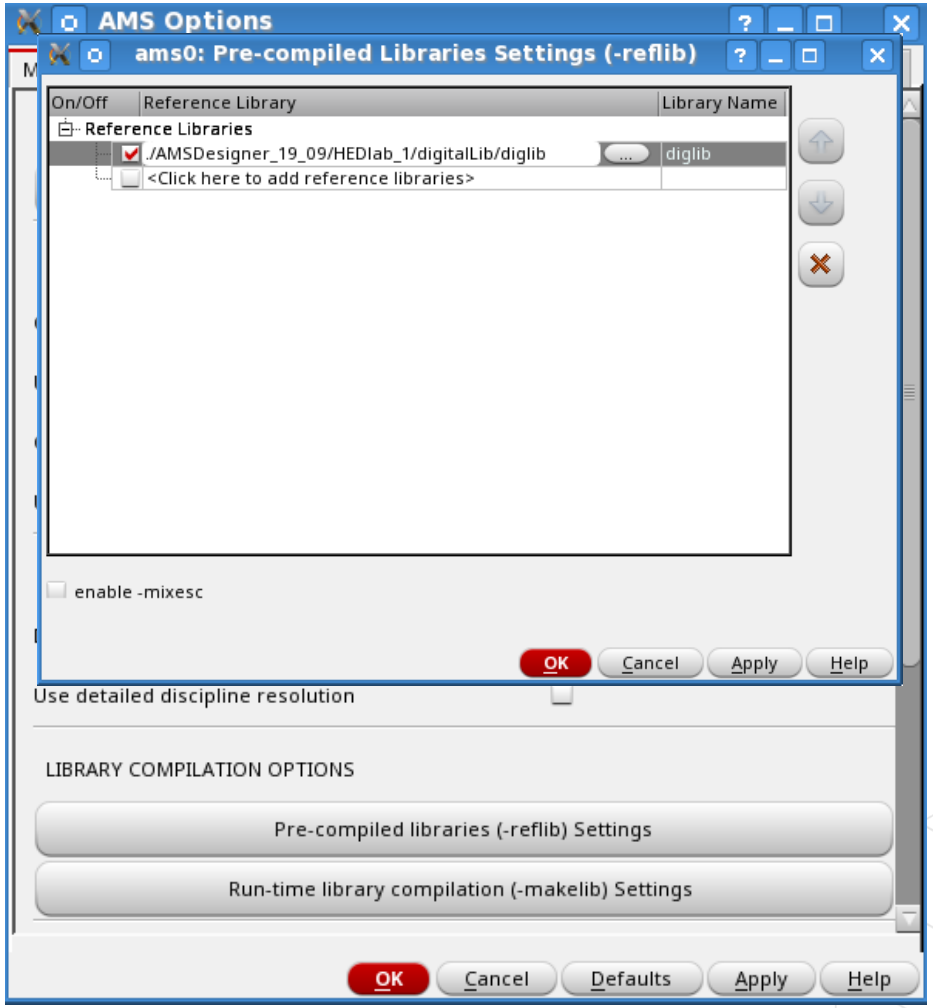
- You can pre-compile “golden” digital design units into a reference library using the `–makelib` option.
  - When top-level design files are compiled, the reference library is scanned for components instantiated in the design.
- This pre-compiled library is reusable and you can include the compiled cells in your mixed-signal simulations.

Referencing Text Files Outside DFII:
Using Library Compilation Options (–makelib/-reflib)

Using the –reflib option, you can specify the library directory containing precompiled source files that can be referenced during simulation.

Using –reflib Option:

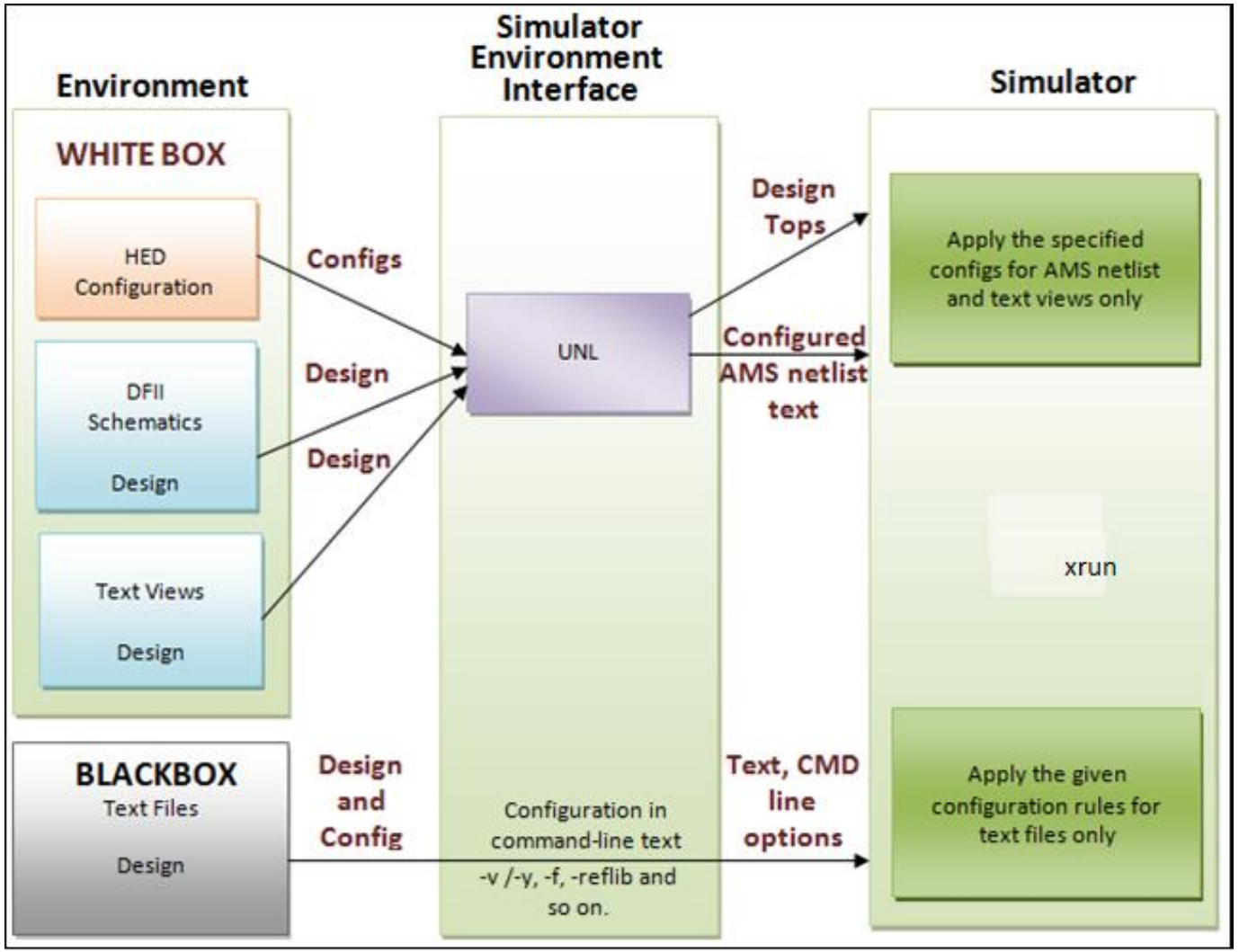
- Clicking the Pre-compiled libraries (-reflib) Settings button opens a form with a similar name.
- Browse and specify your precompiled library in the provided field.



Important: When using `-reflib`, we assume that the text libraries were pre-compiled external to Virtuoso using the *`xrun -makelib`* command.

WDUs and BDUs: Example

The ADE AMS UNL flow comprises a netlisting phase using the Unified Netlister, followed by the binding phase during elaboration and then finally simulation using the xrun executable.



BDUs and WDUs are considered together during elaboration and the complete design config is simulated.

Day 3 Conclusion

What we did cover

- Chip assemble, AoT, DoT
- STA analysis for mixed signal design
- Power analysis
  - Power
  - Resistance
  - IRDROP
  - EM
- Physical checks
  - DRC
  - LVS
- Simulation

To go deeper on part 3, useful labs

Static Timing Analysis (Signoff Timing Analysis) using Tempus 20.1 (08 Oct 2020)

Static Timing Analysis on Schematic-based Mixed-Signal Design (Common UI) (29 Jul 2022)

Power and Rail Analysis Using Voltus 20.13 (Signoff Power Analysis) (15 Mar 2021)

Running Pegasus DRC, LVS within Innovus Implementation System (21 May 2021)

Pegasus Hierarchical Metal Fill Flow in Innovus: Overview (05 Mar 2021)

ECO on Schematic-based Mixed-Signal Design - Stylus Version (29 Aug 2022)

cādence®