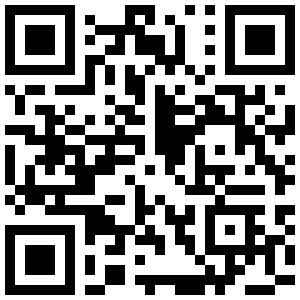
UE20EC313 Digital System Design  
Cadence Tools Command Reference

*(For any edit suggestions to this document, please mail-* [saigovardhanmc@pesu.pes.edu](mailto:saigovardhanmc@pesu.pes.edu) *or* [samahithsa@pesu.pes.edu](mailto:samahithsa@pesu.pes.edu) *)*

*Use the QR code below / web link* [*http://bit.ly/cadencelabpesu*](http://bit.ly/cadencelabpesu) *to access this document*



# Table of Contents

[Getting Started on Linux for Cadence Lab](#_p1az0nsxwdz3)

[Simulation Tool (ncverilog)](#_ldja3otk8nxk)

[Checking Code Coverage in ncverilog using IMC (Incisive Metrics Center)](#_4qilqb1f5ui4)

[Synthesis Tool (genus)](#_hv7kzmc6k0co)

[Syntax for writing the Constraints file (.sdc)](#_d2var5thbz2m)

[Equivalence Checking](#_p83yu69633tt) (conformal LEC)

[Linting](#_r31zh48l3ka3)

Static Timing Analysis ([Tempus](#_ttweb63331hj))

[Additional References](#_pl0pxa2m65tl)

# Getting Started on Linux for Cadence Lab

> create a folder in the desktop, with your srn/name

> open the folder

> right-click and create files for design and testbench,   
eg. db\_fsm.v and db\_tb.v

> right-click on the files and open them using *gedit*, save the design and testbench codes in the respective files

> right-click inside the folder and select *open in terminal*

> enter the following commands in the terminal

| csh |
| --- |

Enters the C-Shell

| source /home/<install location>/cshrc |
| --- |

Navigates to the Cadence Tools install path and starts the tool

Note: You can use the upper arrow in the terminal to navigate quickly to the already used paths/commands and use tab-key to auto-complete commands.

> A new window appears that welcomes the user to the Cadence Design Suite, the following tools can be invoked in this window.

# Simulation Tool (*ncverilog*)

>To start reading the design and testbench files, to obtain a waveform in the Graphical User Interface (*simvision),* enter the following commands.   
Note: No space between +access and +rw, but mandatory space between +rw and +gui. (make sure to follow all similar spacing patterns given in the tool reference)

| ncverilog <design> <testbench> +access+rw +gui |
| --- |

eg. ncverilog db\_fsm.v db\_tb.v +access+rw +gui

Note: the +gui starts up the *ncverilog* GUI window.

> navigate through the design hierarchy and select the signals you want to analyze in the design browser (hold down ctrl-key while selecting), right-click and select *send to waveform*

> in the *simvision* window, select the play button, followed by the pause button to start and stop the simulation. The simulation will end automatically if the $finish statement is executed in the HDL.

> select the ‘=’ symbol at the top right corner of the window, to fit the waveform’s entirety in the same frame.  
> drag the red marker to the beginning of the waveform and select on the ‘+’ symbol on the top right corner, to magnify until the waveform pulses are visible for verifying the functionality of the design.

# Checking Code Coverage in ncverilog using IMC (*Incisive Metrics Center*)

| ncverilog design.v tb.v +access+rw +gui +nccoverage+all |
| --- |

> Check for the path of the file “cov\_work” generated in the terminal then type:

(Invoke Incisive Metrics Center)

>enter the command ‘imc’ in the terminal which will launch the IMC GUI.

| imc |
| --- |

> In he IMC’s Graphical User Interface, you can navigate and select the file to check the Code Coverage (block, branch, expression, toggle) and FSM Coverage, represented in percentages.

# Synthesis Tool (*genus)*

| genus -gui |
| --- |

Opens the genus tool with gui, alternatively you can show and hide gui using command gui\_show and gui\_hide

| read\_libs <*path of .lib file>* |
| --- |

Reads library file for synthesis, from the specified path. Eg. saed90nm\_typ.lib the 90 nanometer typical library

| read\_hdl <*path of design file>* |
| --- |

Reads design file to be synthesized, written in HDL (eg. verilog, systemverilog)

| elaborate |
| --- |

Elaborates the design in the tool, and can be viewed in the GUI by selecting Hier Cell > Schematic View(Module) > in New.

## \*For Synthesis with constraints\*

| read\_sdc <path of .sdc constraints file> |
| --- |

| syn\_generic |
| --- |

Synthesizes the design to the G Tech cells (default cells for the Cadence Tool)

| syn\_map |
| --- |

maps the synthesized cells to the library specified earlier in read\_libs command

| syn\_opt -incremental |
| --- |

Incrementally optimizes the synthesized design

| report\_timing > (path for .rpt file to save timing report) |
| --- |

Reports timing Time Borrowed, Uncertainty, Required Time, Launch Clock, Data Path and Slack.

| report\_area > (path for .rpt file to save area report) |
| --- |

Reports area of the synthesized design in micro-meters-square

| report\_power > (path for .rpt file to save power report) |
| --- |

Reports power in nano Watts (nW)

| write\_hdl > (path for .v file for netlist to be written) |
| --- |

Writes the netlist in HDL format in the path specified

| quit |
| --- |

Exits the genus tool

# Syntax for writing the Constraints file (.sdc)

## Creating Clock “clk”

| set EXTCLK "clk" ;  set\_units -time <units eg.1.0ns> ;  set EXTCLK\_PERIOD <time period eg. 20.0>;  create\_clock -name "$EXTCLK" -period "$EXTCLK\_PERIOD" -waveform "0 [expr $EXTCLK\_PERIOD/2]" [get\_ports clk] |
| --- |

## Setting clock skew

| set SKEW <skew value eg. 0.200>  set\_clock\_uncertainty $SKEW [get\_clocks $EXTCLK] |
| --- |

## Setting source rise and fall latency

| set SRLATENCY <source rise latency eg. 0.80>  set SFLATENCY <source fall latency eg. 0.75> |
| --- |

## Setting Rise times and Fall Times

| set MINRISE <minimum rise time eg. 0.20>  set MAXRISE <maximum rise time eg. 0.25>  set MINFALL <minimum fall time eg. 0.15>  set MAXFALL <maximum fall time eg. 0.10>  set\_clock\_transition -rise -min $MINRISE [get\_clocks $EXTCLK]  set\_clock\_transition -rise -max $MAXRISE [get\_clocks $EXTCLK]  set\_clock\_transition -fall -min $MINFALL [get\_clocks $EXTCLK]  set\_clock\_transition -fall -max $MAXFALL [get\_clocks $EXTCLK] |
| --- |

## Setting Input and Output Delays for ports in the design

| set INPUT\_DELAY <input delay value eg. 0.5>  set OUTPUT\_DELAY <output delay value eg. 0.5>  set\_input\_delay -clock [get\_clocks $EXTCLK] -add\_delay 0.3 [get\_ports <input port name>]  set\_output\_delay -clock [get\_clocks $EXTCLK] -add\_delay 0.3 [get\_ports <output port name>] |
| --- |

***Note: the port names must correlate with the names used in the modules, and can be validated by using the report\_timing command while synthesizing with constraints in the Cadence Genus tool***

# Equivalence Checking (*conformal LEC*)

## Opening the Cadence Conformal Logical Equivalence Checking tool

| lec -LPGXL |
| --- |

Note: The GUI can be used alternative to entering the commands

## Reading Library

| GUI: file -> Read library -> choose the library’s .v file  Command Line: read library -Both -Replace -sensitive -Verilog technology\_file.v -nooptimize |
| --- |

## Reading Verilog file (GOLDEN)

| GUI: file -> Read design for RTL(Verilog code) -Golden -> choose the RTL verilog code  Command Line: read design design.v -Verilog -Golden -sensitive -continuousassignment Bidirectional -nokeep\_unreach -nosupply |
| --- |

## Reading Netlist file (REVISED)

| GUI: file -> Read design for RTL netlist or other code we want to compare with -Revised  Command Line: read design design2\_or\_netlist.v -Verilog -Revised -sensitive -continuousassignment Bidirectional -nokeep\_unreach -nosupply |
| --- |

## Set mode:

| set system mode lec |
| --- |

## Add points at which equivalence should be checked(Here we check all points):

| add compared points -all |
| --- |

## 

## Run the comparison:

| compare |
| --- |

The Equivalent and Non-Equivalent points are displayed in the GUI window, and can Non Equivalent modules/ sub-modules can be viewed by clicking on the ‘*Non-Equivalent*’ label

# Linting (*irun)*

## To run linting on the design file

| irun -superLint <path of the HDL file> |
| --- |

**The rtlchecks.log and modelchecks.log files are generated in your present working directory.**

The tool uses automated structural analysis on the design so that it complies with design coding rules that prevent synthesis issues & functional bugs and enforce coding styles for readability & re-use. These rtlchecks.log and modelchecks.log files present the \*N - notes, \*W - warnings and \*E - errors for the design.

# Static Timing Analysis (*Tempus)*

## Invoke the Tempus tool and open GUI

| tempus |
| --- |

## Alternatively: Invoke Tempus without GUI

| tempus -no\_gui |
| --- |

## Read library used for synthesis:

| read\_lib library\_file.lib |
| --- |

## Read synthesized gate level netlist:

| read\_verilog netlist.v |
| --- |

## Set the top module in your design:

| set\_top\_module top |
| --- |

## Read the constraints file:

| read\_sdc constraints.sdc |
| --- |

## Setting things to check: Signal integrity check

| set\_delay\_cal\_mode -siAware true  set\_si\_mode -enable\_delay\_report true  set\_si\_mode -enable\_glitch\_report true  set\_si\_mode -enable\_glitch\_propagation true |
| --- |

## Update the settings to report all timing parameters

| update\_timing -full |
| --- |

## Report timing: (general report with parameters like slack and critical path)

| report\_timing |
| --- |

## Report slack alone:

| report\_slack |
| --- |

| report\_timing -path\_type summary\_slack\_only |
| --- |

## Report clocks:

| report\_clocks |
| --- |

## Analysis report generation:

| report\_analysis\_coverage |
| --- |

# 

# 

# Additional References

**Changing power engine from joules to legacy in Genus Tool**

| set\_db power\_engine legacy |
| --- |

**Sending files from one system to another in cadence lab**

| scp <filename> <sender’s username>@<ipaddress of reciever>:<~ or receiving path >  Eg. scp /home/cmos/Desktop/DSD\_lab.zip @10.3.32.69:</home/cmos/Desktop> |
| --- |

**Incisive Formal Verifier(IFV):**

IFV tool can either read scripts(f file) or directly read from .sv file. Optionally a bind file can be created to link separate design(.sv) file and assertions file(.sva).

**Run basic:**

ifv example.sv

**Will display in the command prompt if the assertions failed or passed.**

ifv example.sv +gui

-> Click on the run button on the top right corner of the tool bar.

-> Right click on “Pass” or “Failed” to look for traces where it passed or failed.

**Will open simvision to display the traces.**

exit