DEPARTMENT OF ELECTRONICS AND COMMUNICATION

ATRIA INSTITUTE OF TECHNOLOGY

(Affiliated To Visvesvaraya Technological University, Belgaum)

Anandanagar, Bangalore-24

VLSI LAB MANUAL

7th SEMESTER ELECTRONICS AND COMMUNICATION

SUBJECT CODE: 15ECL77/17ECL77

2020-21

ASIC Design Flow Manual Using Cadence Tool Suite

Tools used for ASIC Flow:

1. **INCISIVE** - Used for Functional Simulation
2. **GENUS** - Used for Synthesis and pre-Layout Timing Analysis
3. **INNOVUS** - Used for Physical Design

Getting Started :

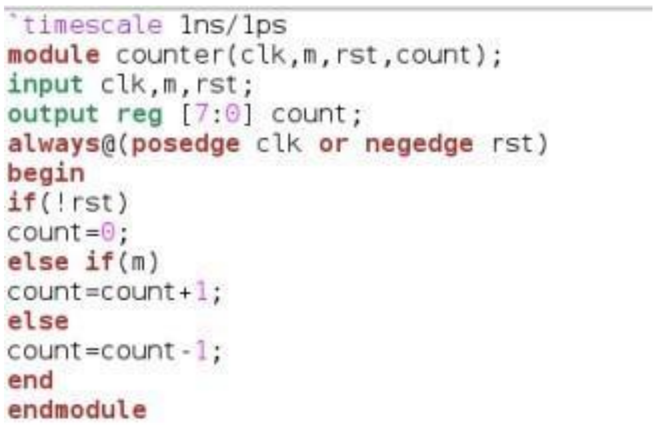
1. Make sure the Licensing Server is switched ON and the client is connected to server."
2. Open the "**counter**" directory and make a right click to "**Open in Terminal**"**.**
3. To open the tools to be used, type in the command "**csh**" (Press Enter) followed by "**source /home/install/cshrc**" <Or the path of tools whichever is applicable>.
4. A welcome string "**Welcome to Cadence Tool Suite**" appears indicating terminal ready to invoke Cadence Tools available for you.

Module 1: Creating an RTL Code

In order to create an RTL Code, you can open a text editor and type in your Verilog code or VHDL Code.

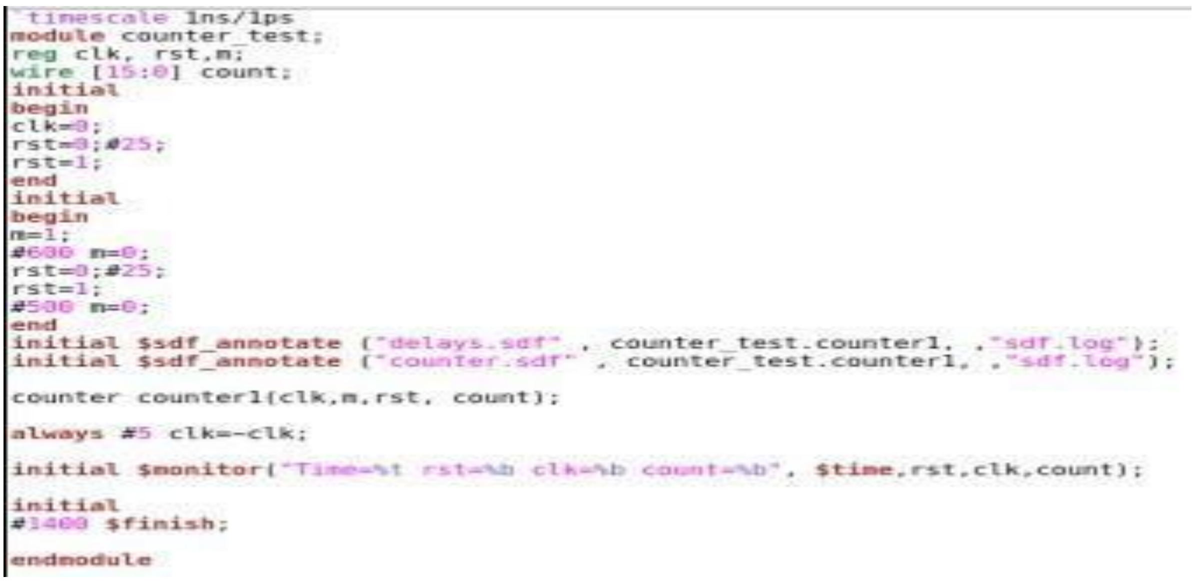
1. In the terminal, type in "**gedit <filename>.v [OR] <filename>.vhdl**"**.** The file extensions depends on the type of RTL Code you write as shown.

2. Similarly, using same command, Test Bench also could be written as shown below.



```
`timescale 1ns/1ps
module counter(clk,m,rst,count);
input clk,m,rst;
output reg [7:0] count;
always@(posedge clk or negedge rst)
begin
if(!rst)
count=0;
else if(m)
count=count+1;
else
count=count-1;
end
endmodule
```

RTL Code for a 16-bit Synchronous Up-Down Counter

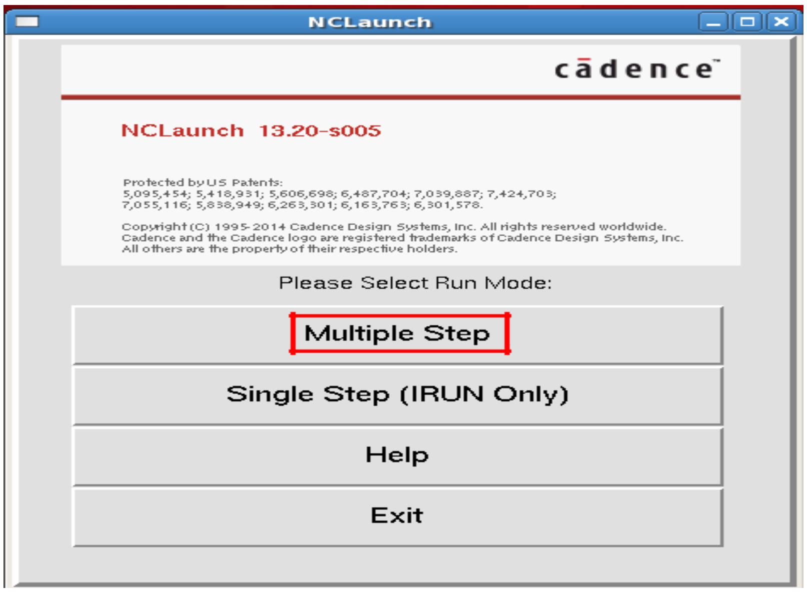


```
`timescale 1ns/1ps
module counter_test;
reg clk, rst,m;
wire [15:0] count;
initial
begin
clk=0;
rst=0;#25;
rst=1;
end
initial
begin
m=1;
#600 m=0;
rst=0;#25;
rst=1;
#500 m=0;
end
initial $sdf_annotate ("delays.sdf" , counter_test.counter1, ,"sdf.log");
initial $sdf_annotate ("counter.sdf" , counter_test.counter1, ,"sdf.log");

counter counter1(clk,m,rst, count);

always #5 clk=~clk;

initial $monitor("Time=%t rst=%b clk=%b count=%b", $time,rst,clk,count);

initial
#1400 $finish;

endmodule
```

Test Bench for the Up-Down Counter

Module 2: Functional Simulation

1. To perform Functional Simulation, “**Incisive**” tool is to be used.
2. In your terminal, type the command “**nclaunch -new**” to open the tool.

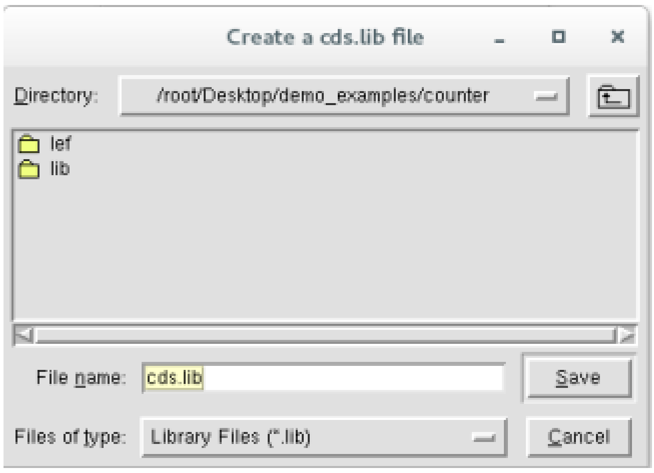


Protected by US Patents:
5,095,454; 5,418,931; 5,606,698; 6,487,704; 7,039,887; 7,424,703;
7,055,116; 5,838,949; 6,263,301; 6,163,763; 6,301,578.

Copyright (C) 1995-2014 Cadence Design Systems, Inc. All rights reserved worldwide.
Cadence and the Cadence logo are registered trademarks of Cadence Design Systems, Inc.
All others are the property of their respective holders.

3. Select “**Multiple Step**”. And then select “**Create cds.lib**”

Note : The ‘-new’ switch is used only for the first time the design is being run. For the next time on wards, the command to be used could be ‘nclaunch’ only.

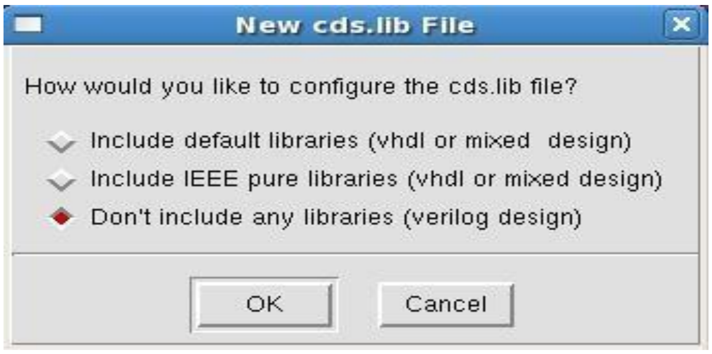


Directory: /root/Desktop/demo_examples/counter

File name: cds.lib

Files of type: Library Files (*.lib)

4. Save the cds.lib file. It is a tool file that holds the design location information for easy access by the tool.


New cds.lib File

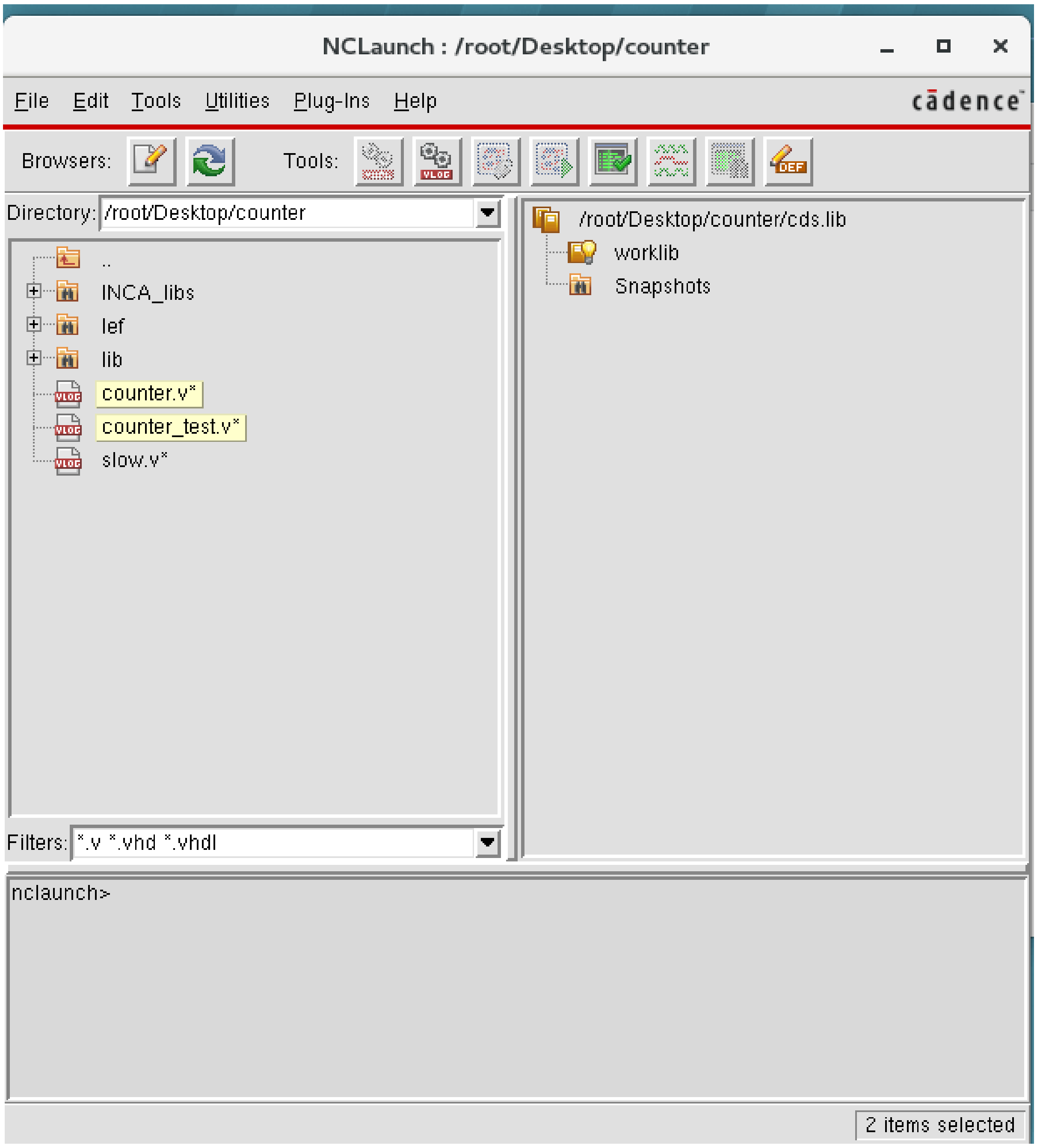
How would you like to configure the cds.lib file?

- Include default libraries (vhdl or mixed design)
- Include IEEE pure libraries (vhdl or mixed design)
- [✓ Don't include any libraries (verilog design)]

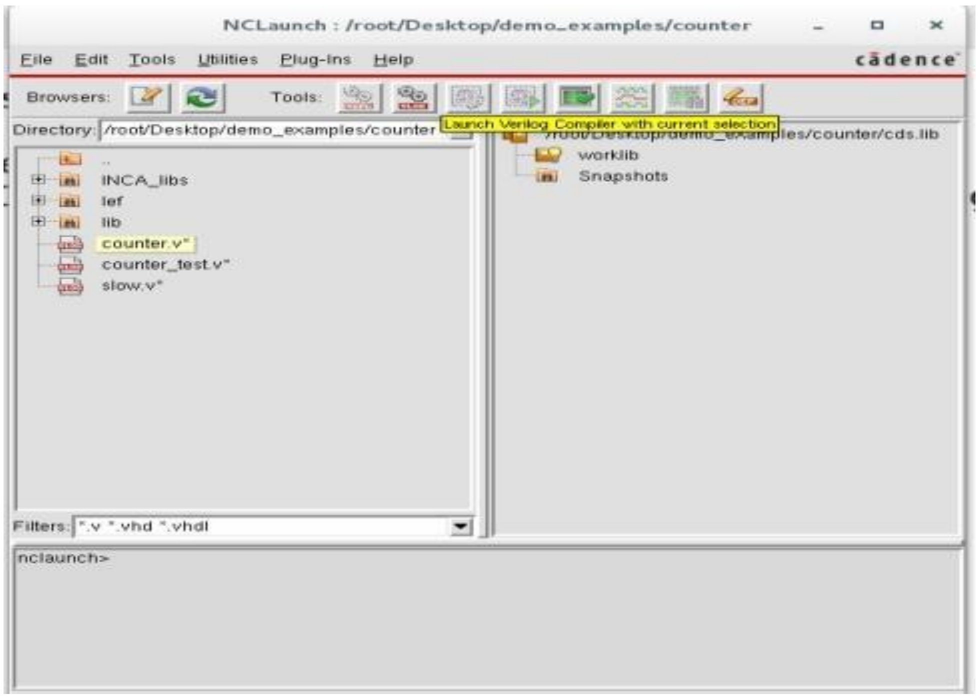
5. Based on the Libraries available and the type of RTL Code written, one of the three shown above is to be selected. Cadence tool suite provides default gpdk libraries. Here, counter RTL is of Verilog Format and hence third option is selected.
6. A new pop-up "nclaunch" opens which will contain all the .v and .vhdl files as per the cds.lib file created.
7. **Functional Simulation using Cadence runs in 3 stages:**
   → Compilation of Verilog/VHDL Code and/or Test Bench
   → Elaboration of the Code & Test Bench Compiled
   → Simulating the Test Bench or Top Module[in absence of Test Bench]

8. A set of tools are shown in the nclaunch window which refer to **VHDL Compiler, Verilog Compiler, Elaborator, Simulator** corresponding from Left To Right.
9. Select the .v or .vhdl files to be compiled and launch Compiler. On successful completion of compilation, on the Right hand Side, the modules appear under "**Worklib**" .
10. Select the Module under Worklib and "**Launch Elaborator**" . On successful completion of Elaboration, "**Snapshots**" are generated.
11. Select the Test Bench under snapshots and "**Launch Simulator**" .

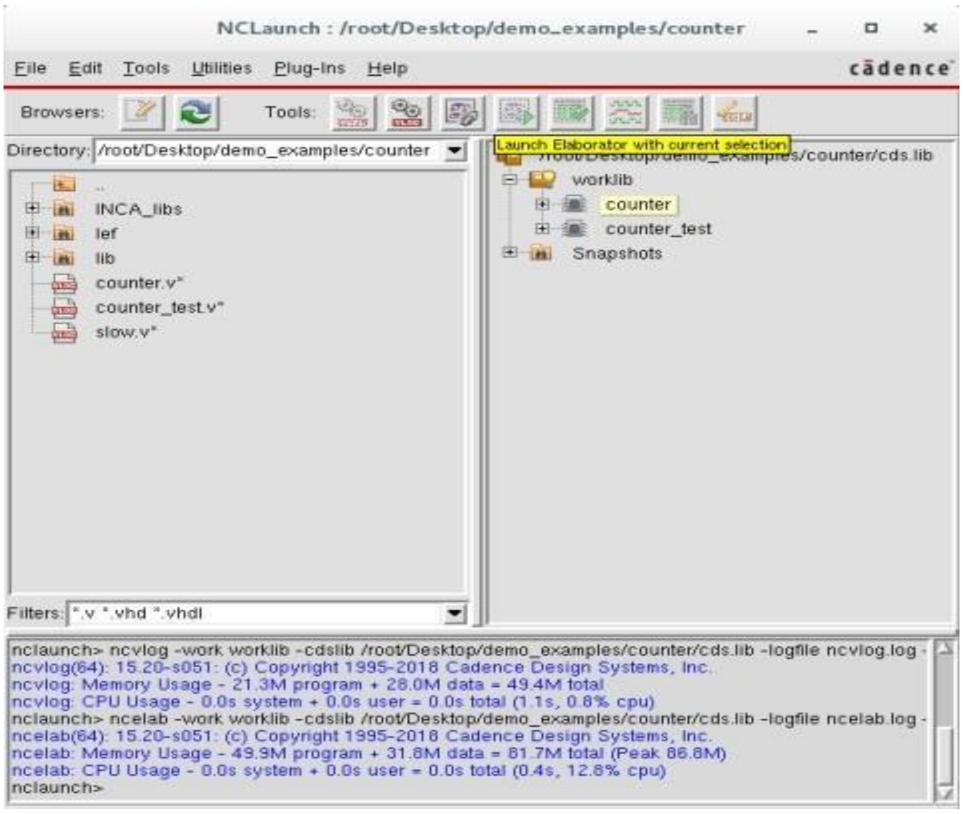
12.The above steps are depicted under following snapshosts.


NCLaunch : /root/Desktop/counter
File Edit Tools Utilities Plug-Ins Help
cādence
Browsers:
Tools:
Directory: /root/Desktop/counter
..
INCA_libs
lef
lib
counter.v*
counter_test.v*
slow.v*
Filters: *.v *.vhd *.vhdl
/root/Desktop/counter/cds.lib
worklib
Snapshots
2 items selected

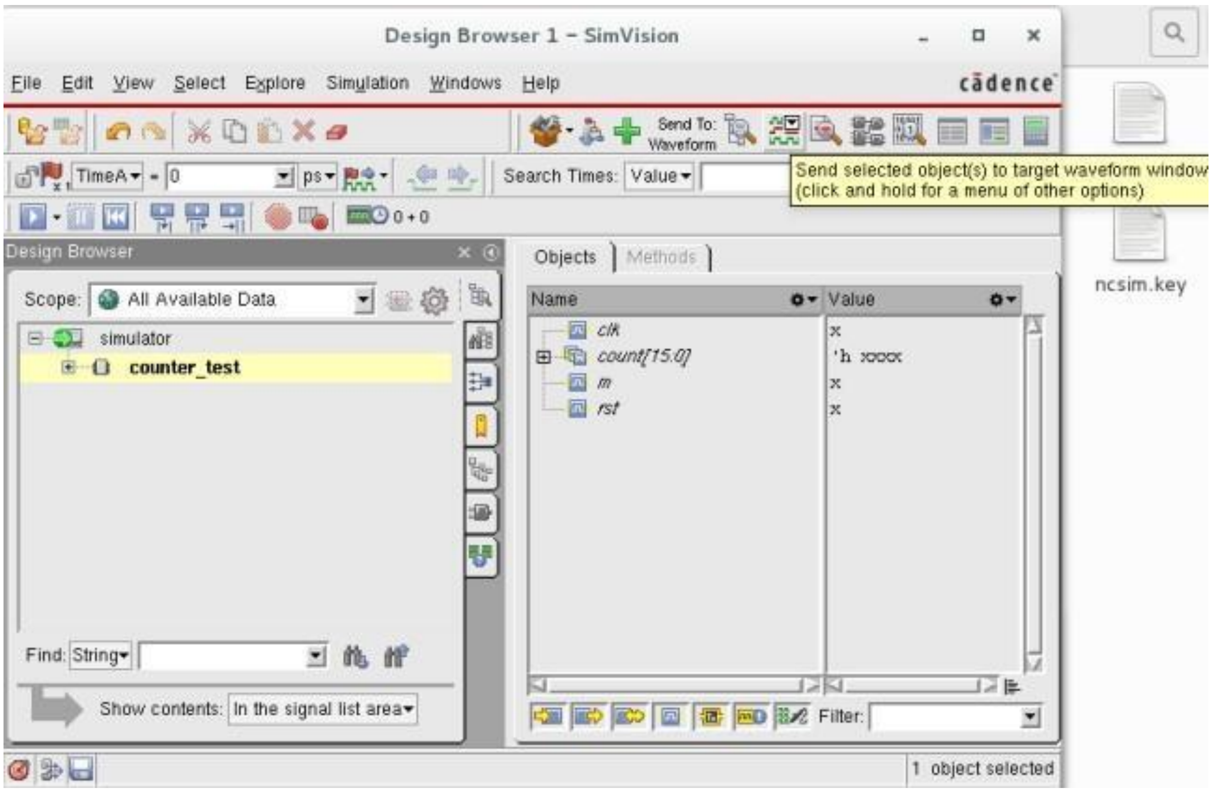
```
nclaunch>
```


NCLaunch : /root/Desktop/demo_examples/counter
File Edit Tools Utilities Plug-Ins Help
cādence
Browsers:
Tools:
Launch Verilog Compiler with current selection
Directory: /root/Desktop/demo_examples/counter
/root/Desktop/demo_examples/counter/cds.lib
worklib
Snapshots
..
INCA_libs
lef
lib
counter.v*
counter_test.v*
slow.v*
Filters: *.v *.vhd *.vhdl

```
nclaunch>
```


NCLaunch : /root/Desktop/demo_examples/counter
File Edit Tools Utilities Plug-Ins Help
cādence
Browsers:
Tools:
Launch Elaborator with current selection
Directory: /root/Desktop/demo_examples/counter
/root/Desktop/demo_examples/counter/cds.lib
worklib
counter
counter_test
Snapshots
..
INCA_libs
lef
lib
counter.v*
counter_test.v*
slow.v*
Filters: *.v *.vhd *.vhdl

```
nclaunch> ncvlog -work worklib -cdslib /root/Desktop/demo_examples/counter/cds.lib -logfile ncvlog.log -
ncvlog(64): 15.20-s051: (c) Copyright 1995-2018 Cadence Design Systems, Inc.
ncvlog: Memory Usage - 21.3M program + 28.0M data = 49.4M total
ncvlog: CPU Usage - 0.0s system + 0.0s user = 0.0s total (1.1s, 0.8% cpu)
nclaunch> ncelab -work worklib -cdslib /root/Desktop/demo_examples/counter/cds.lib -logfile ncelab.log -
ncelab(64): 15.20-s051: (c) Copyright 1995-2018 Cadence Design Systems, Inc.
ncelab: Memory Usage - 49.9M program + 31.8M data = 81.7M total (Peak 86.8M)
ncelab: CPU Usage - 0.0s system + 0.0s user = 0.0s total (0.4s, 12.8% cpu)
nclaunch>
```



Send selected object(s) to target waveform window
(click and hold for a menu of other options)

Scope: All Available Data

simulator
counter_test

Find: String

Show contents: In the signal list area

Name	Value
clk	x
count[15:0]	'h xxxx
m	x
rst	x

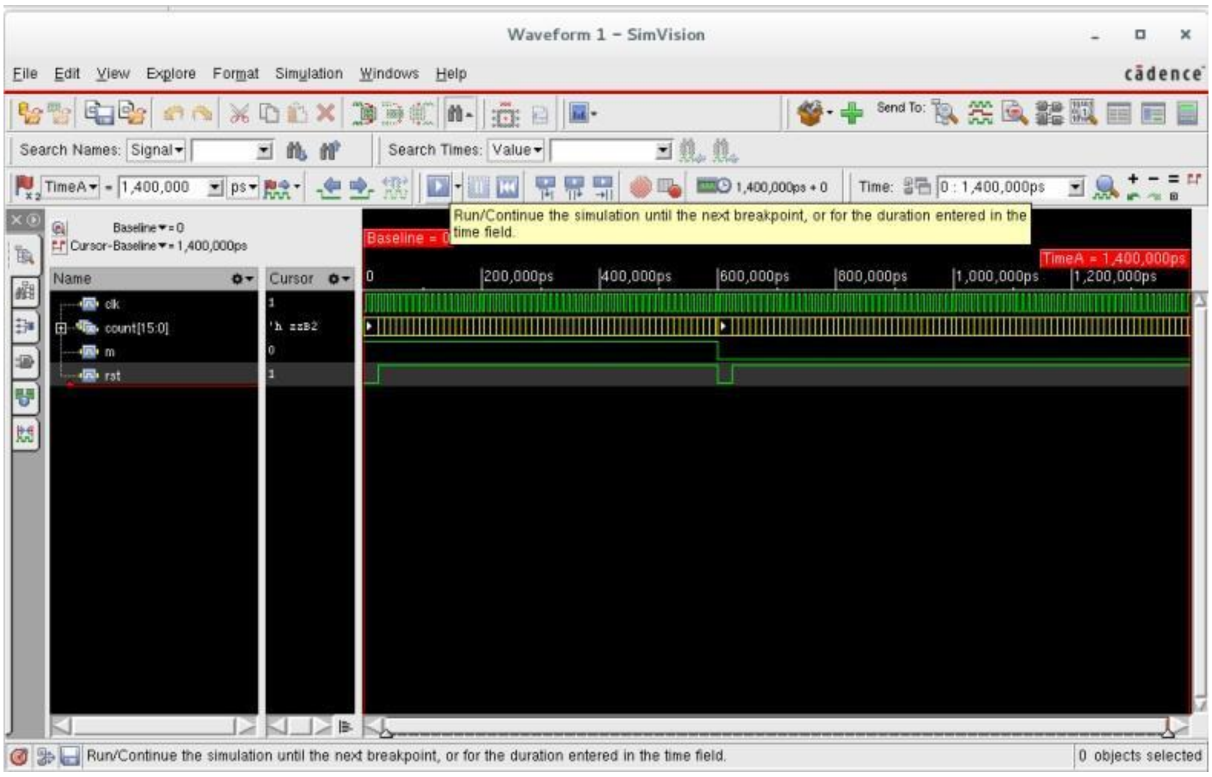
Filter:

1 object selected

The Design Browser pops-up and The Test Bench Module name can be seen on the left and the Pin list on the right when selected. For Simulation, The number of Pins / Ports to be simulated can be selected.

Make a right click on the selected and Select "Send to Waveform Window".

In the waveform window, we can see different ports in the design. Now click on the Run simulation key to start the simulation. Use the 'pause' key to interrupt or stop the simulation. Use different options like zoom in, zoom out etc to analyze the plot.


Waveform 1 - SimVision
File Edit View Explore Format Simulation Windows Help
cādence
Search Names: Signal
Search Times: Value
TimeA = 1,400,000 ps
1,400,000ps + 0
Time: 0 : 1,400,000ps
Baseline = 0
Cursor-Baseline = 1,400,000ps
Run/Continue the simulation until the next breakpoint, or for the duration entered in the time field.
Baseline = 0
TimeA = 1,400,000ps
Name
Cursor
clk
count[15:0]
m
rst
1
'h zzB2
0
1
0
200,000ps
400,000ps
600,000ps
800,000ps
1,000,000ps
1,200,000ps
Run/Continue the simulation until the next breakpoint, or for the duration entered in the time field.
0 objects selected

Module 3: Synthesis

Inputs for Synthesis :

1. **RTL Code (.v or .vhdl)**
2. **Chip Level SDC (System Design Constraints)**
3. **Liberty Files (.lib)**

Expected Outputs of Synthesis :

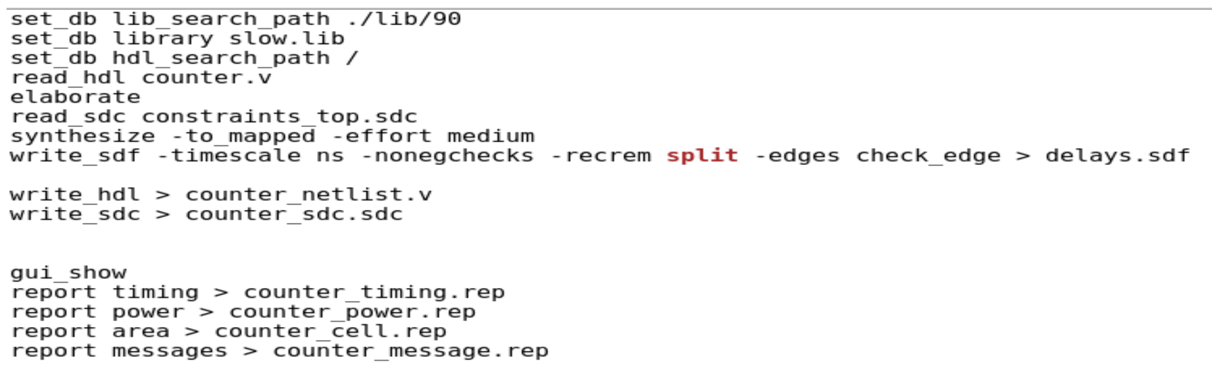
1. **Gate Level Netlist**
2. **Block Level Netlist**
3. **Timing, Area, Power reports**

Synthesis is a 3-stage process which converts Virtual RTL Logic into Physical Gates in order to give a Physical Shape to the design through Physical Design.

Synthesis runs in following stages :

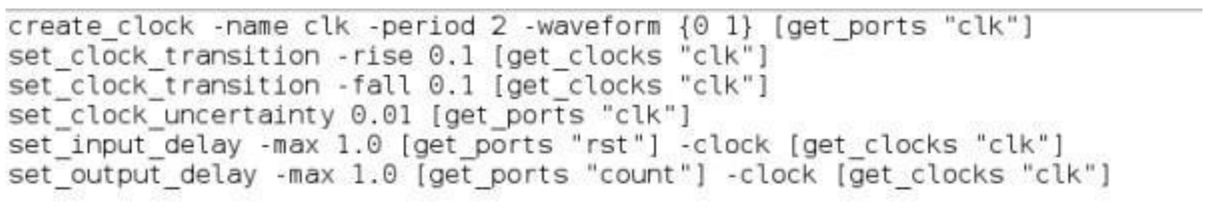
→ Translation - RTL Codes are compiled
→ Elaboration / Mapping - Pieces of Logic are replaced with corresponding Gates from Libraries with same Functionality
→ Optimization - Tool tries to reduce cell count without affecting the functionality

To run the synthesis, the following script can be used.



```
set_db lib_search_path ./lib/90
set_db library slow.lib
set_db hdl_search_path /
read_hdl counter.v
elaborate
read_sdc constraints_top.sdc
synthesize -to_mapped -effort medium
write_sdf -timescale ns -nonegchecks -recrem split -edges check_edge > delays.sdf

write_hdl > counter_netlist.v
write_sdc > counter_sdc.sdc


gui_show
report timing > counter_timing.rep
report power > counter_power.rep
report area > counter_cell.rep
report messages > counter_message.rep
```

Chip Level SDC is as follows :



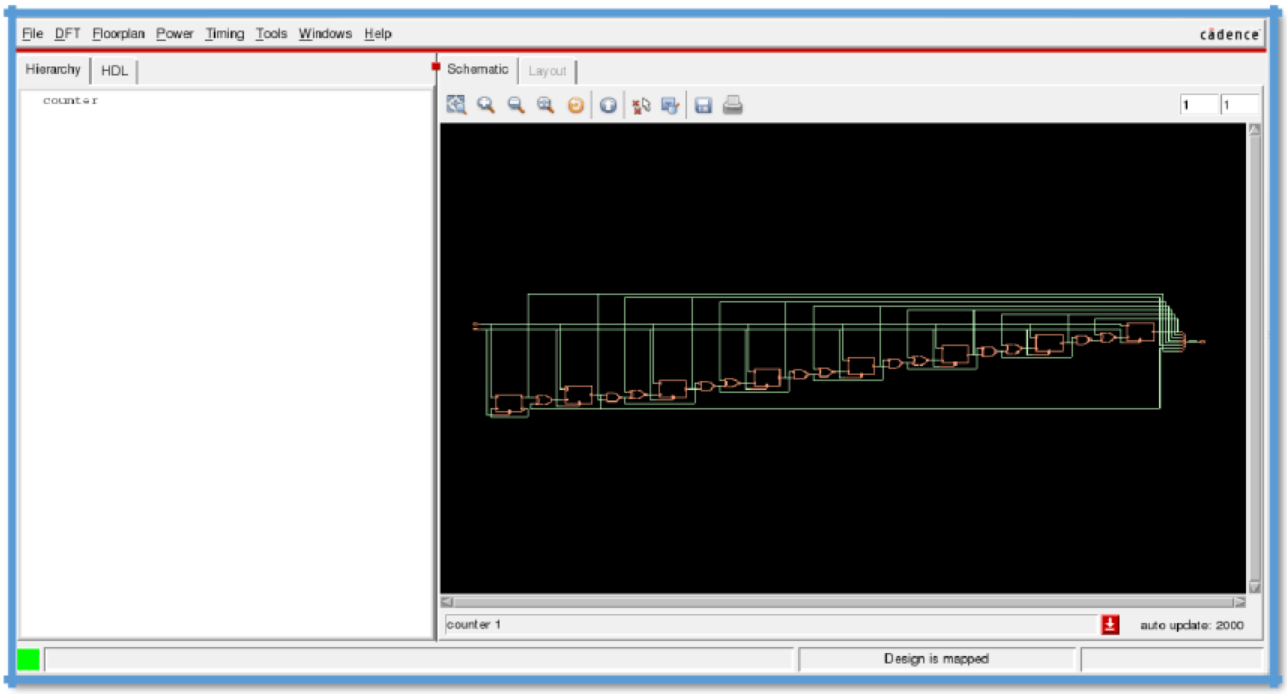
```
create_clock -name clk -period 2 -waveform {0 1} [get_ports "clk"]
set_clock_transition -rise 0.1 [get_clocks "clk"]
set_clock_transition -fall 0.1 [get_clocks "clk"]
set_clock_uncertainty 0.01 [get_ports "clk"]
set_input_delay -max 1.0 [get_ports "rst"] -clock [get_clocks "clk"]
set_output_delay -max 1.0 [get_ports "count"] -clock [get_clocks "clk"]
```

Close out all INCISIVE windows and in the same terminal type in the following command to run Synthesis.

genus -f rc_script.tcl

The tcl [Tool Command Language] script runs executing each command one after the other.

A window of Genus GUI pops – up with the top hier cell on the left top. Make a right click and select Schematic Viewer → In Main.



The Gate level Circuit that implements the RTL Logic can be seen and analysed.

As from the script, Block Level SDC, Gate Level Netlist, Timing, Power and Area reports are generated which are readable.

The Timing report gives the path with Worst timing.

The area report gives Cell count and Total area occupied by them.

The total power consumed by those cells are given in Power report.

Module 4 : Physical Design

Mandatory Inputs for PD :

1. **Gate Level Netlist [Output of Synthesis]**
2. **Block Level SDC [Output of Synthesis]**
3. **Liberty Files (.lib)**
4. **LEF Files (Layer Exchange Format)**

Expected Outputs from PD :

1. **GDS II File (Graphical Data Stream for Information Interchange – Feed In for Fabrication Unit).**

Close out all windows relating to Genus and in the terminal, type the command

innovus (Press Enter)

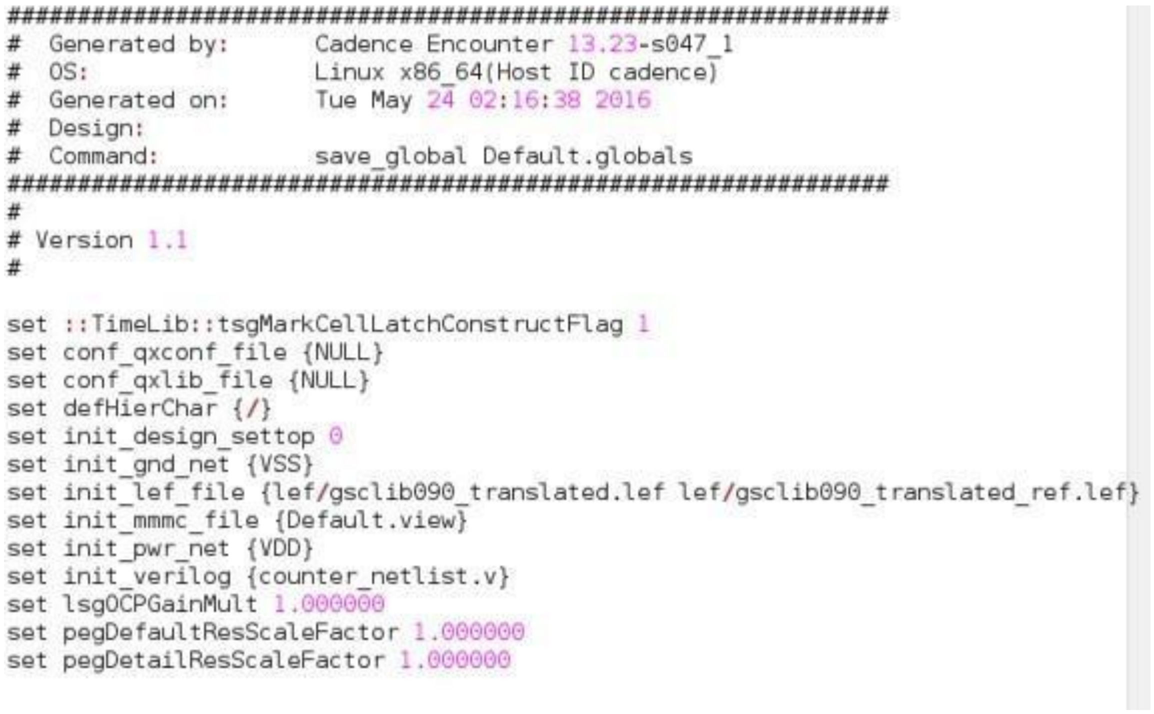
For Innovus tool, a GUI opens and also the terminal enters into innovus command prompt where in the tool commands can be entered.

Physical Design involves 5 stages as following :

After Importing Design,
→ Floor Planning
→ Power Planning
→ Placement
→ CTS (Clock Tree Synthesis)
→ Routing

Module 4.1 : Importing Design

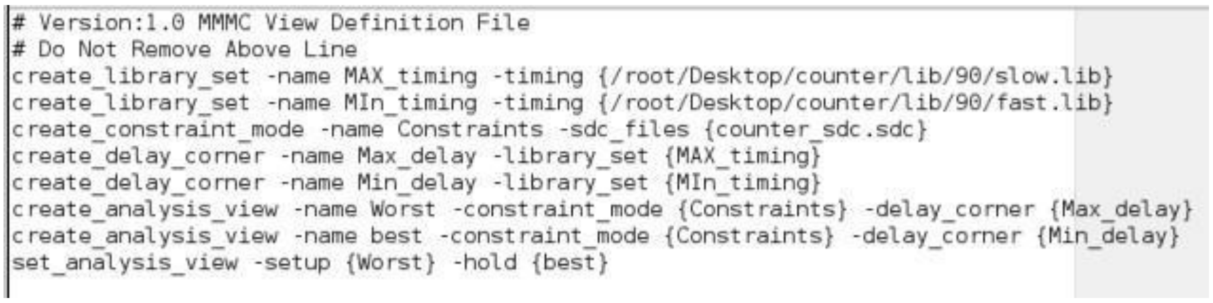
To Import Design, all the Mandatory Inputs are to be loaded and this can be done using script files named with .globals and .view/.tcl



```
###################################################################
#  Generated by:       Cadence Encounter 13.23-s047_1
#  OS:                 Linux x86_64(Host ID cadence)
#  Generated on:       Tue May 24 02:16:38 2016
#  Design:
#  Command:            save_global Default.globals
###################################################################
#
# Version 1.1
#

set ::TimeLib::tsgMarkCellLatchConstructFlag 1
set conf_qxconf_file {NULL}
set conf_qxlib_file {NULL}
set defHierChar {/}
set init_design_settop 0
set init_gnd_net {VSS}
set init_lef_file {lef/gsclib090_translated.lef lef/gsclib090_translated_ref.lef}
set init_mmmc_file {Default.view}
set init_pwr_net {VDD}
set init_verilog {counter_netlist.v}
set lsgOCPGainMult 1.000000
set pegDefaultResScaleFactor 1.000000
set pegDetailResScaleFactor 1.000000
```

Globals File to import design using Mandatory Inputs

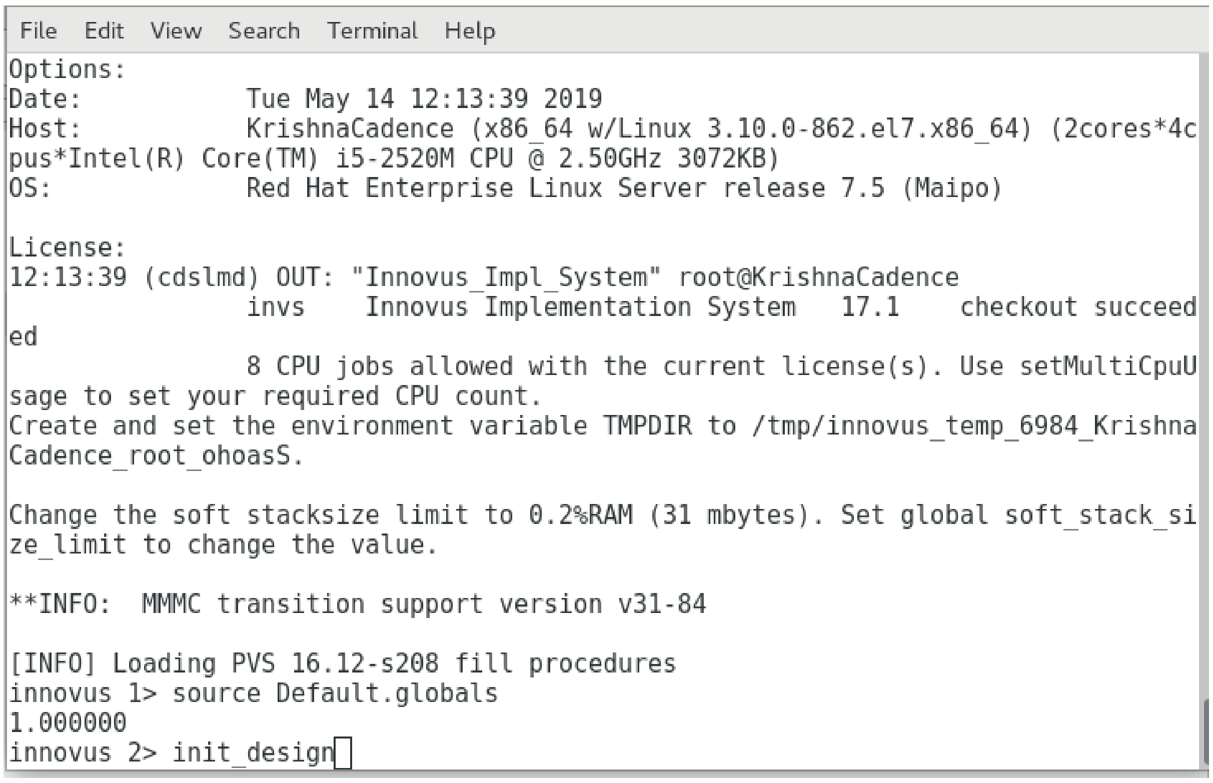
The Globals file reads in the LEF's and Gate Level Netlist and .view file implicitly.



```
# Version:1.0 MMMC View Definition File
# Do Not Remove Above Line
create_library_set -name MAX_timing -timing {/root/Desktop/counter/lib/90/slow.lib}
create_library_set -name MIn_timing -timing {/root/Desktop/counter/lib/90/fast.lib}
create_constraint_mode -name Constraints -sdc_files {counter_sdc.sdc}
create_delay_corner -name Max_delay -library_set {MAX_timing}
create_delay_corner -name Min_delay -library_set {MIn_timing}
create_analysis_view -name Worst -constraint_mode {Constraints} -delay_corner {Max_delay}
create_analysis_view -name best -constraint_mode {Constraints} -delay_corner {Min_delay}
set_analysis_view -setup {Worst} -hold {best}
```

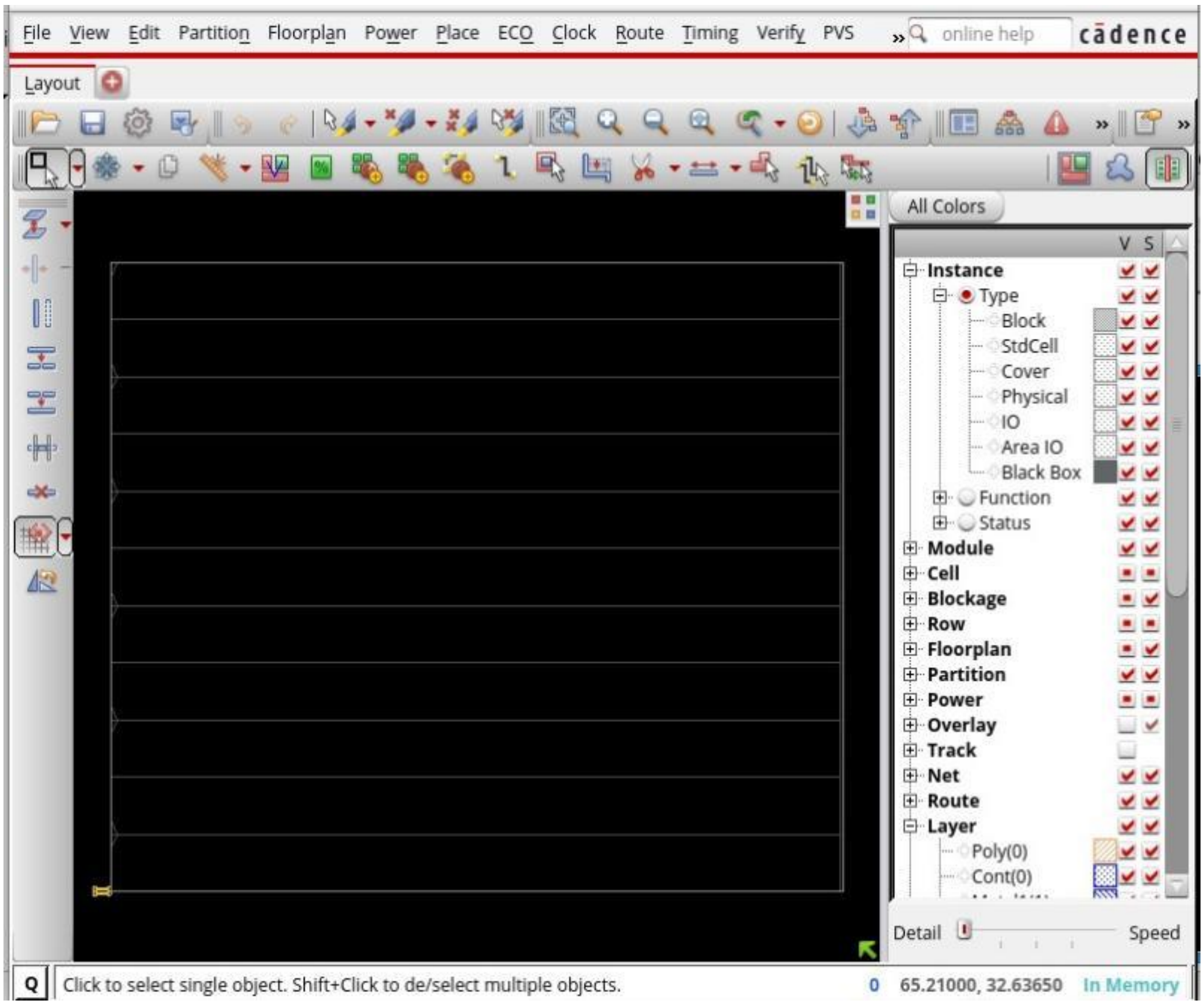
The .view file reads Liberty Files and Block Level SDC to create various PVT Corners for analysis.

In the terminal command prompt, type the commands as shown. The design is imported and "Core Area" is calculated by tool and shown on GUI.



```
Options:
Date:           Tue May 14 12:13:39 2019
Host:           KrishnaCadence (x86_64 w/Linux 3.10.0-862.el7.x86_64) (2cores*4c
pus*Intel(R) Core(TM) i5-2520M CPU @ 2.50GHz 3072KB)
OS:             Red Hat Enterprise Linux Server release 7.5 (Maipo)

License:
12:13:39 (cdslmd) OUT: "Innovus_Impl_System" root@KrishnaCadence
                invs    Innovus Implementation System   17.1    checkout succeed
ed
                8 CPU jobs allowed with the current license(s). Use setMultiCpuU
sage to set your required CPU count.
Create and set the environment variable TMPDIR to /tmp/innovus_temp_6984_Krishna
Cadence_root_ohoasS.

Change the soft stacksize limit to 0.2%RAM (31 mbytes). Set global soft_stack_si
ze_limit to change the value.

**INFO:  MMMC transition support version v31-84

[INFO] Loading PVS 16.12-s208 fill procedures
innovus 1> source Default.globals
1.000000
innovus 2> init_design
```

The Horizontal Lines on the GUI across the Core Area are alternative VDD and VSS tracks and Standard Cell Placement Rows.



Q Click to select single object. Shift+Click to de/select multiple objects. 0 65.21000, 32.63650 In Memory

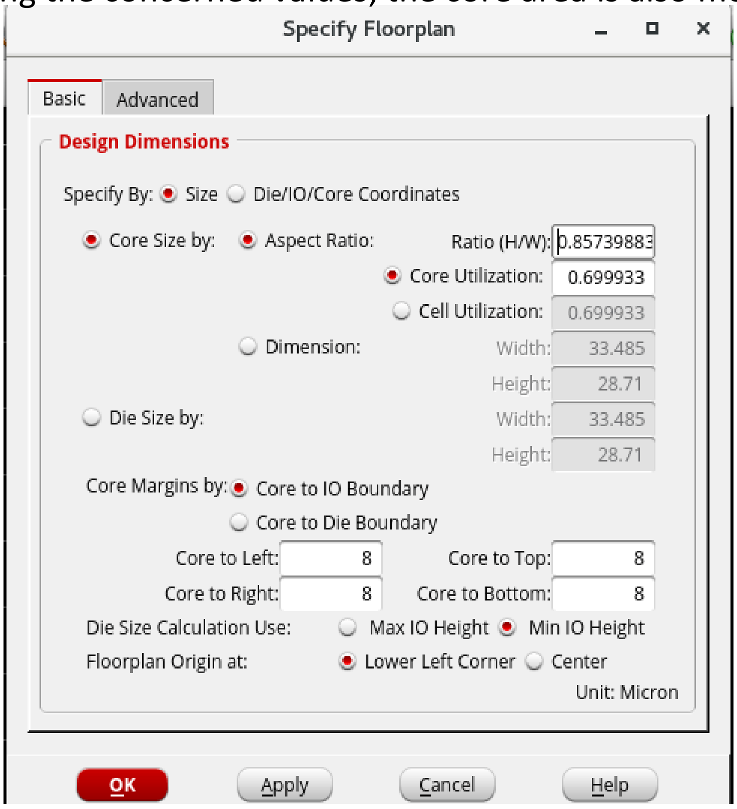
Module 4.2 : Floorplan

Steps under Floorplan :

1. **Aspect Ratio** [Ratio of Vertical Height to Horizontal Width of Core]
2. **Core Utilisation** [The total Core Area % to be used for Floor Planning]
3. **Channel Spacing between Core Boundary to IO Boundary**

Select **Floorplan → Specify Floorplan** to modify/add concerned values to the above Factors.

On adding/modifying the concerned values, the core area is also modified.



Specify Floorplan

Basic | Advanced

Design Dimensions

Specify By: [✓ Size] Die/IO/Core Coordinates

[✓ Core Size by:] [✓ Aspect Ratio:] Ratio (H/W): 0.85739883

[✓ Core Utilization:] 0.699933

Cell Utilization: 0.699933

Dimension: Width: 33.485

Height: 28.71

Die Size by: Width: 33.485

Height: 28.71

Core Margins by: [✓ Core to IO Boundary]

Core to Die Boundary

Core to Left: 8 Core to Top: 8

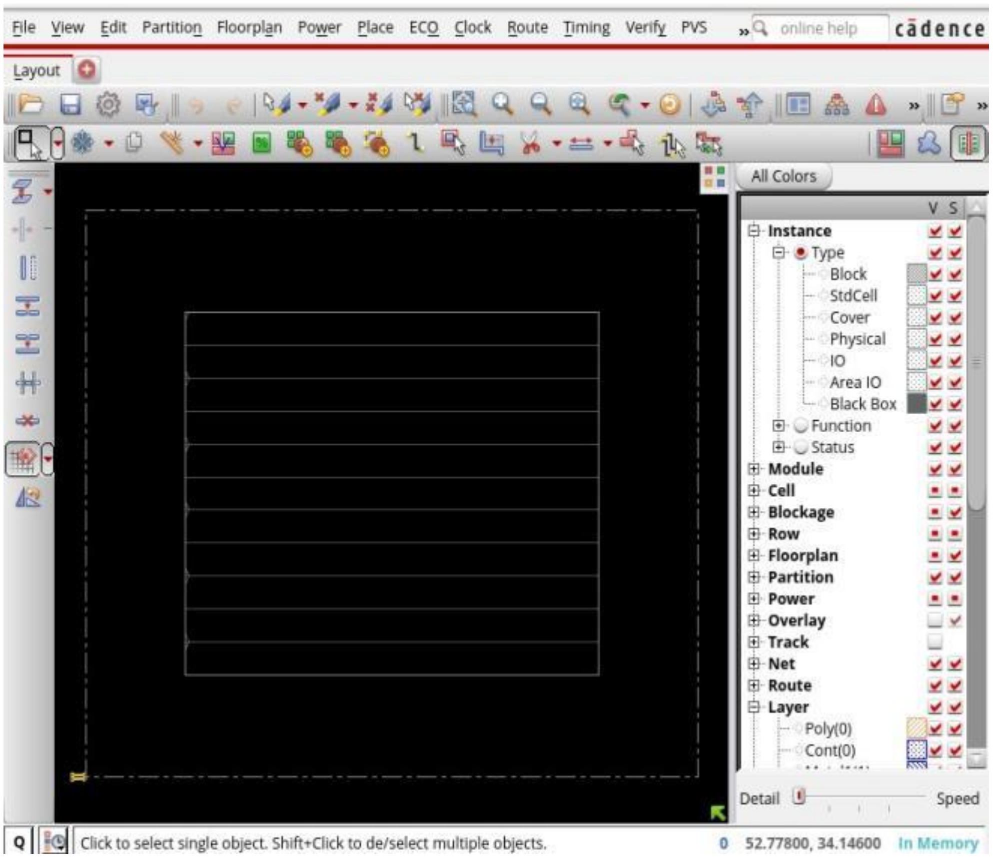
Core to Right: 8 Core to Bottom: 8

Die Size Calculation Use: Max IO Height [✓ Min IO Height]

Floorplan Origin at: [✓ Lower Left Corner] Center

Unit: Micron

OK Apply Cancel Help



Click to select single object. Shift+Click to de/select multiple objects. 0 52.77800, 34.14600 In Memory

The Yellow patch on the Left Bottom are the group of "Unassigned pins" which are to be placed along the IO Boundary along with the Standard Cells [Gates].

Module 4.3 : Power Planning

Steps under Power Planning :

1. **Connect Global Net Connects**
2. **Adding Power Rings**
3. **Adding Power Strings**
4. **Special Route**

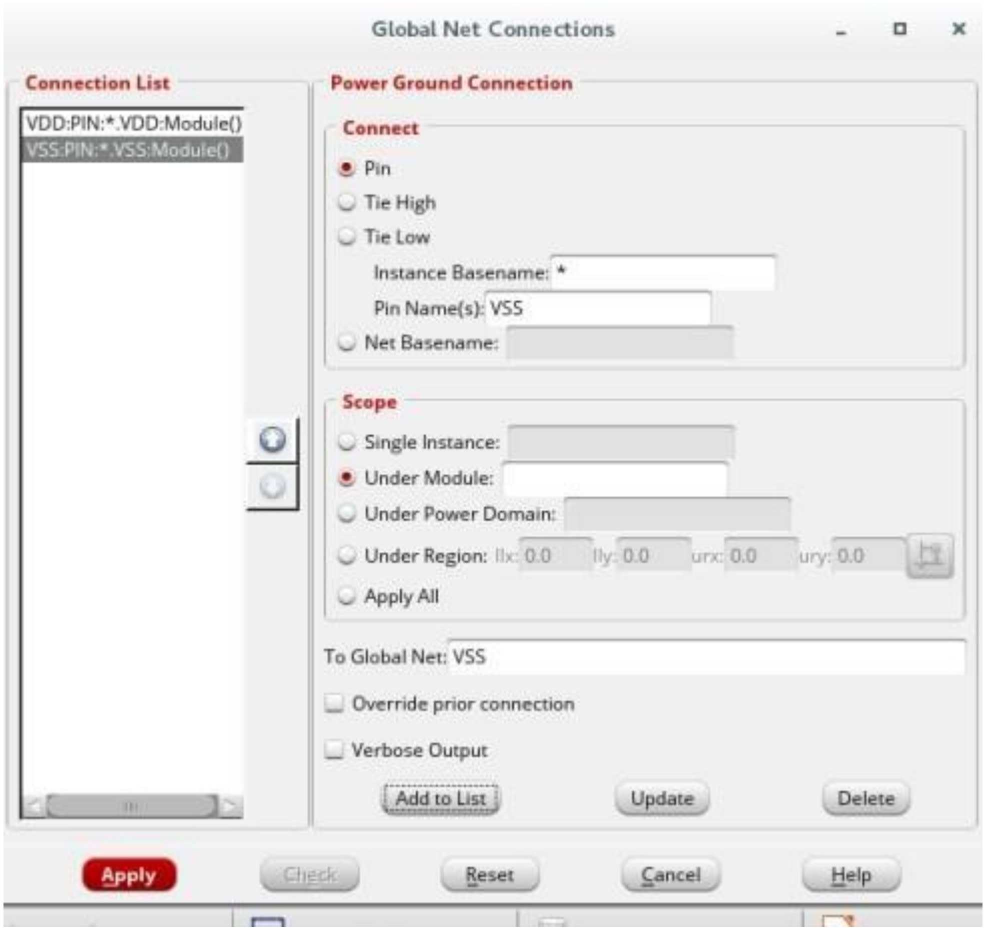
During the stage of Importing Design, under the Globals file, Two command lines state the names of Power and Ground Nets.

However, in order to Current flow through these Power nets, they are to converge at a point, preferably a common net connected to a Pin.

Under **Connect Global Net Connects**, we create two pins, one for VDD and one for VSS connecting them to corresponding Global Nets as mentioned in Globals file.

Select **Power → Connect Global Nets..** to create "Pin" and "Connect to Global Net" as shown and use "Add to list".

Click on "Apply" to direct the tool in enforcing the Pins and Net connects to Design and then Close the window.


Global Net Connections

Connection List
VDD:PIN:*.VDD:Module()
VSS:PIN:*.VSS:Module()

Power Ground Connection

Connect
● Pin
○ Tie High
○ Tie Low
Instance Basename: *
Pin Name(s): VSS
○ Net Basename:

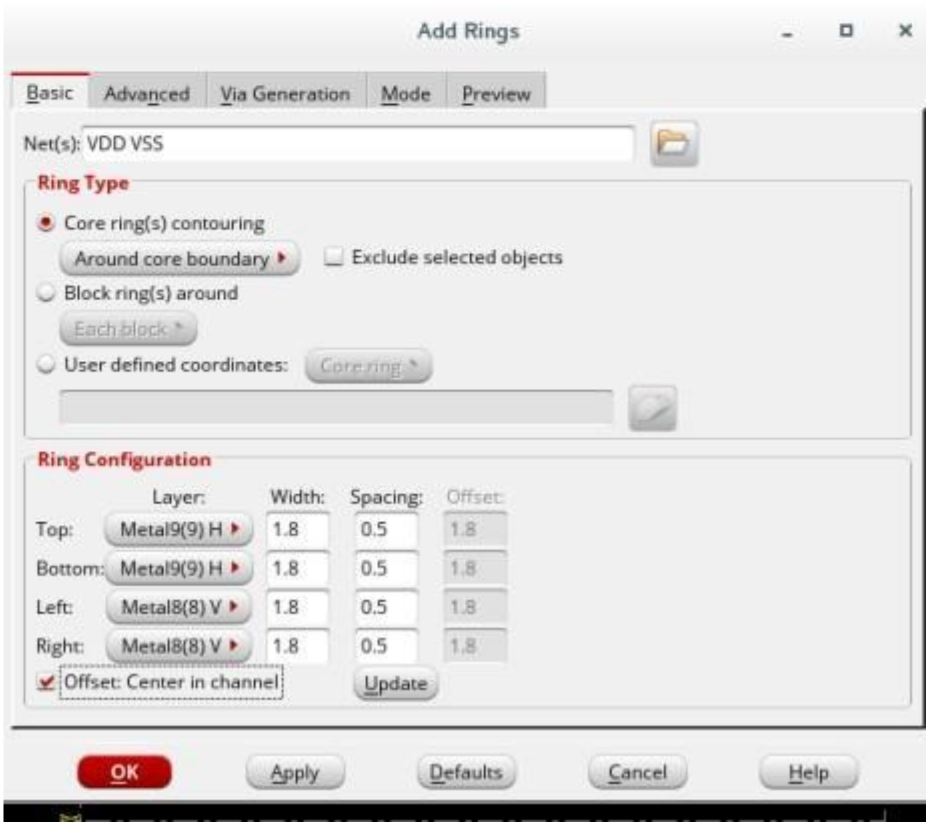
Scope
○ Single Instance:
● Under Module:
○ Under Power Domain:
○ Under Region: llx: 0.0 lly: 0.0 urx: 0.0 ury: 0.0
○ Apply All

To Global Net: VSS
☐ Override prior connection
☐ Verbose Output

In order to Tap in Power from a distant Power supply, Wider Nets and Parallel connections improve efficiency. Moreover, the cells that would be placed inside the core area are expected to have shorter Nets for lower resistance.

Hence Power Rings [Around Core Boundary] and Power Stripes [Across Core Boundary] are added which satisfies the above conditions.

Select **Power → Power Planning → Add Rings** to add Power rings ‘around Core Boundary’.


Add Rings

Basic | Advanced | Via Generation | Mode | Preview

Net(s): VDD VSS

Ring Type

Core ring(s) contouring [✓]
Around core boundary
Exclude selected objects [ ]
Block ring(s) around [ ]
Each block
User defined coordinates: [ ] Core ring

Ring Configuration

	Layer:	Width:	Spacing:	Offset:
Top:	Metal9(9) H	1.8	0.5	1.8
Bottom:	Metal9(9) H	1.8	0.5	1.8
Left:	Metal8(8) V	1.8	0.5	1.8
Right:	Metal8(8) V	1.8	0.5	1.8

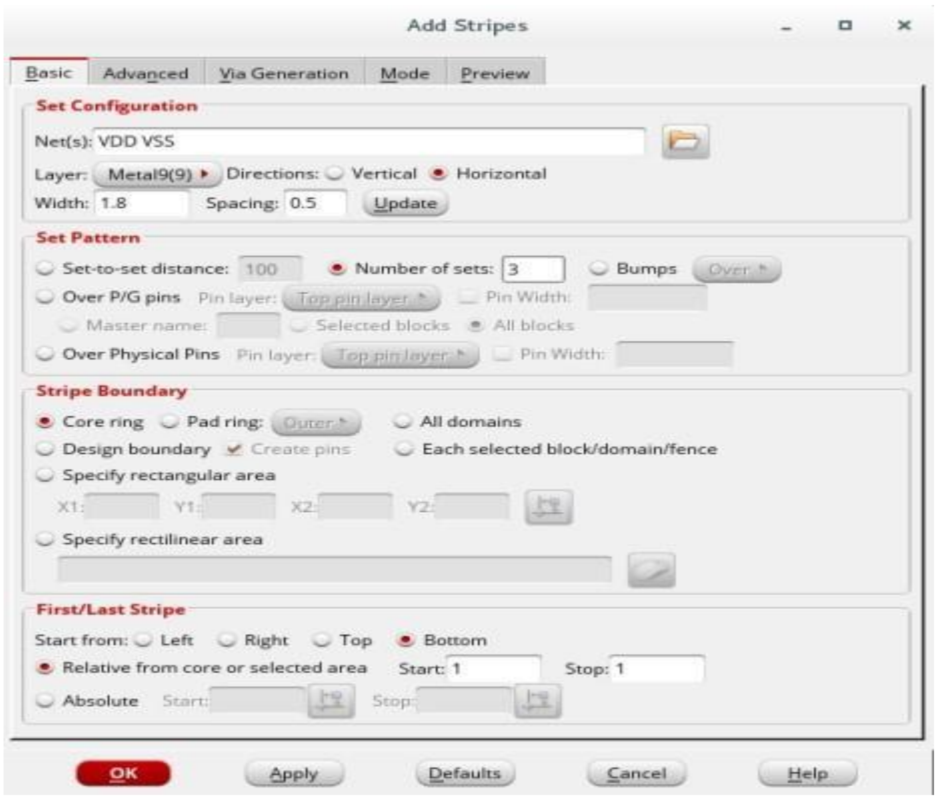
Offset: Center in channel [✓]

→ Select the Nets from Browse option OR Directly type in the Global Net Names separated by a space being Case and Spelling Senstive.

→ Select the Highest Metals marked ‘H’ [Horizontal] for Top and Bottom and Metals marked ‘V’ [Vertical] for Right and Bottom. This is because Highest metals have Highest Widths and thus Lowest Resistance.

→ Click on Update after the selection and “Set Offset : Center in Channel” in order to get the Minimum Width and Minimum Spacing of the corresponding Metals and then Click “OK”.

→ Similarly, Power Stripes are added using similar content to that of Power Rings.



Set Configuration

Net(s): VDD VSS

Layer: Metal9(9) Directions: Vertical [✓ Horizontal]

Width: 1.8 Spacing: 0.5 Update

Set Pattern

Set-to-set distance: 100 [✓ Number of sets: 3] Bumps Over

Over P/G pins Pin layer: Top pin layer Pin Width:

Master name: Selected blocks [✓ All blocks]

Over Physical Pins Pin layer: Top pin layer Pin Width:

Stripe Boundary

[✓ Core ring] Pad ring: Outer All domains

Design boundary [✓ Create pins] Each selected block/domain/fence

Specify rectangular area

X1: Y1: X2: Y2:

Specify rectilinear area

First/Last Stripe

Start from: Left Right Top [✓ Bottom]

[✓ Relative from core or selected area] Start: 1 Stop: 1

Absolute Start: Stop:

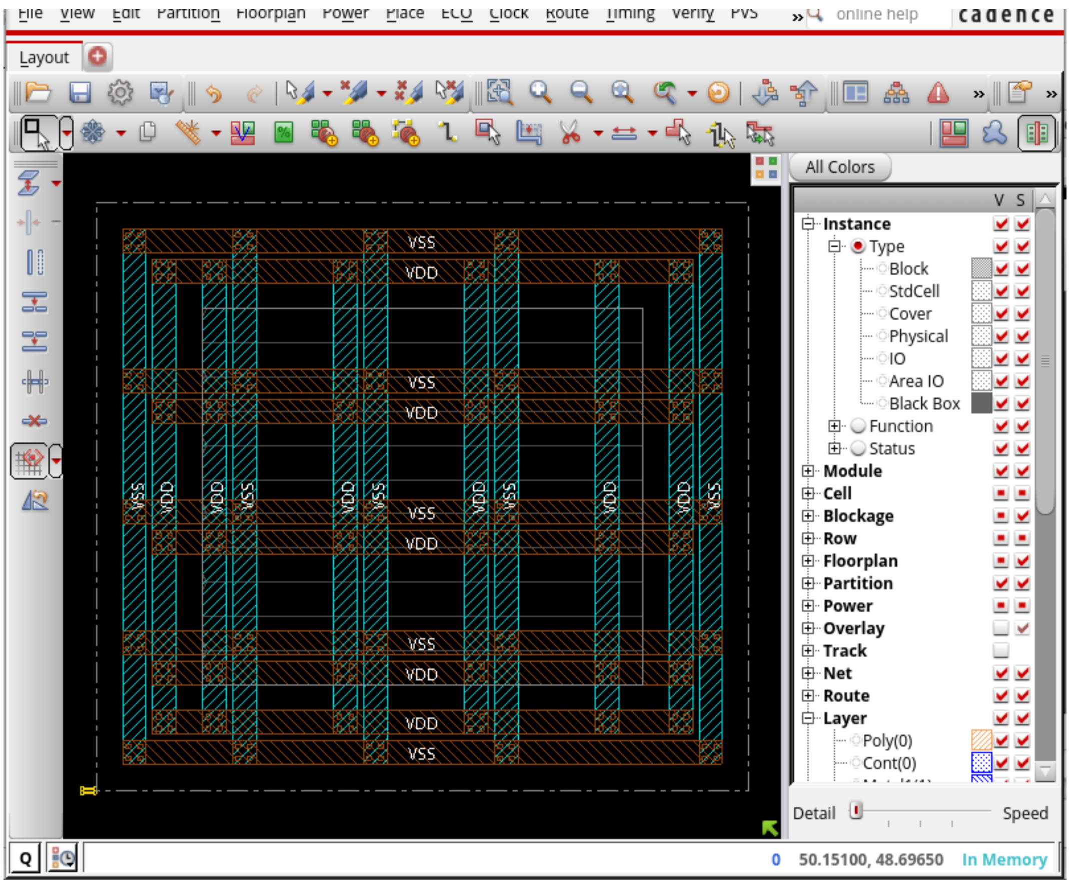
Factors to be considered under Power Stripes :

→ Nets

→ Metal and It's Direction

→ Width and Spacing [Updated]

→ Set to Set Distance = (Minimum Width of Metal + Min. Spacing) x 2

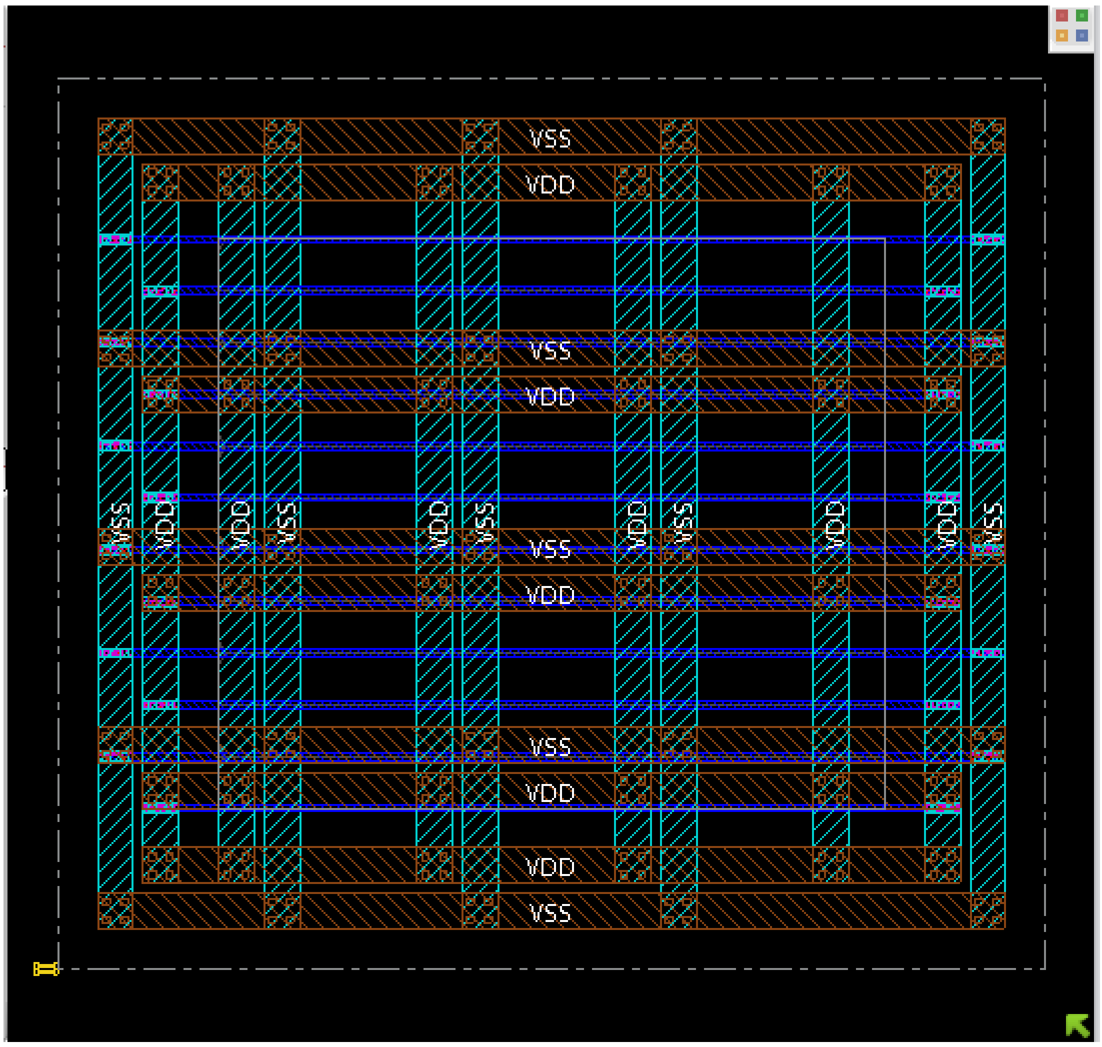


On adding Power Stripes, The Power mesh setup is complete as shown. However, There are no Vias that could connect Metal 9 or Metal 8 directly with Metal 1 [VDD or VSS of Standard Cells are generally made up of Metal 1].

The connection between the Highest and Lowest Metals is done through Stacking of Vias done using “Special Route”.

To perform Special Route, **Select Route → Special Route → Add Nets → OK.**

After the Special Route is complete, all the Standard Cell Rows turn to the Color coded for Metal 1 as shown below.

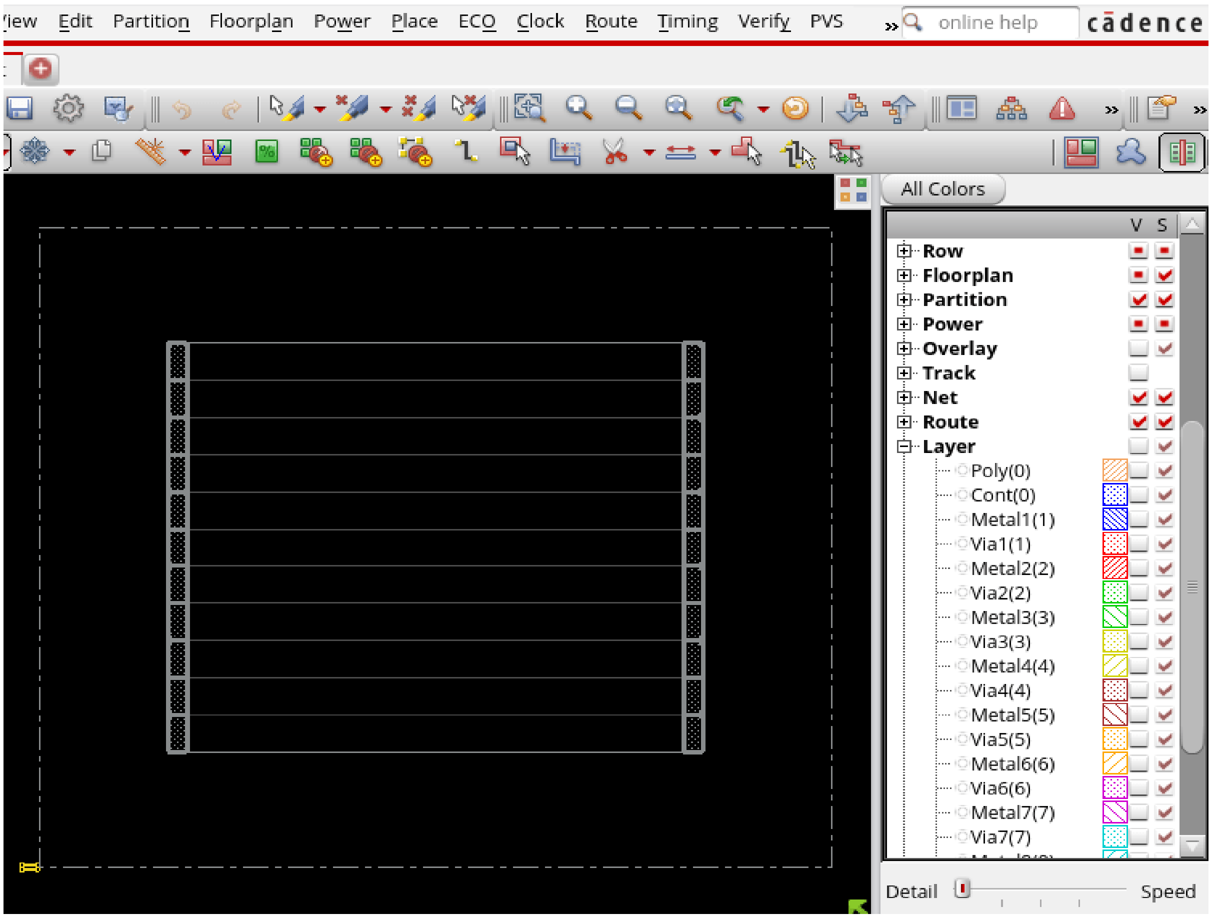


The complete Power Planning process makes sure Every Standard Cell receives enough power to operate smoothly.

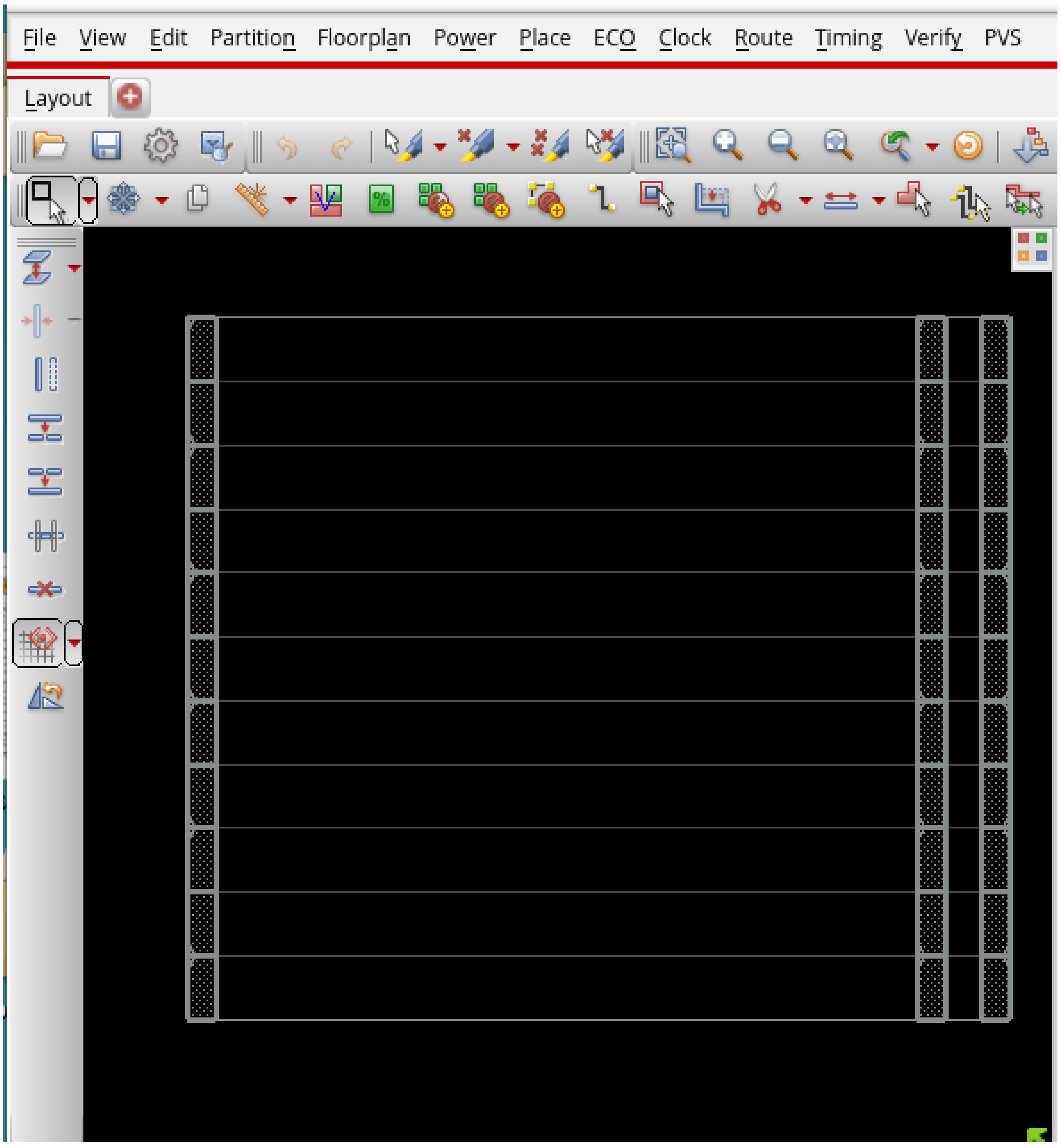
Module 4.4.1 : Pre - Placement

→ After Power Planning, a few Physical Cells are added namely, **End Caps and Well Taps**.
→ **End Caps :** They are Physical Cells which are added to the Left and Right Core Boundaries acting as blockages to avoid Standard Cells from moving out of boundary.
→ **Well Taps :** They act like Shunt Resistance to avoid Latch Up effects.

1. To add End Caps, Select **Place → Physical Cell → Add End Caps** and "Select" the FILL's from the available list. Higher Fills have Higher Widths. As shown Below, The End Caps are added below your Power Mesh.

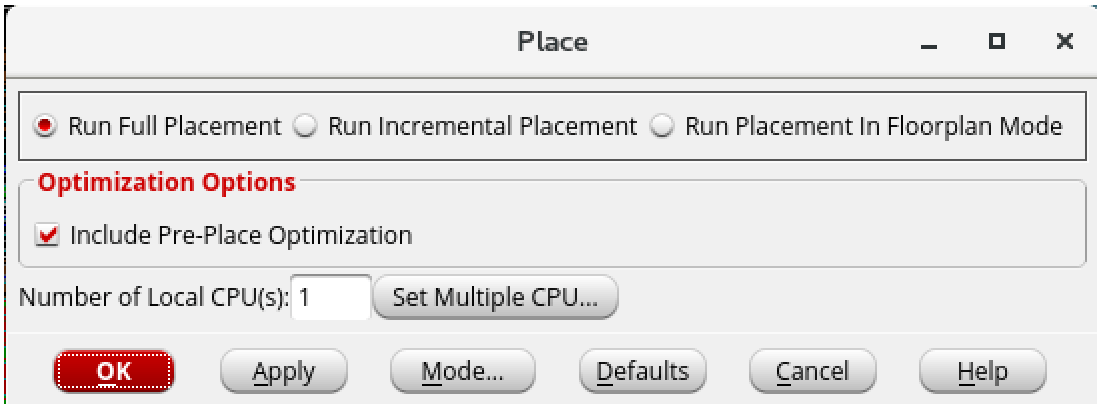


2. To add Well Taps, Select **Place → Physical Cell → Add Well Tap → Select → FillX** [X → Strength of Fill = 1,2,4 etc] **→ Distance Interval** [Could be given in range of 30-45u] **→ OK**



Module 4.5 : Placement

1. The Placement stage deals with Placing of Standard Cells as well as Pins.
2. Select **Place → Place Standard Cell → Run Full Placement → Mode → Enable 'Place I/O Pins' → OK → OK .**

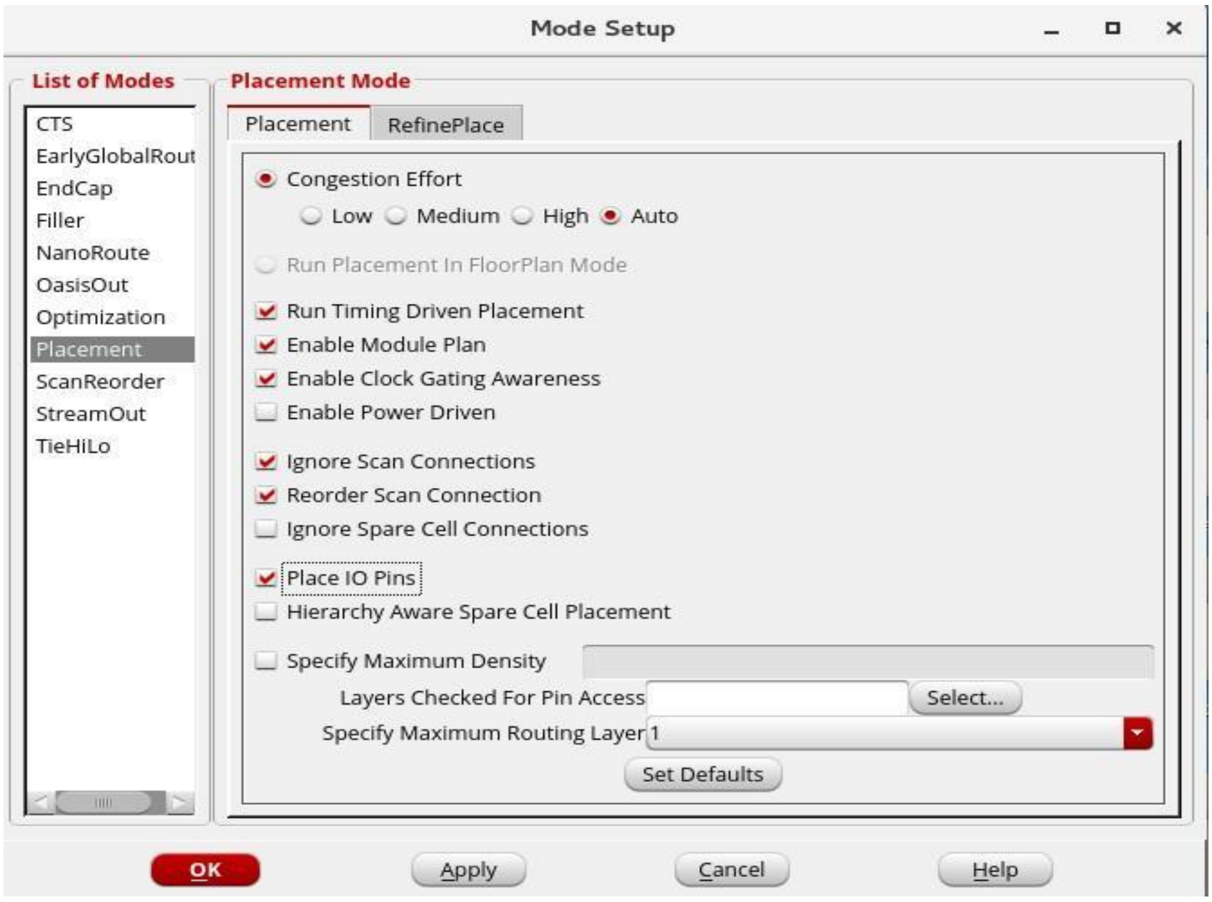

Place

[✓ Run Full Placement] Run Incremental Placement Run Placement In Floorplan Mode

Optimization Options

[✓] Include Pre-Place Optimization

Number of Local CPU(s): 1


Mode Setup

List of Modes

CTS
EarlyGlobalRout
EndCap
Filler
NanoRoute
OasisOut
Optimization
Placement
ScanReorder
StreamOut
TieHiLo

Placement Mode

Placement | RefinePlace

[✓] Congestion Effort
Low Medium High [✓ Auto]

Run Placement In FloorPlan Mode

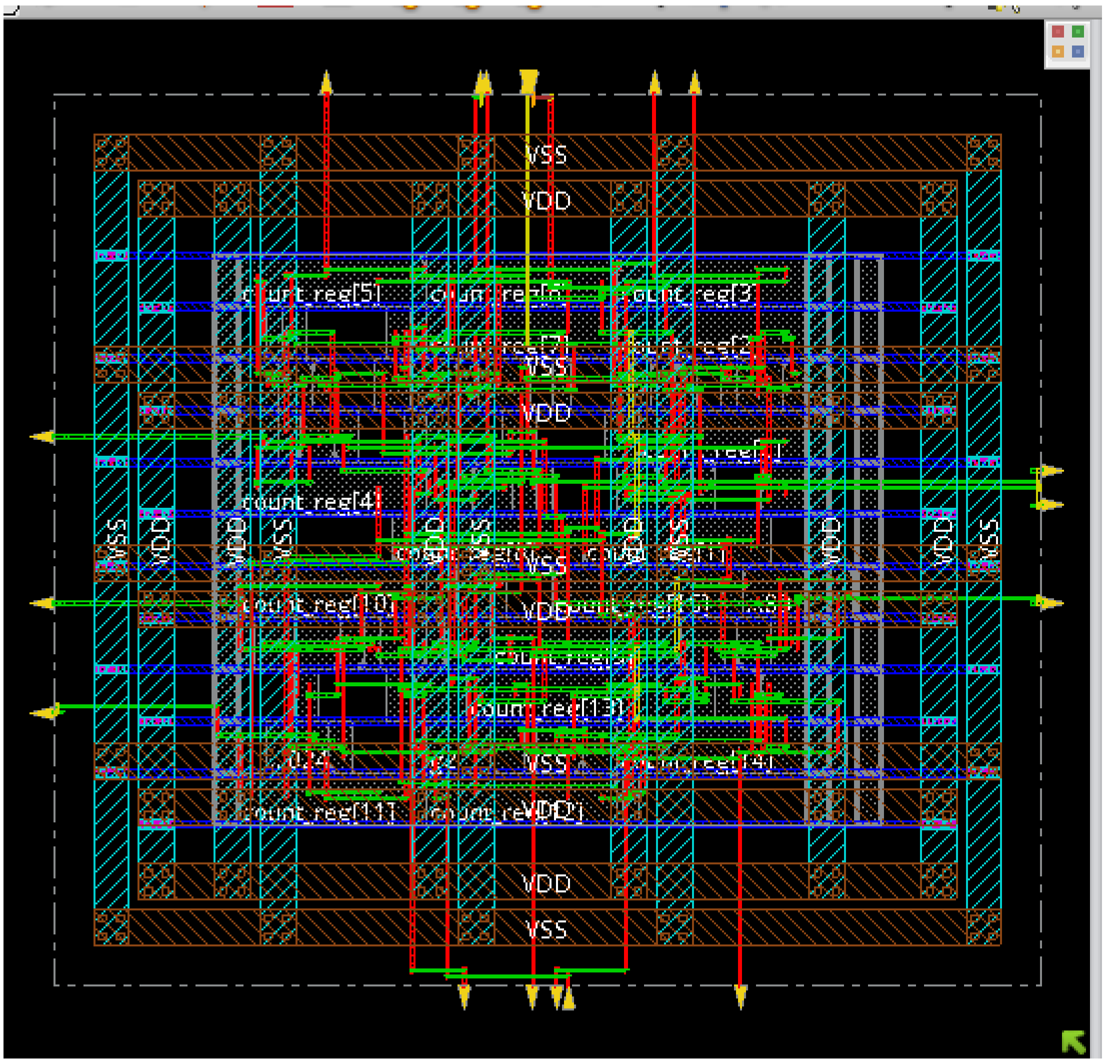
[✓] Run Timing Driven Placement
[✓] Enable Module Plan
[✓] Enable Clock Gating Awareness
[ ] Enable Power Driven

[✓] Ignore Scan Connections
[✓] Reorder Scan Connection
[ ] Ignore Spare Cell Connections

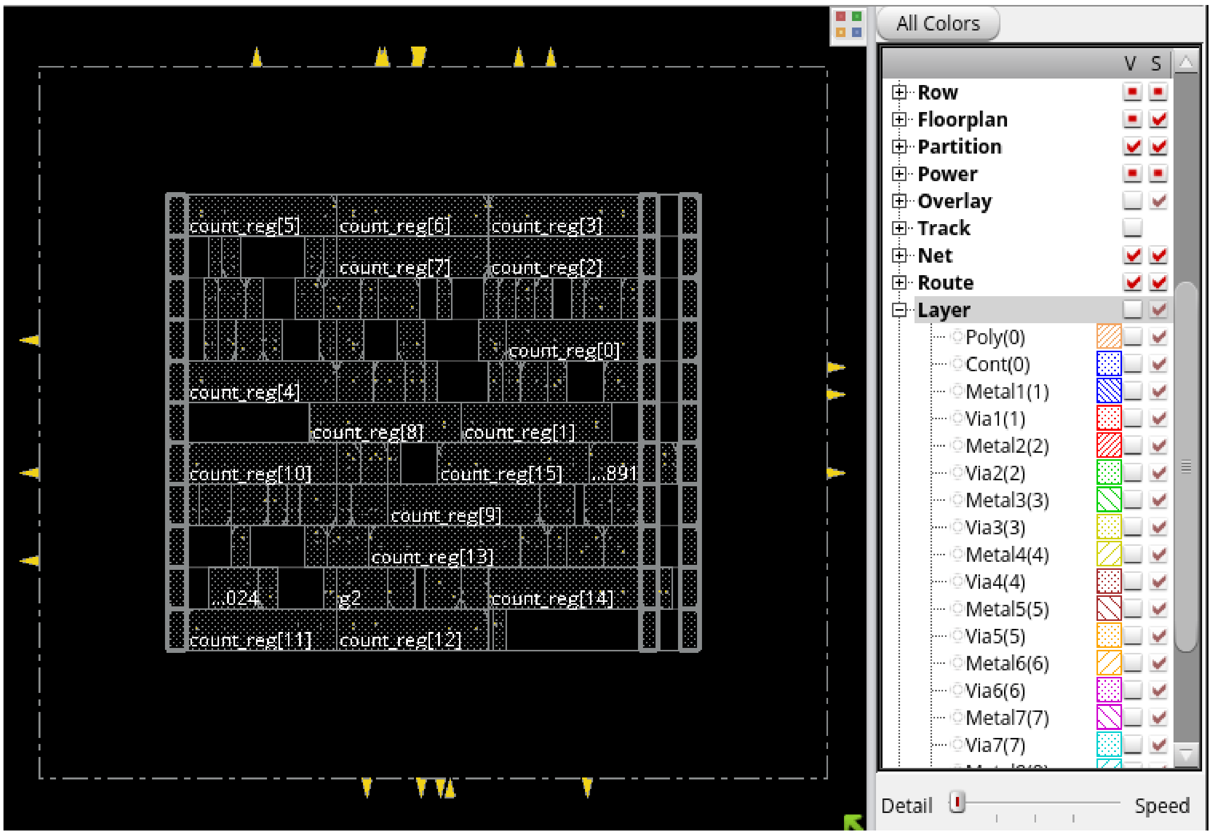
[✓] Place IO Pins
[ ] Hierarchy Aware Spare Cell Placement

[ ] Specify Maximum Density
Layers Checked For Pin Access
Specify Maximum Routing Layer 1

All the Standard Cells and Pins are placed as per the communication between them, i.e., Two communicating Cells are placed as close as possible so that shorter Net lengths can be used for connections as Shorter Net Lengths enable Better Timing Results.



Placed Design



Standard Cells Placed

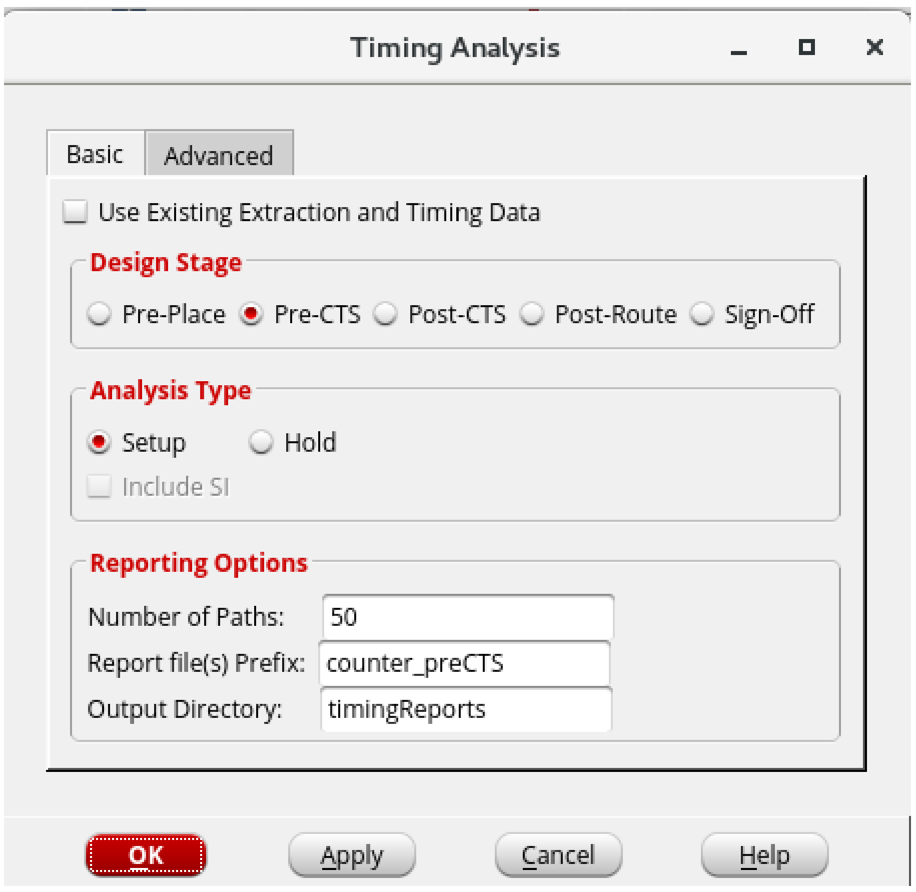
You can toggle the Layer Visibility from the list on the Right.

Report Generation and Optimization :

→ Timing Report :

To generate Timing Report, **Timing → Report Timing → Design Stage – PreCTS → Analysis Type – Setup → OK**

The Timing report Summary can be seen on the Terminal.



[ ] Use Existing Extraction and Timing Data

Design Stage: Pre-Place [✓ Pre-CTS] Post-CTS Post-Route Sign-Off

Analysis Type: [✓ Setup] Hold

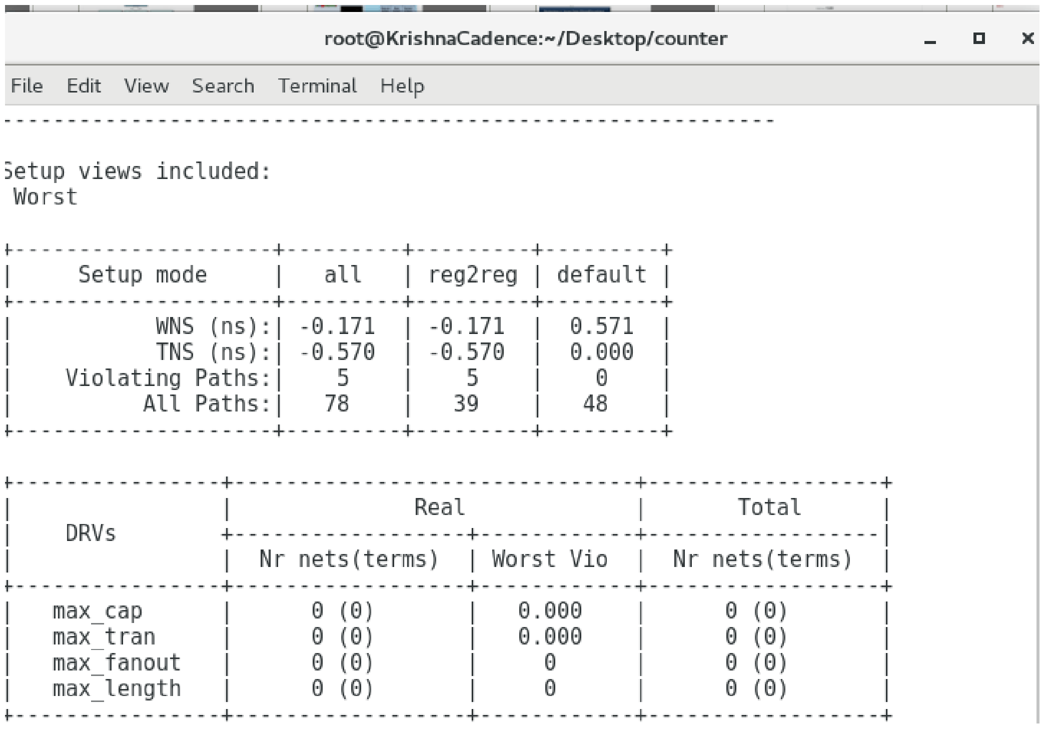
[ ] Include SI

Reporting Options

Number of Paths: 50

Report file(s) Prefix: counter_preCTS

Output Directory: timingReports

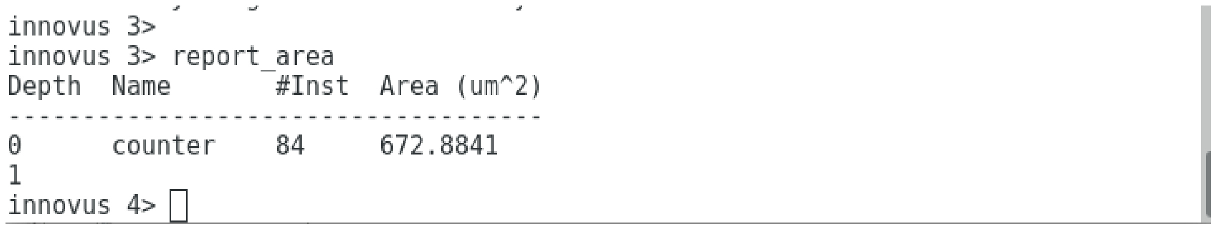

root@KrishnaCadence:~/Desktop/counter
File Edit View Search Terminal Help

```
------------------------------------------------------------

Setup views included:
 Worst

+-------------------+----------+---------+----------+
|     Setup mode    |   all    | reg2reg | default  |
+-------------------+----------+---------+----------+
|           WNS (ns):| -0.171  | -0.171  |  0.571   |
|           TNS (ns):| -0.570  | -0.570  |  0.000   |
|    Violating Paths:|    5    |    5    |    0     |
|          All Paths:|   78    |   39    |   48     |
+-------------------+----------+---------+----------+

+----------------+-------------------------------+------------------+
|                |             Real              |       Total      |
|    DRVs        +------------------+------------+------------------|
|                |  Nr nets(terms)  | Worst Vio  |  Nr nets(terms)  |
+----------------+------------------+------------+------------------+
|   max_cap      |      0 (0)       |   0.000    |      0 (0)       |
|   max_tran     |      0 (0)       |   0.000    |      0 (0)       |
|   max_fanout   |      0 (0)       |     0      |      0 (0)       |
|   max_length   |      0 (0)       |     0      |      0 (0)       |
+----------------+------------------+------------+------------------+
```

→ Area Report :

To generate Area Report, Switch to the Terminal and type the command ,

report_area to see the Cell Count and Area Occupied.

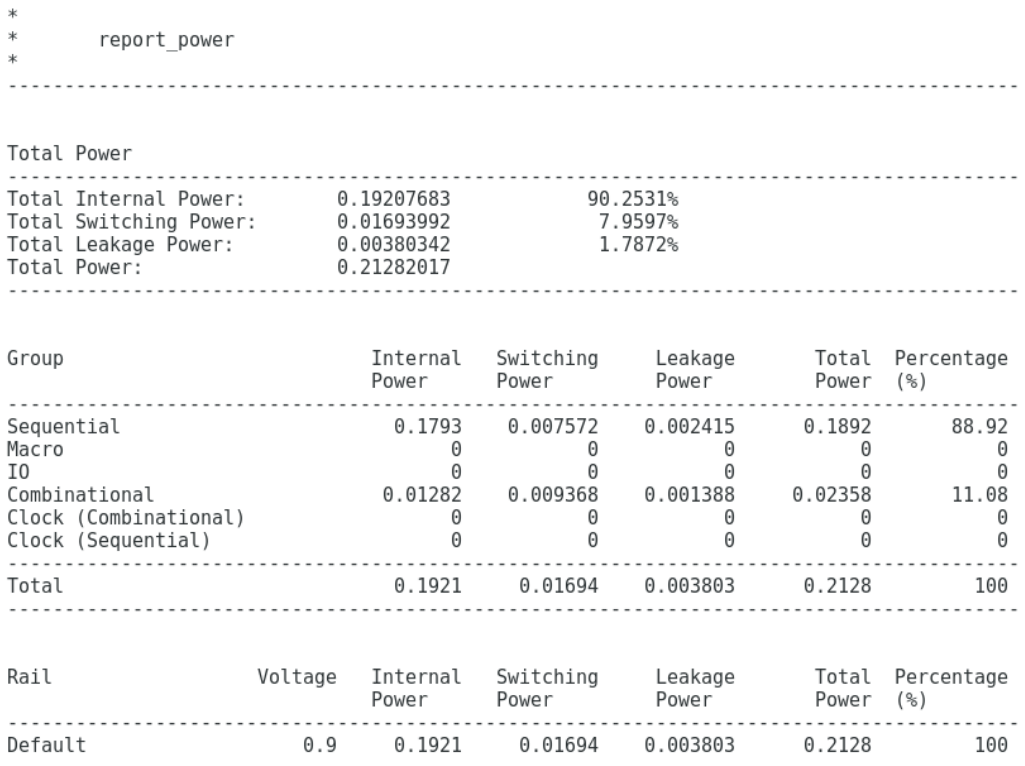


```
innovus 3>
innovus 3> report_area
Depth  Name       #Inst  Area (um^2)
------------------------------------
0      counter    84     672.8841
1
innovus 4>
```

→ Power report :

To generate Power Report, In the Terminal type the command

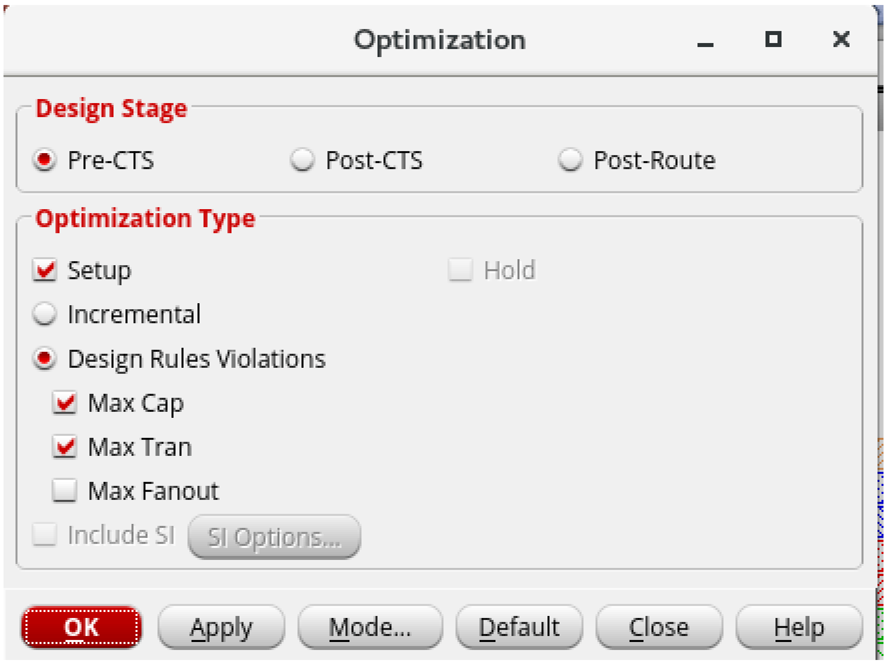
report_power to see the Power Consumption numbers.



```
*
*       report_power
*
-----------------------------------------------------------------------------------------


Total Power
-----------------------------------------------------------------------------------------
Total Internal Power:       0.19207683           90.2531%
Total Switching Power:      0.01693992            7.9597%
Total Leakage Power:        0.00380342            1.7872%
Total Power:                0.21282017
-----------------------------------------------------------------------------------------


Group                           Internal   Switching     Leakage       Total  Percentage
                                Power      Power         Power         Power  (%)
-----------------------------------------------------------------------------------------
Sequential                        0.1793    0.007572    0.002415      0.1892       88.92
Macro                                  0           0           0           0           0
IO                                     0           0           0           0           0
Combinational                    0.01282    0.009368    0.001388     0.02358       11.08
Clock (Combinational)                  0           0           0           0           0
Clock (Sequential)                     0           0           0           0           0
-----------------------------------------------------------------------------------------
Total                             0.1921     0.01694    0.003803      0.2128         100
-----------------------------------------------------------------------------------------


Rail                  Voltage   Internal   Switching     Leakage       Total  Percentage
                                Power      Power         Power         Power  (%)
-----------------------------------------------------------------------------------------
Default                   0.9     0.1921     0.01694    0.003803      0.2128         100
```

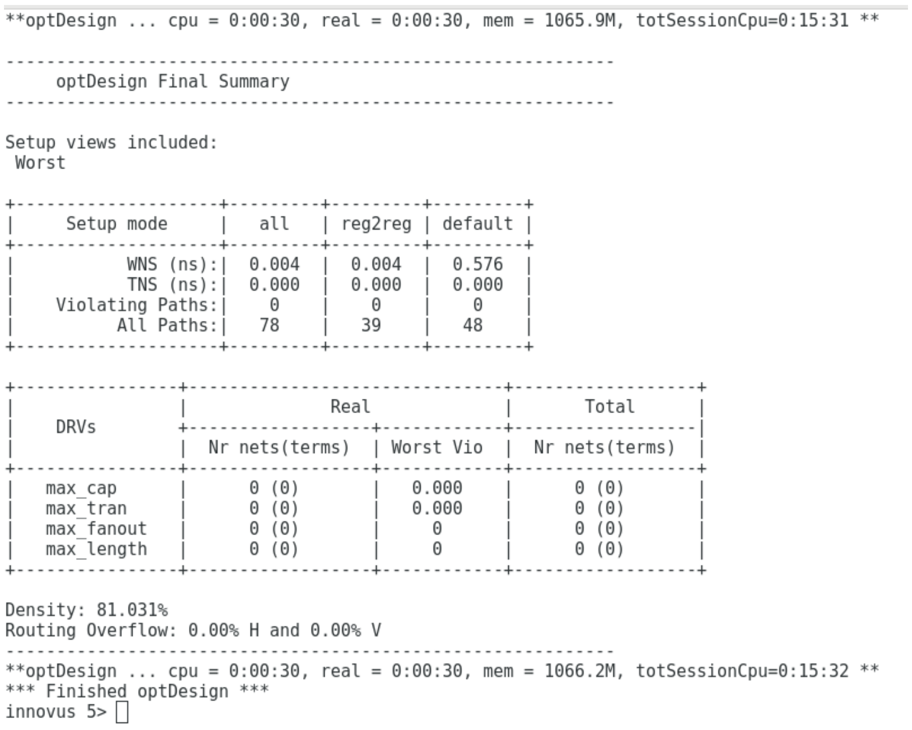
Design Optimization :

To optimize the Design, Select **ECO → Optimize Design → Design Stage [PreCTS] → Optimization Type – Setup → OK**


Optimization

Design Stage
[✓ Pre-CTS] Post-CTS Post-Route

Optimization Type
[✓ Setup] Hold
Incremental
[✓ Design Rules Violations]
[✓ Max Cap]
[✓ Max Tran]
Max Fanout
Include SI


```
**optDesign ... cpu = 0:00:30, real = 0:00:30, mem = 1065.9M, totSessionCpu=0:15:31 **

------------------------------------------------------------
     optDesign Final Summary
------------------------------------------------------------

Setup views included:
 Worst

+--------------------+---------+---------+---------+
|     Setup mode     |   all   | reg2reg | default |
+--------------------+---------+---------+---------+
|           WNS (ns):|  0.004  |  0.004  |  0.576  |
|           TNS (ns):|  0.000  |  0.000  |  0.000  |
|    Violating Paths:|    0    |    0    |    0    |
|          All Paths:|   78    |   39    |   48    |
+--------------------+---------+---------+---------+

+----------------+-------------------------------+------------------+
|                |              Real             |       Total      |
|    DRVs        +------------------+------------+------------------|
|                |  Nr nets(terms)  | Worst Vio  |  Nr nets(terms)  |
+----------------+------------------+------------+------------------+
|   max_cap      |     0 (0)        |   0.000    |     0 (0)        |
|   max_tran     |     0 (0)        |   0.000    |     0 (0)        |
|   max_fanout   |     0 (0)        |     0      |     0 (0)        |
|   max_length   |     0 (0)        |     0      |     0 (0)        |
+----------------+------------------+------------+------------------+

Density: 81.031%
Routing Overflow: 0.00% H and 0.00% V
------------------------------------------------------------
**optDesign ... cpu = 0:00:30, real = 0:00:30, mem = 1066.2M, totSessionCpu=0:15:32 **
*** Finished optDesign ***
innovus 5> 
```

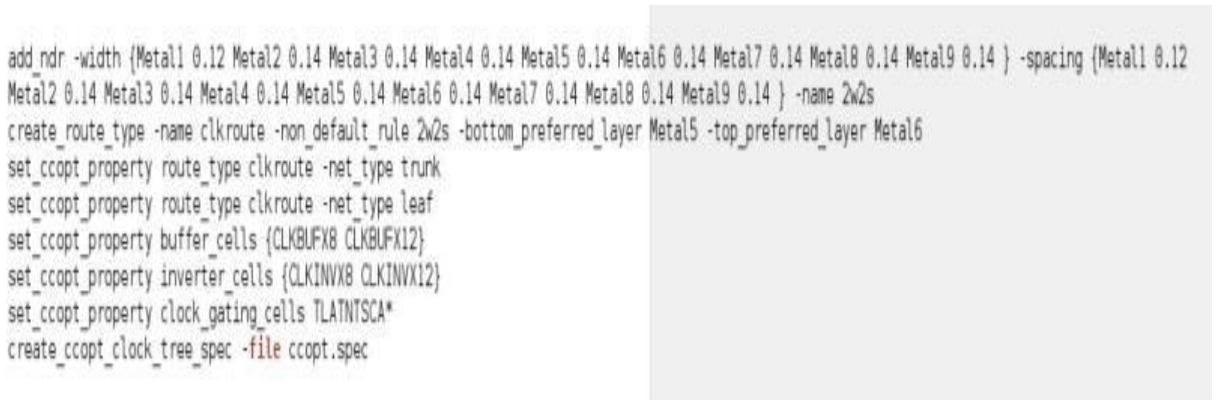
This step Optimizes your design in terms of Timing, Area and Power.

You can Generate Timing, Area, Power in similar way as above report Post – Optimization to compare the Reports.

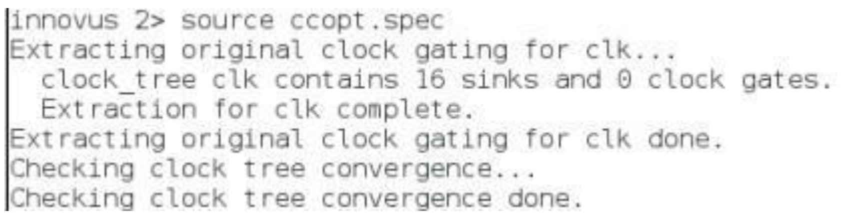
Module 4.6 : Clock Tree Synthesis

The CTS Stage is meant to build a Clock Distribution Network such that every Register (Flip Flop) acquires Clock at the same time (Atleast Approximately) to keep them in proper communication.

A Script can be used to Build the Clock Tree as follows :

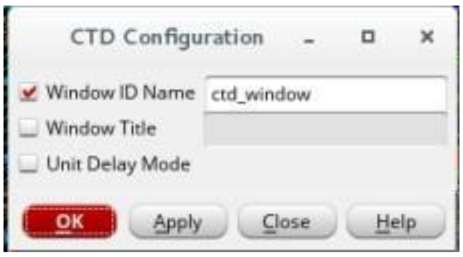


```
add_ndr -width {Metal1 0.12 Metal2 0.14 Metal3 0.14 Metal4 0.14 Metal5 0.14 Metal6 0.14 Metal7 0.14 Metal8 0.14 Metal9 0.14 } -spacing {Metal1 0.12
Metal2 0.14 Metal3 0.14 Metal4 0.14 Metal5 0.14 Metal6 0.14 Metal7 0.14 Metal8 0.14 Metal9 0.14 } -name 2w2s
create_route_type -name clkroute -non_default_rule 2w2s -bottom_preferred_layer Metal5 -top_preferred_layer Metal6
set_ccopt_property route_type clkroute -net_type trunk
set_ccopt_property route_type clkroute -net_type leaf
set_ccopt_property buffer_cells {CLKBUFX8 CLKBUFX12}
set_ccopt_property inverter_cells {CLKINVX8 CLKINVX12}
set_ccopt_property clock_gating_cells TLATNTSCA*
create_ccopt_clock_tree_spec -file ccopt.spec
```

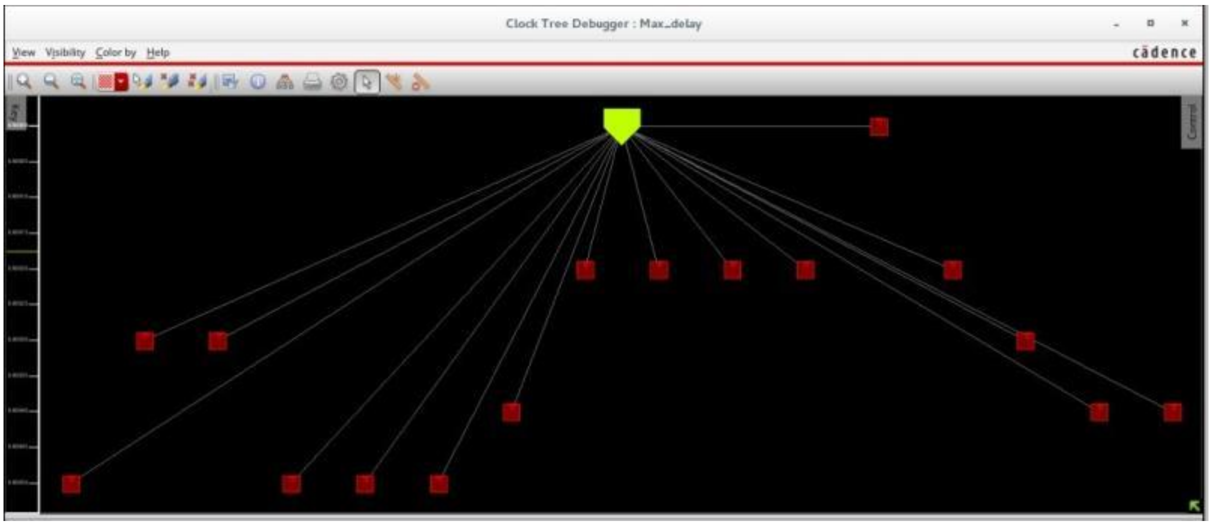


```
innovus 2> source ccopt.spec
Extracting original clock gating for clk...
  clock_tree clk contains 16 sinks and 0 clock gates.
  Extraction for clk complete.
Extracting original clock gating for clk done.
Checking clock tree convergence...
Checking clock tree convergence done.
```

Source the Script as shown in the above snapshot through the Terminal and then Select **Clock → CCOpt Clock Tree Debugger → OK** to build and view clock tree.



- [✓] Window ID Name: ctd_window
- [ ] Window Title
- [ ] Unit Delay Mode


Clock Tree Debugger : Max_delay
View Visibility Color by Help
cādence

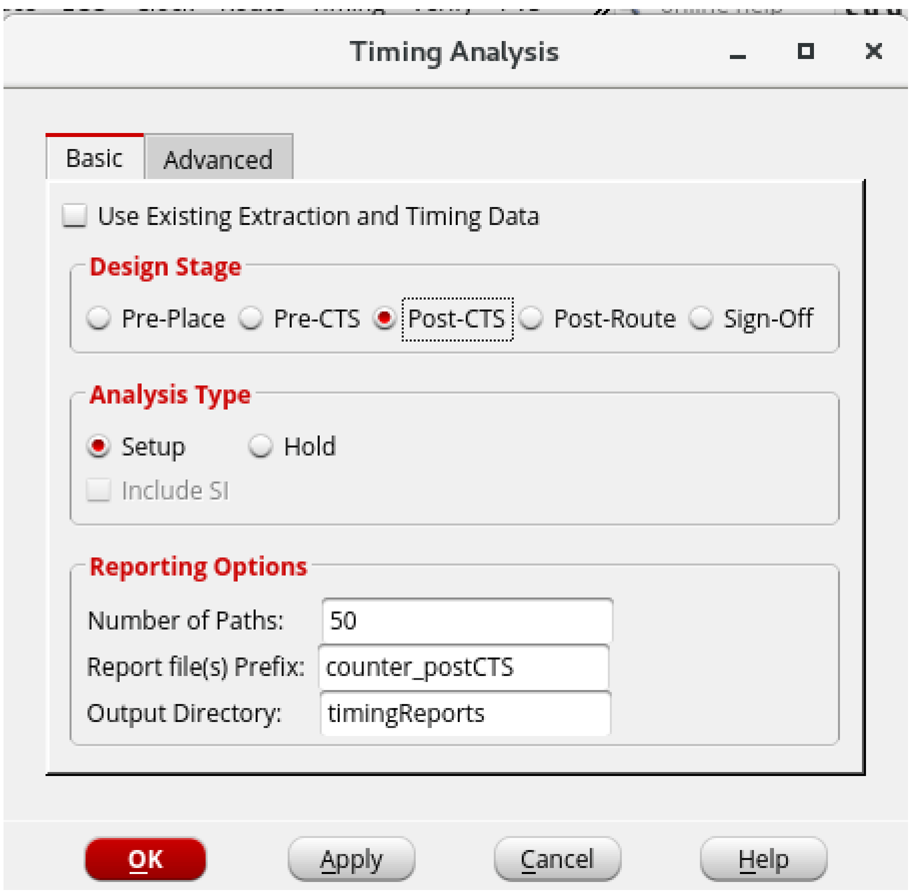
The Red Boxes are the Clock Pins of various Flip Flops in the Design while Yellow Pentagon on the top represents Clock Source.

The Clock Tree is built with Clock Buffers and Clock Inverters added to boost up the Clock Signal.

Report Generation and Design Optimization :

CTS Stage adds real clock into the Design and hence “Hold” Analysis also becomes prominent. Hence, **Optimizations can be done for both Setup & Hold, Timing Reports are to be Generated for Setup and Hold Individually**.

Setup Timing Analysis :



[ ] Use Existing Extraction and Timing Data

Design Stage: Pre-Place Pre-CTS [✓ Post-CTS] Post-Route Sign-Off

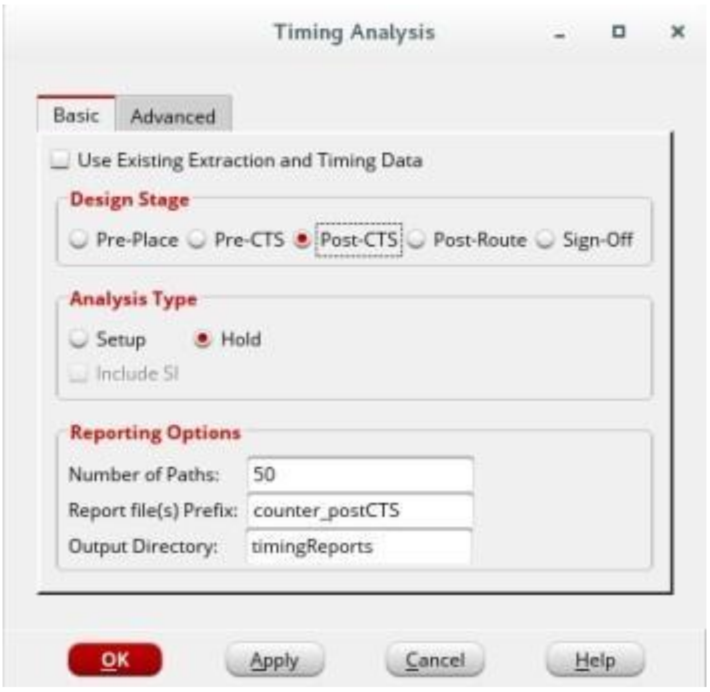
Analysis Type: [✓ Setup] Hold

[ ] Include SI

Reporting Options

Number of Paths:	50
Report file(s) Prefix:	counter_postCTS
Output Directory:	timingReports

Hold Timing Analysis :



Use Existing Extraction and Timing Data

Design Stage: Pre-Place [✓ Post-CTS] Pre-CTS Post-Route Sign-Off

Analysis Type: Setup [✓ Hold]

Include SI

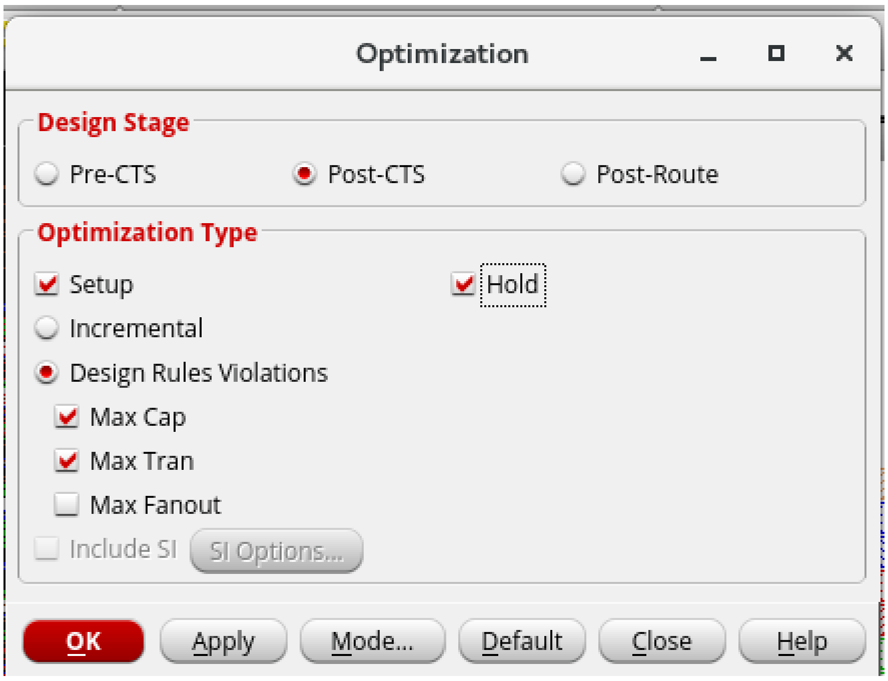
Reporting Options

Number of Paths:	50
Report file(s) Prefix:	counter_postCTS
Output Directory:	timingReports

For Area and Power Report Generation,

report_area & report_power commands can be used.

Design Optimizations :



Design Stage: Pre-CTS [✓ Post-CTS] Post-Route

Optimization Type

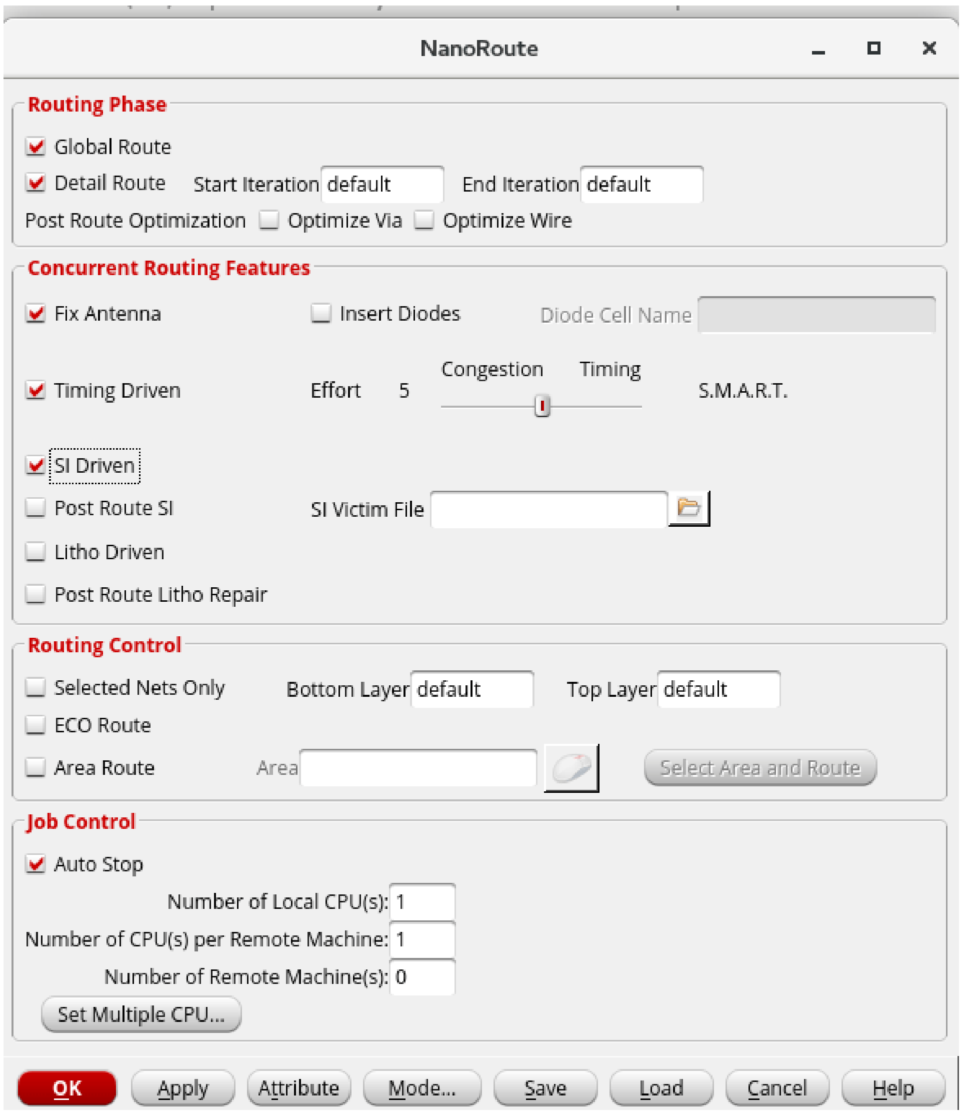
- [✓ Setup] [✓ Hold]
- Incremental
- [✓ Design Rules Violations]
  - [✓ Max Cap]
  - [✓ Max Tran]
  - Max Fanout
- Include SI

Module 4.7 : Routing

→ All the net connections shown in the GUI till CTS are only based on the Logical connectivity.

→ These connections are to be replaced with real Metals avoiding Opens, Shorts, Signal Integrity [Cross Talks], Antenna Violations etc.

To run Routing, Select **Route → Nano Route → Route** and enable **Timing Driven** and **SI Driven** for Design Physical Efficiency and Reliability.


NanoRoute

Routing Phase
- [✓] Global Route
- [✓] Detail Route   Start Iteration: default   End Iteration: default
- Post Route Optimization [ ] Optimize Via [ ] Optimize Wire

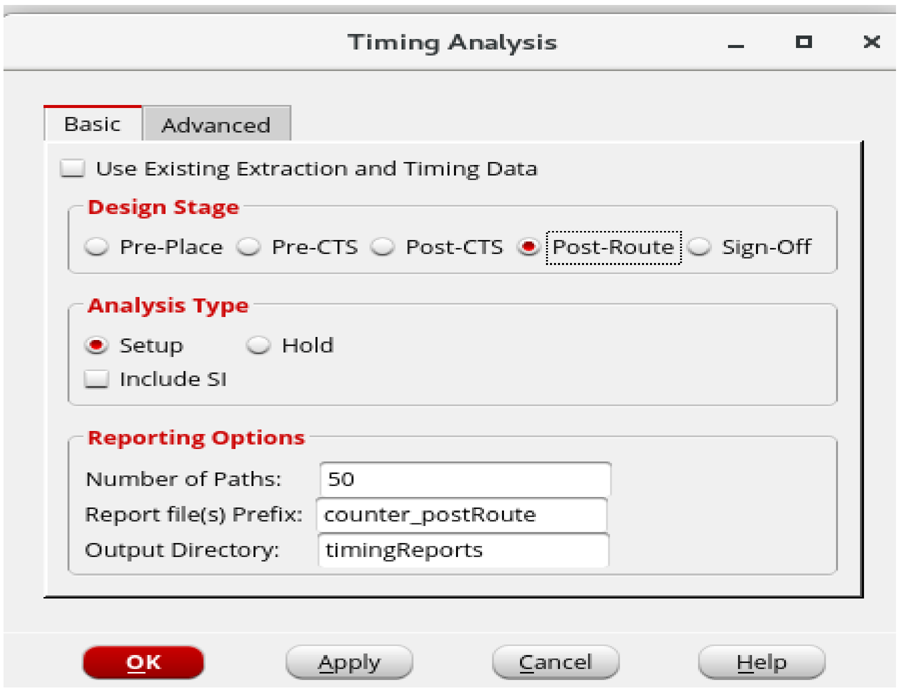
Concurrent Routing Features
- [✓] Fix Antenna   [ ] Insert Diodes   Diode Cell Name:
- [✓] Timing Driven   Effort 5   Congestion / Timing   S.M.A.R.T.
- [✓] SI Driven
- [ ] Post Route SI   SI Victim File:
- [ ] Litho Driven
- [ ] Post Route Litho Repair

Routing Control
- [ ] Selected Nets Only   Bottom Layer: default   Top Layer: default
- [ ] ECO Route
- [ ] Area Route   Area: Select Area and Route

Job Control
- [✓] Auto Stop
- Number of Local CPU(s): 1
- Number of CPU(s) per Remote Machine: 1
- Number of Remote Machine(s): 0
- Set Multiple CPU...

Report Generation and Design Optimization :

Setup Report :



Use Existing Extraction and Timing Data

Design Stage: Pre-Place Pre-CTS Post-CTS [✓ Post-Route] Sign-Off

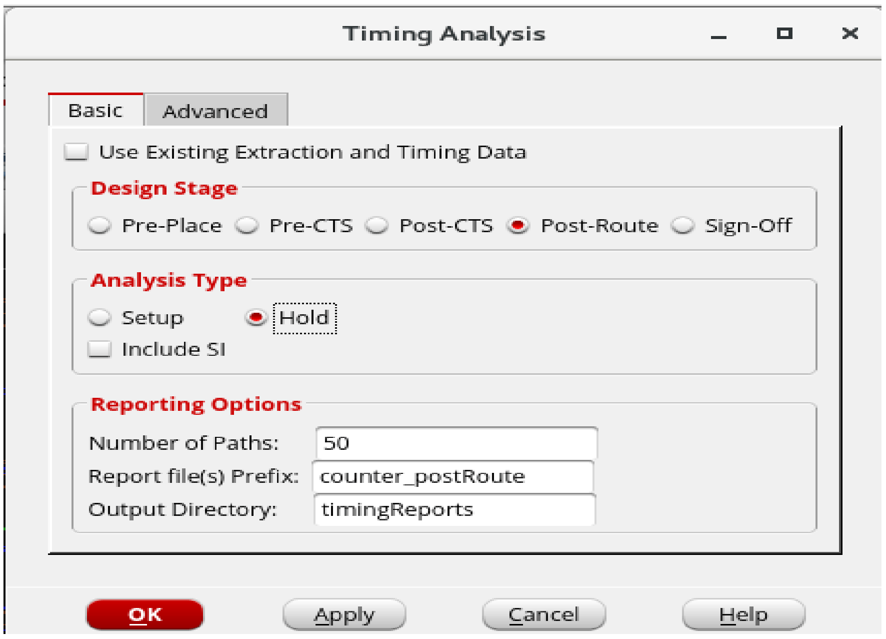
Analysis Type: [✓ Setup] Hold

Include SI

Reporting Options

Number of Paths:	50
Report file(s) Prefix:	counter_postRoute
Output Directory:	timingReports

Hold Report :



Use Existing Extraction and Timing Data

Design Stage: Pre-Place Pre-CTS Post-CTS [✓ Post-Route] Sign-Off

Analysis Type: Setup [✓ Hold]

Include SI

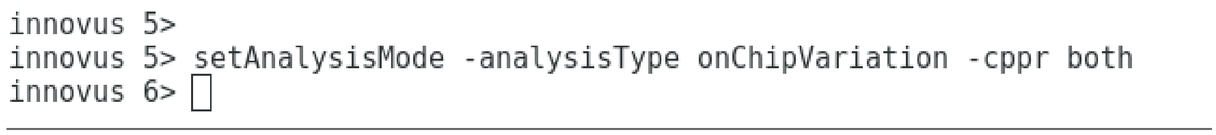
Reporting Options

Number of Paths:	50
Report file(s) Prefix:	counter_postRoute
Output Directory:	timingReports

Area and Power Reports :

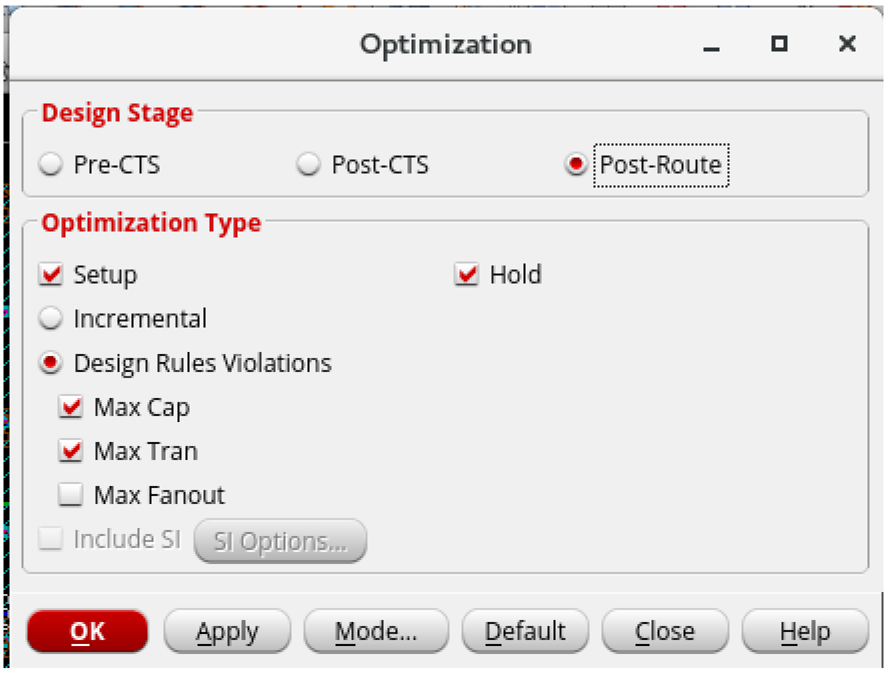
Use the commands **report_area** and **report_power** for Area and Power Reports respectively.

Design Optimization :



```
innovus 5>
innovus 5> setAnalysisMode -analysisType onChipVariation -cppr both
innovus 6>
```

Enter the above shown command in the Terminal in order to run the Design Optimization first Post-Route.



Optimization

Design Stage

Pre-CTS Post-CTS [✓ Post-Route]

Optimization Type

[✓ Setup] [✓ Hold]

Incremental

[✓ Design Rules Violations]

[✓ Max Cap]

[✓ Max Tran]

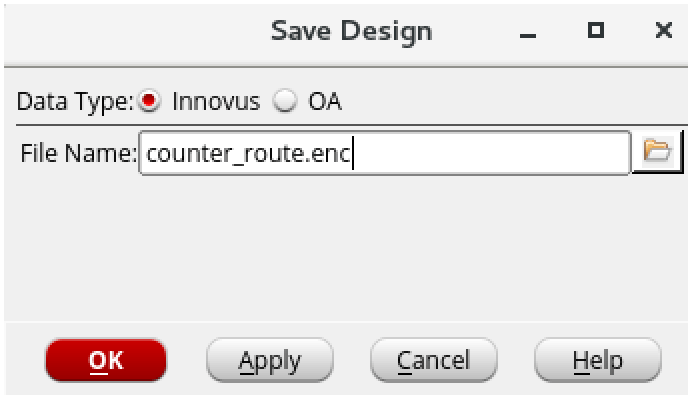
Max Fanout

Include SI

The Report generation is same as shown prior to Design Optimization.

Saving Database :

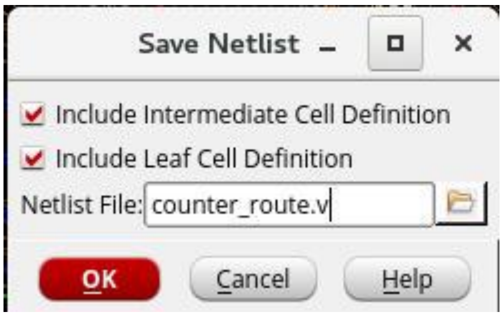
1. **Saving Design => File → Save Design → Data Type : Innovus → <DesignName>.enc → OK**



Data Type: [✓ Innovus] OA

File Name: counter_route.enc

2. **Saving Netlist => File → Save → Netlist → <NetlistName>.v → OK**



✓ Include Intermediate Cell Definition

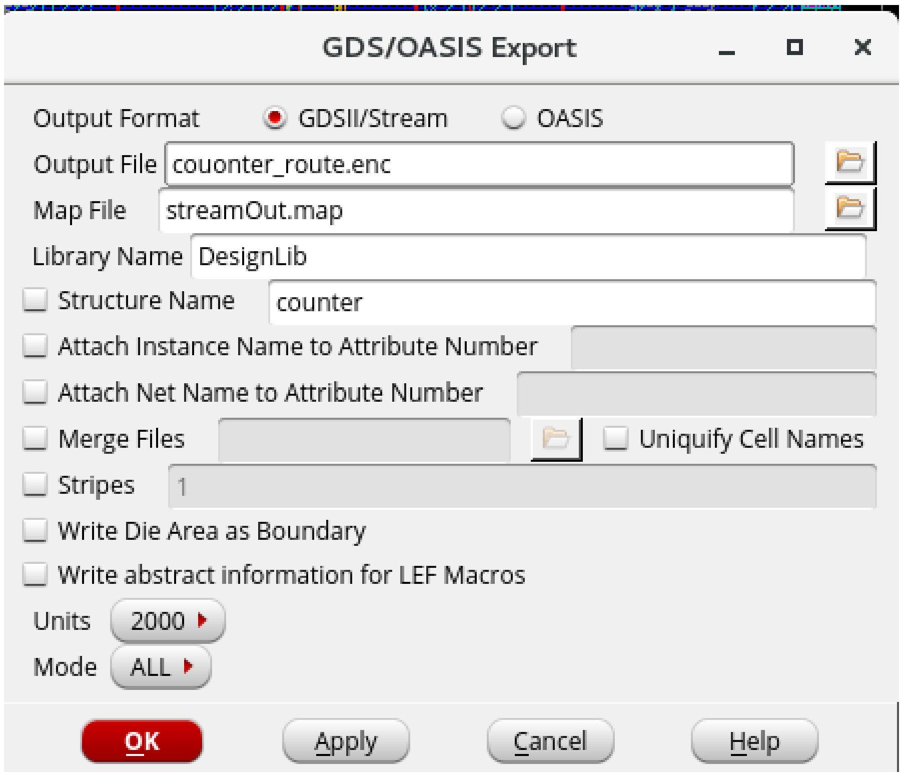
✓ Include Leaf Cell Definition

Netlist File: counter_route.v

It is recommended to save Netlist and Design at every stage.

To restore a Design Data Base, type **source <DesignName>.enc** in the terminal.

3. Saving GDS => File → Save → GDS/OASIS → <FileName>.gds → OK



GDS/OASIS Export

Output Format [✓ GDSII/Stream] OASIS

Output File: couonter_route.enc

Map File: streamOut.map

Library Name: DesignLib

☐ Structure Name: counter

☐ Attach Instance Name to Attribute Number

☐ Attach Net Name to Attribute Number

☐ Merge Files ☐ Uniquify Cell Names

☐ Stripes: 1

☐ Write Die Area as Boundary

☐ Write abstract information for LEF Macros

Units: 2000

Mode: ALL

The GDS File is a Binary Format File which is not Readable and is fed to the Fabrication unit with data of various layers used depicted in terms of Geometrical Shapes.

LAB 1 : INVERTER

Objective : To compile and simulate the verilog code of the Inverter circuit and observe the output waveform.

Inverter :

```
      `timescale 1ps/1ns
```

// Define our own Inverter,

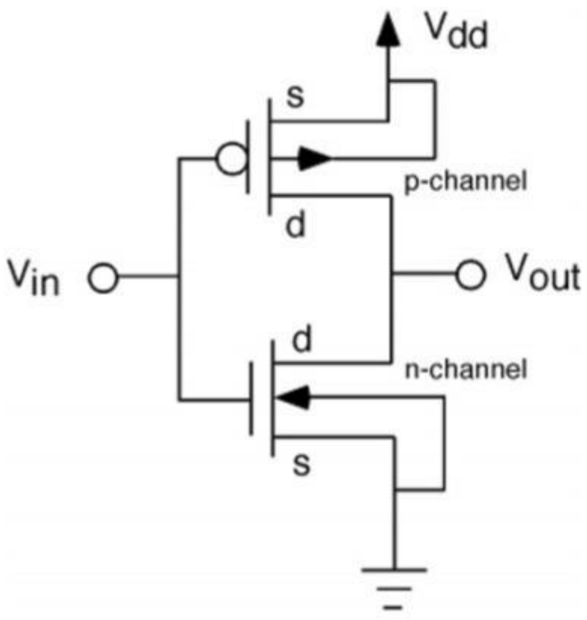
```
      module inverter (out, in);
```

// Declarations of I/O, Power and Ground Lines

```
      output out;
      input in;
      supply1 pwr;
      supply0 gnd;
```

// Instantiate pmos and nmos switches

```
      pmos (out, pwr, in);
      nmos (out, gnd, in);

      endmodule
```



Test Bench of Inverter :

```
`timescale 1ps/1ns
```

//Test Bench of Inverter Module

```
module inv_test;

wire out;
reg in;
```

//Instantiate inverter module

```
inverter uut (out, in);
```

//Display

```
begin
$display (“time=%d, $time, “ns”, “input=”,in,”output=”,out); end
```

//Apply Stimulus

```
initial
begin
in=1’b0;   #10   ;   display;
in=1’b1;   #10   ;   display;
in=1’bx;   #10   ;   display;
in=1’bz;   #10   ;   display;
end

endmodule
```

Constraints file for Synthesis file : inverter.sdc

```
set_input_delay -max 1.0 [get_ports “in”]
set_output_delay -min 1.0 [get_ports “out”]
```

Truth Table :

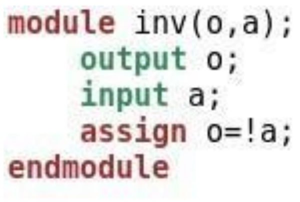
Input (in)	Output (out)
0	1
1	0
X	x
Z	x

In **Desktop** Create a folder to do the digital design flow. **Right click** in the Desktop and select **Open in terminal.**

1. mkdir filename represents creating the directory with the any filename.

2. cd filename represents moving to the current directory.

3. gedit filename.v represents opening the gedit window for writing the rtl coding as shown in Figure : 1.



```
module inv(o,a);
    output o;
    input a;
    assign o=!a;
endmodule
```

Figure : 1

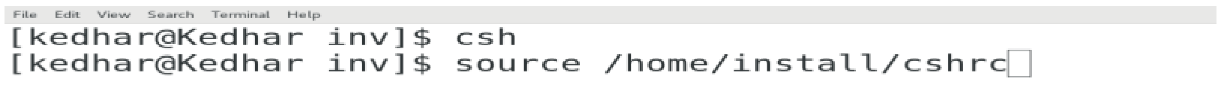
4. Save the rtl coding by pressing **Ctrl+s** or select the **save** option on the top right corner.

5. Similar way create a test bench program also.

Invoke the Cadence environment by typing the below commands as shown in Figure 2.

csh

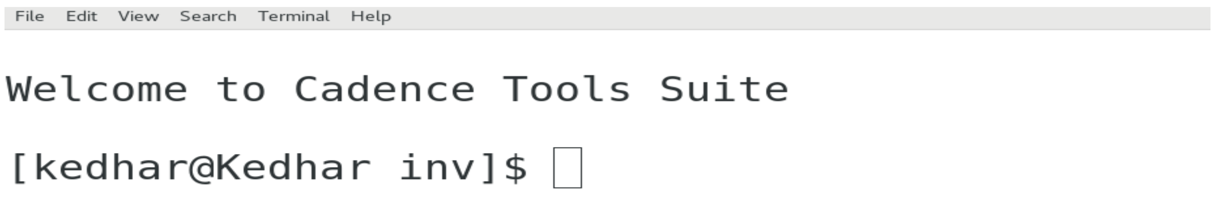
source *home/install/cshrc*



```
[kedhar@Kedhar inv]$ csh
[kedhar@Kedhar inv]$ source /home/install/cshrc
```

Figure : 2

After this you can see the window like as shown in Figure 3 below

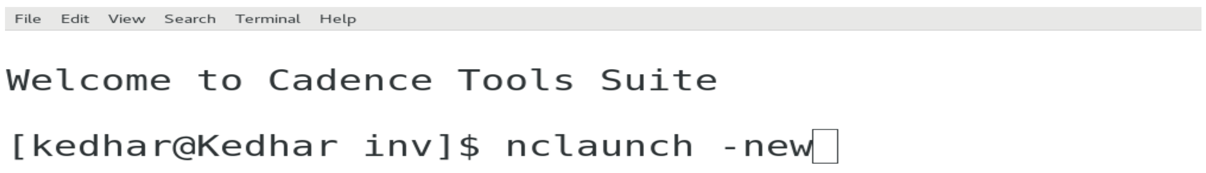


```
Welcome to Cadence Tools Suite

[kedhar@Kedhar inv]$
```

Figure : 3

Functional Simulation :

Use the following command to invoke user friendly GUI as shown in Figure 4:



```
Welcome to Cadence Tools Suite

[kedhar@Kedhar inv]$ nclaunch -new
```

Figure : 4

It will invoke the **nclaunch** window for functional simulation we can compile, elaborate and simulate it using **Multistep.**

The **NCLaunch** tool consists of a single main window that contains multiple browsers, which are integrated with the suite of NC tools. The integrated tools include the following:

1. Compilation
2. Elabarotion
3. Simulation

1. Compilation :

Compilation is the process of reading in source code and the analyzing the source code for syntax and semantic errors. Most HDLs including System Verilog supports compilation as a single file or multiple files using a compilation units.

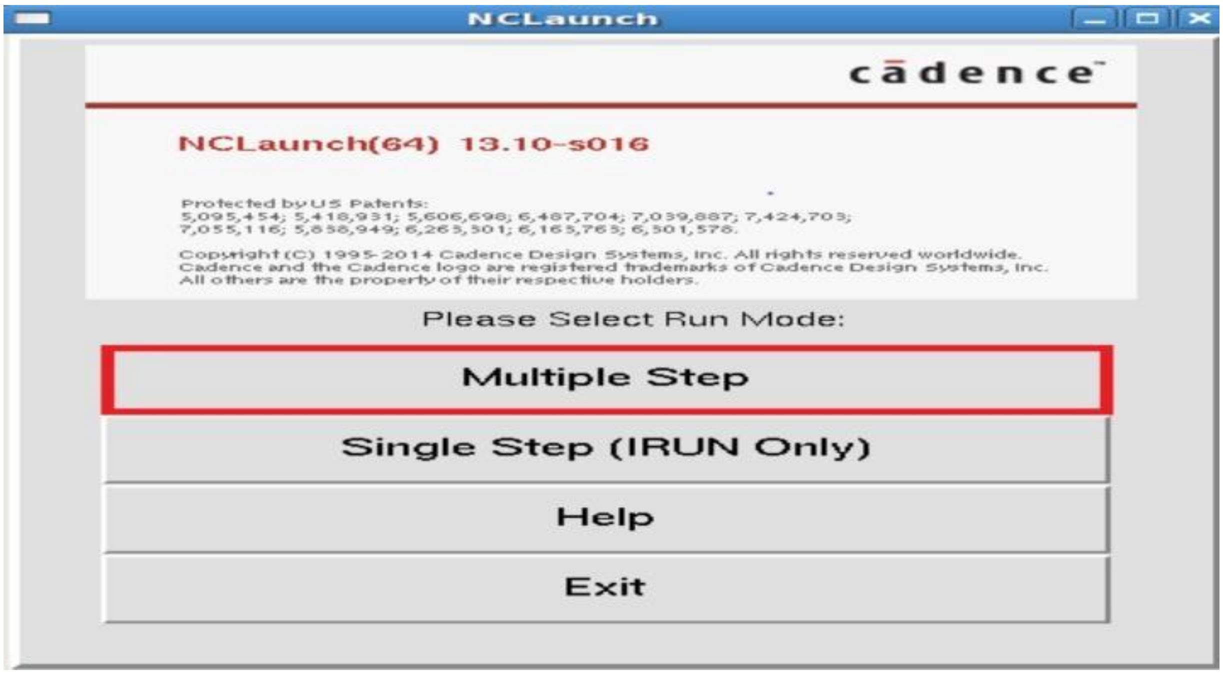
2. Elaboration :

Elaboration is the process that occurs between parsing and simulation. It binds modules to module instances, builds the model hierarchy, computes parameter values, resolve hierarchical names, establishes net connectivity, and prepares all of this for simulation.

3. Simulation :

Simulation is the process by which the design model coded in the HDL gets executed based on a given execution model. An HDL description of the design would consists of several concurrent process, assignments and some connections between then.

Select the Multiple Step option as shown in Figure 5:


NCLaunch

NCLaunch(64) 13.10-s016

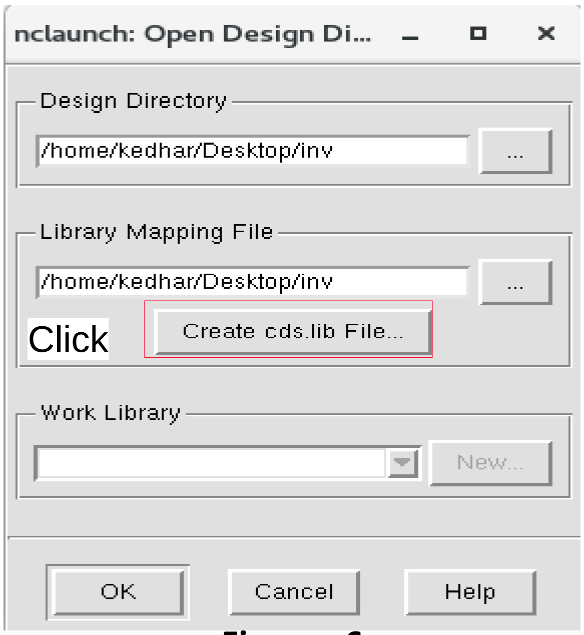
Protected by US Patents:
5,095,454; 5,418,931; 5,606,698; 6,487,704; 7,039,887; 7,424,703;
7,055,116; 5,858,949; 6,263,301; 6,163,763; 6,301,578.

Copyright (C) 1995-2014 Cadence Design Systems, Inc. All rights reserved worldwide.
Cadence and the Cadence logo are registered trademarks of Cadence Design Systems, Inc.
All others are the property of their respective holders.

Figure : 5

We can simulate a design using the **Incisive simulator.**

For that we have to Create the **cds.lib** and **hdl.var** files for to Compile, elaborate and simulate the design and test bench as shown in Figure 6.

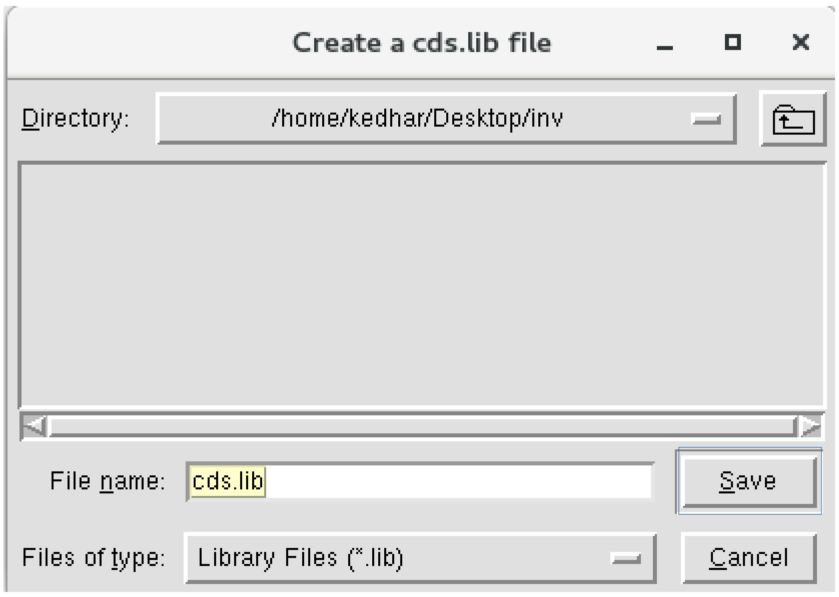

nclaunch: Open Design Di...

Design Directory: /home/kedhar/Desktop/inv

Library Mapping File: /home/kedhar/Desktop/inv

Figure : 6

Click the cds.lib file


Create a cds.lib file

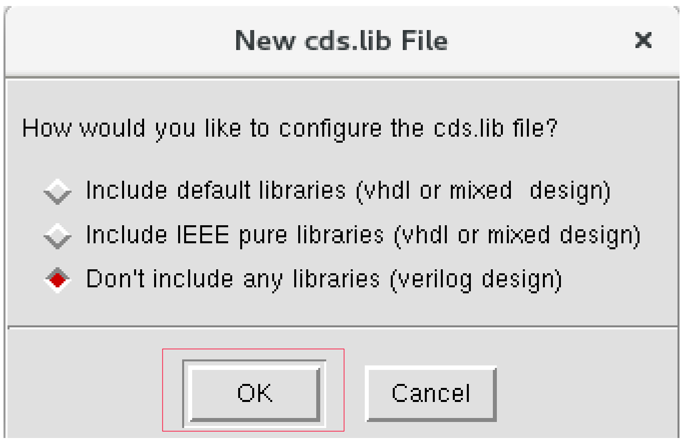
Directory: /home/kedhar/Desktop/inv

File name: cds.lib

Files of type: Library Files (*.lib)

Figure : 7

Save the file as shown in Figure 7.

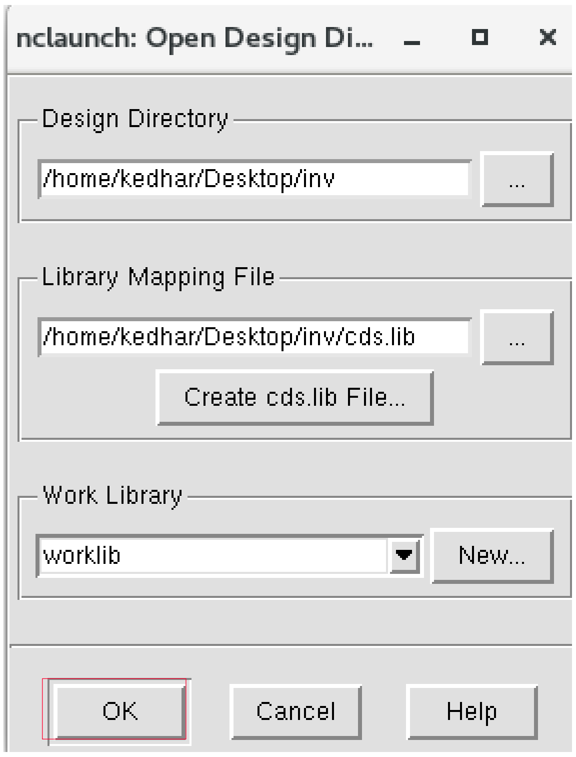

New cds.lib File

How would you like to configure the cds.lib file?

- Include default libraries (vhdl or mixed design)
- Include IEEE pure libraries (vhdl or mixed design)
- [✓ Don't include any libraries (verilog design)]

Figure : 8

Choose any option listed from above as shown in Figure 8.



Design Directory: /home/kedhar/Desktop/inv

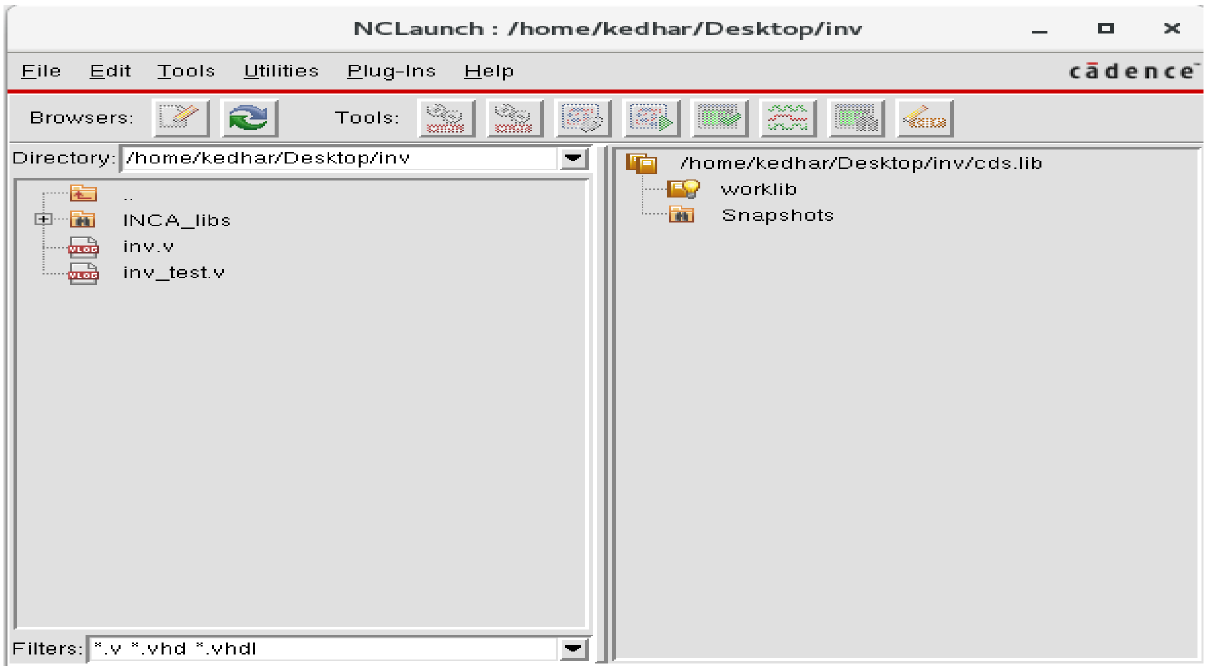
Library Mapping File: /home/kedhar/Desktop/inv/cds.lib

Create cds.lib File...

Work Library: worklib

Figure : 9

After that you can give OK as shown in Figure 9. After clicking on ok, you can see the Figure 10.



Directory: /home/kedhar/Desktop/inv

```
..
INCA_libs
inv.v
inv_test.v
```

```
/home/kedhar/Desktop/inv/cds.lib
    worklib
    Snapshots
```

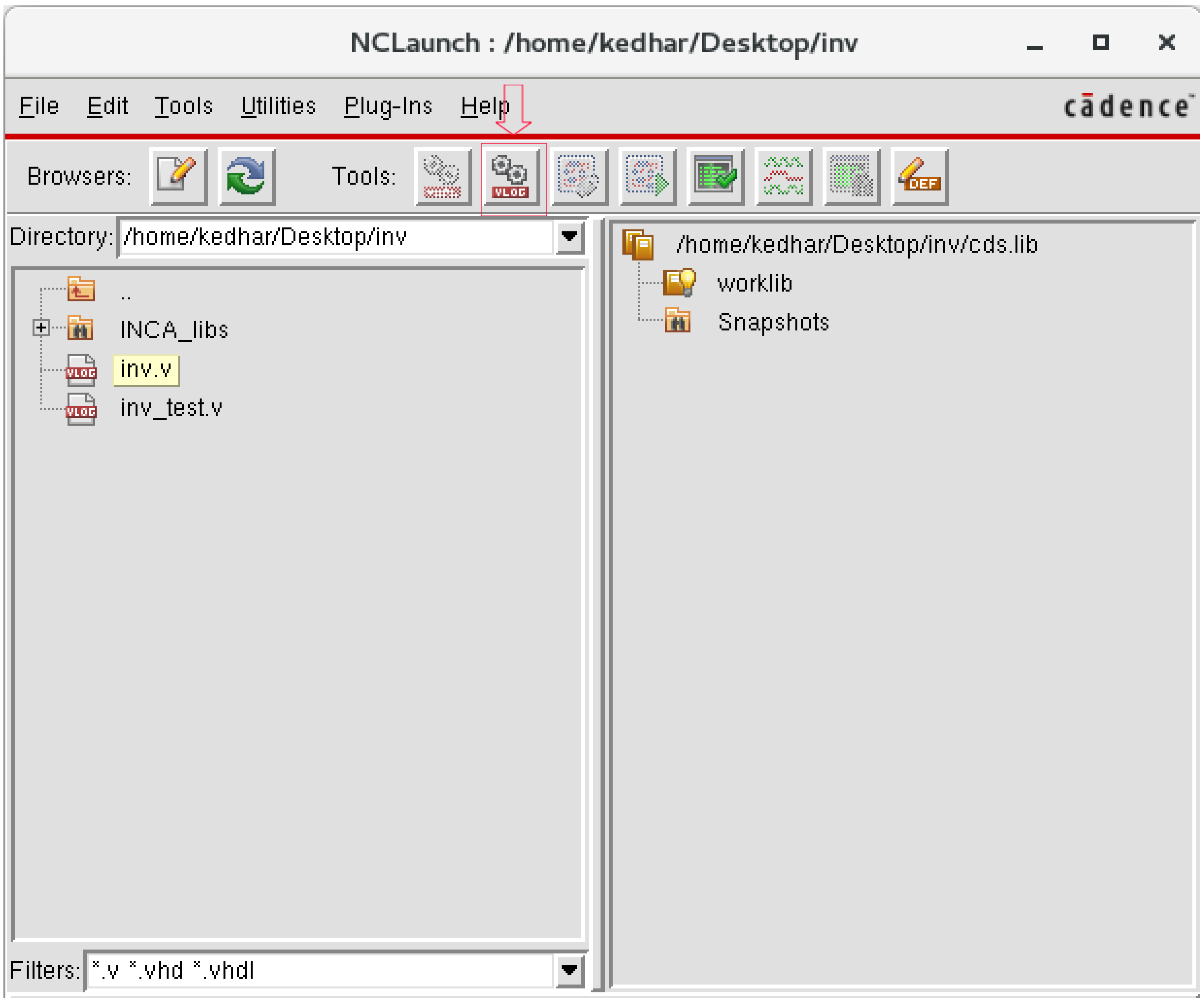
Filters: *.v *.vhd *.vhdl

Figure : 10

Left side you can see the **HDL files**. Right side of the window has **worklib** and **snapshots** directories listed.

Worklib is the directory where all the compiled codes are stored while **Snapshot** will have output of elaboration which in turn goes for simulation.

Compilation :



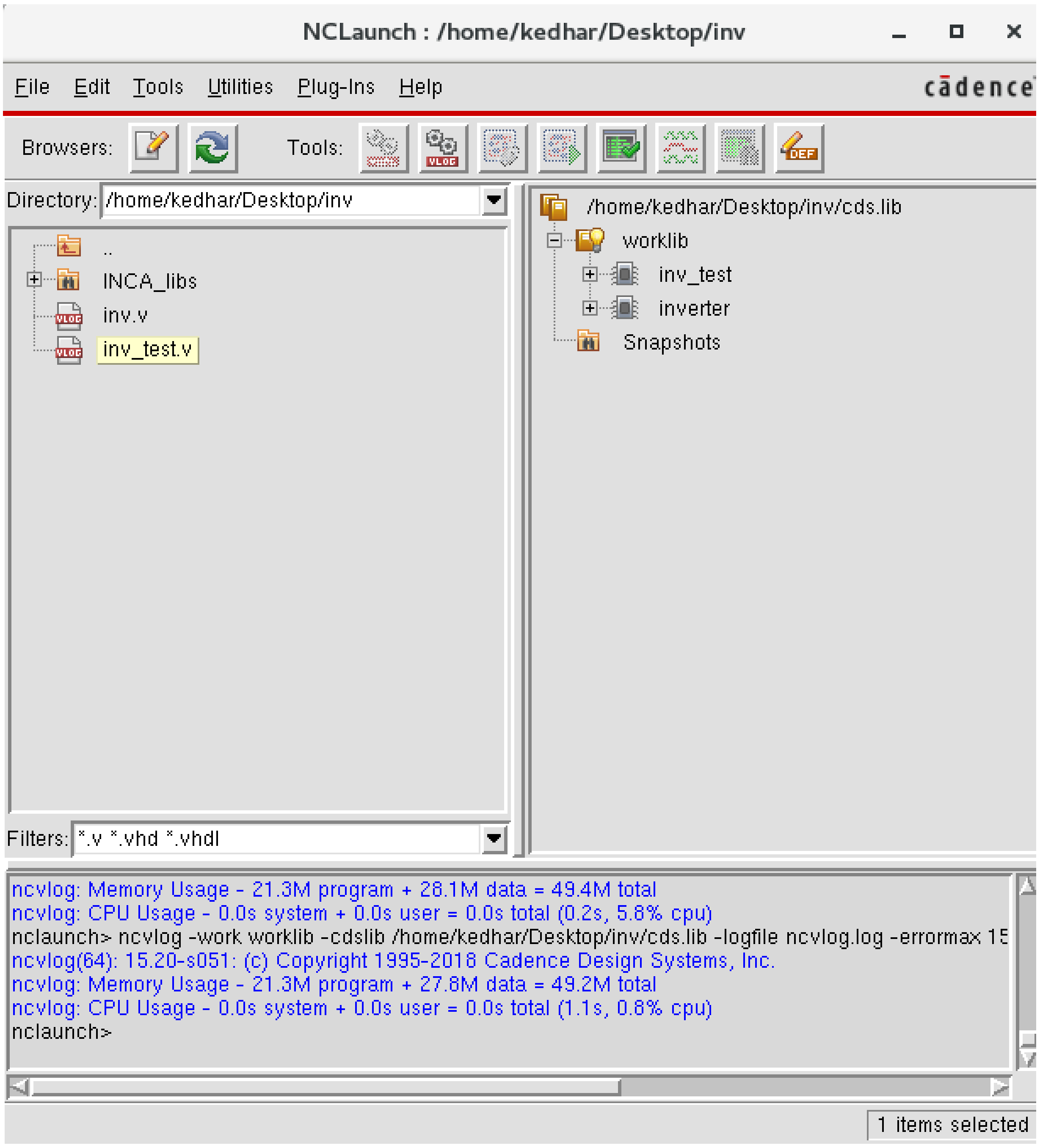
NCLaunch : /home/kedhar/Desktop/inv
File Edit Tools Utilities Plug-Ins Help
cādence
Browsers:
Tools:
VLOG
DEF
Directory: /home/kedhar/Desktop/inv
..
INCA_libs
inv.v
inv_test.v
/home/kedhar/Desktop/inv/cds.lib
worklib
Snapshots
Filters: *.v *.vhd *.vhdl

Figure : 11

Left side select the file and in **Tools : launch verilog compiler with current selection** will get enable as shown in Figure 11.

Click it to compile the code.

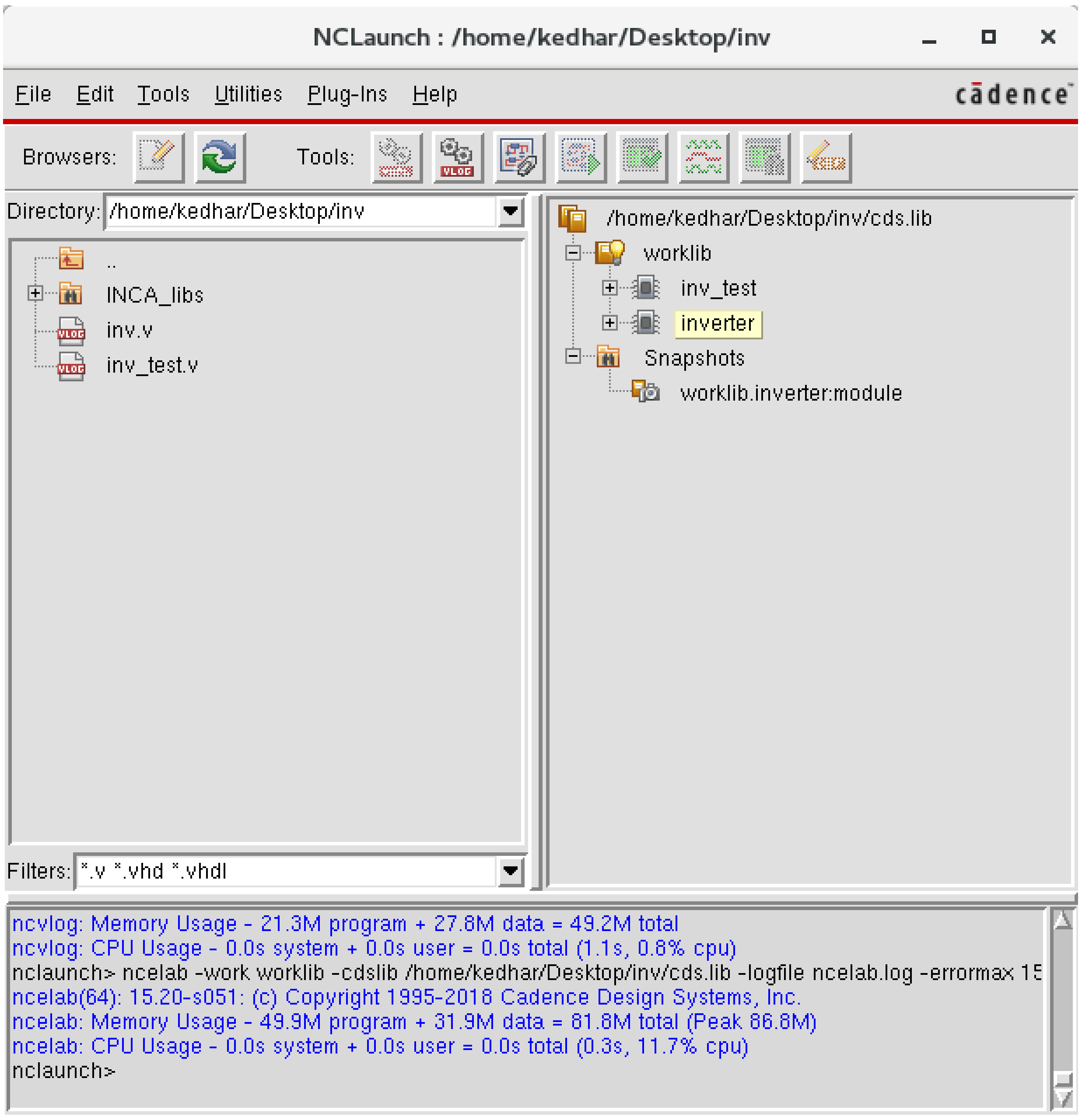
Similarly, select the test module and do the compilation. You can see the worklib being created. Under worklib you can see the module and test-bench as shown in Figure 12. Next is to elaborate the design.


NCLaunch : /home/kedhar/Desktop/inv
File Edit Tools Utilities Plug-Ins Help
cādence
Browsers:
Tools:
Directory: /home/kedhar/Desktop/inv
..
INCA_libs
inv.v
inv_test.v
Filters: *.v *.vhd *.vhdl
/home/kedhar/Desktop/inv/cds.lib
worklib
inv_test
inverter
Snapshots
1 items selected

```
ncvlog: Memory Usage - 21.3M program + 28.1M data = 49.4M total
ncvlog: CPU Usage - 0.0s system + 0.0s user = 0.0s total (0.2s, 5.8% cpu)
nclaunch> ncvlog -work worklib -cdslib /home/kedhar/Desktop/inv/cds.lib -logfile ncvlog.log -errormax 15
ncvlog(64): 15.20-s051: (c) Copyright 1995-2018 Cadence Design Systems, Inc.
ncvlog: Memory Usage - 21.3M program + 27.8M data = 49.2M total
ncvlog: CPU Usage - 0.0s system + 0.0s user = 0.0s total (1.1s, 0.8% cpu)
nclaunch>
```

Figure : 12

Elaboration :

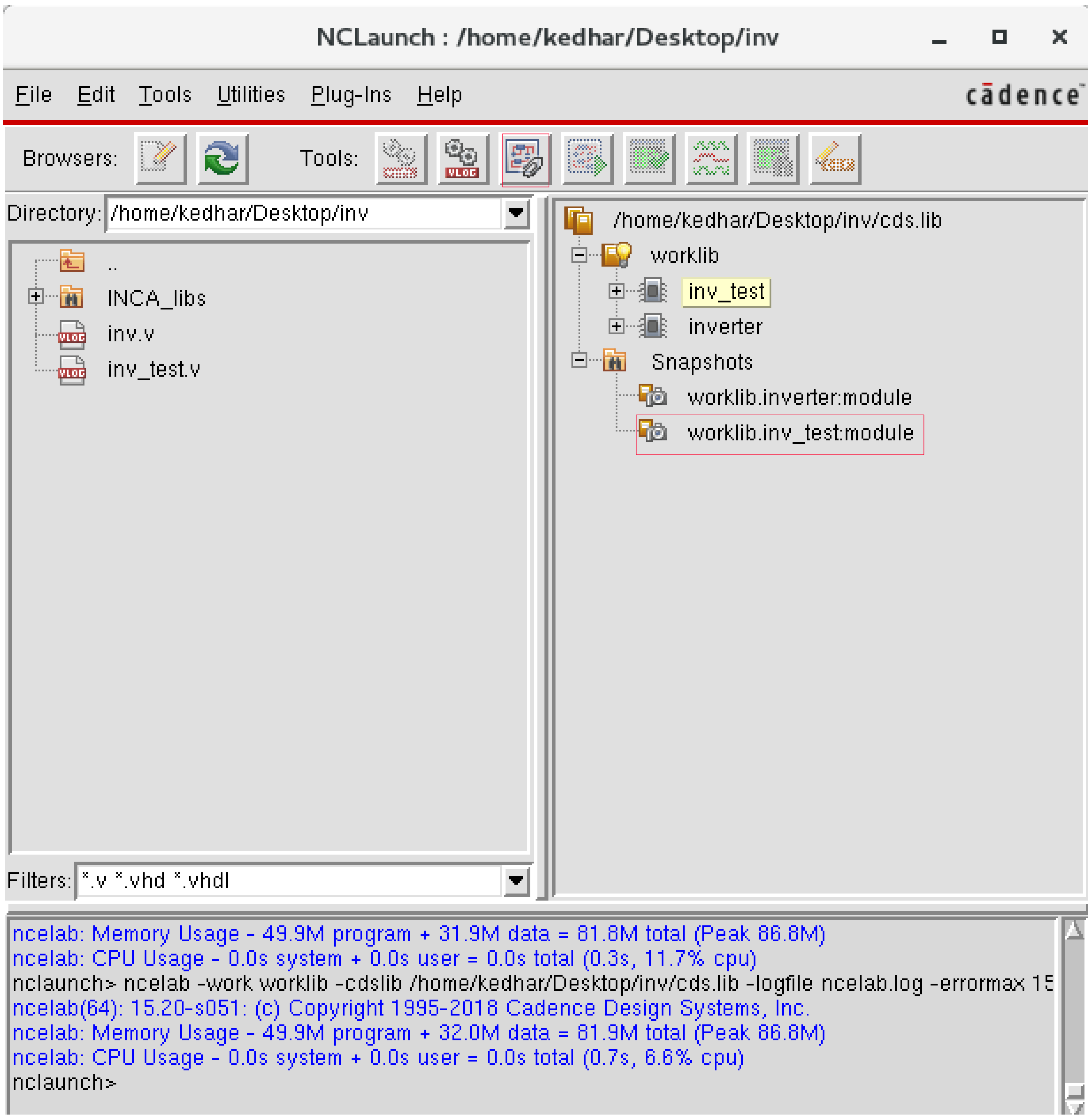

NCLaunch : /home/kedhar/Desktop/inv
File Edit Tools Utilities Plug-Ins Help
cādence
Browsers:
Tools:
Directory: /home/kedhar/Desktop/inv
..
INCA_libs
inv.v
inv_test.v
Filters: *.v *.vhd *.vhdl
/home/kedhar/Desktop/inv/cds.lib
worklib
inv_test
inverter
Snapshots
worklib.inverter:module

```
ncvlog: Memory Usage - 21.3M program + 27.8M data = 49.2M total
ncvlog: CPU Usage - 0.0s system + 0.0s user = 0.0s total (1.1s, 0.8% cpu)
nclaunch> ncelab -work worklib -cdslib /home/kedhar/Desktop/inv/cds.lib -logfile ncelab.log -errormax 15
ncelab(64): 15.20-s051: (c) Copyright 1995-2018 Cadence Design Systems, Inc.
ncelab: Memory Usage - 49.9M program + 31.9M data = 81.8M total (Peak 86.8M)
ncelab: CPU Usage - 0.0s system + 0.0s user = 0.0s total (0.3s, 11.7% cpu)
nclaunch>
```

Figure : 13

Select the file under worklib and in **Tools : launch elaborator with current selection** will get enable. select the **elaborator** to elaborate the design.

Choose the module and test bench and elaborate the design.


NCLaunch : /home/kedhar/Desktop/inv
File Edit Tools Utilities Plug-Ins Help
cādence
Browsers:
Tools:
Directory: /home/kedhar/Desktop/inv
..
INCA_libs
inv.v
inv_test.v
Filters: *.v *.vhd *.vhdl
/home/kedhar/Desktop/inv/cds.lib
worklib
inv_test
inverter
Snapshots
worklib.inverter:module
worklib.inv_test:module

```
ncelab: Memory Usage - 49.9M program + 31.9M data = 81.8M total (Peak 86.8M)
ncelab: CPU Usage - 0.0s system + 0.0s user = 0.0s total (0.3s, 11.7% cpu)
nclaunch> ncelab -work worklib -cdslib /home/kedhar/Desktop/inv/cds.lib -logfile ncelab.log -errormax 15
ncelab(64): 15.20-s051: (c) Copyright 1995-2018 Cadence Design Systems, Inc.
ncelab: Memory Usage - 49.9M program + 32.0M data = 81.9M total (Peak 86.8M)
ncelab: CPU Usage - 0.0s system + 0.0s user = 0.0s total (0.7s, 6.6% cpu)
nclaunch>
```

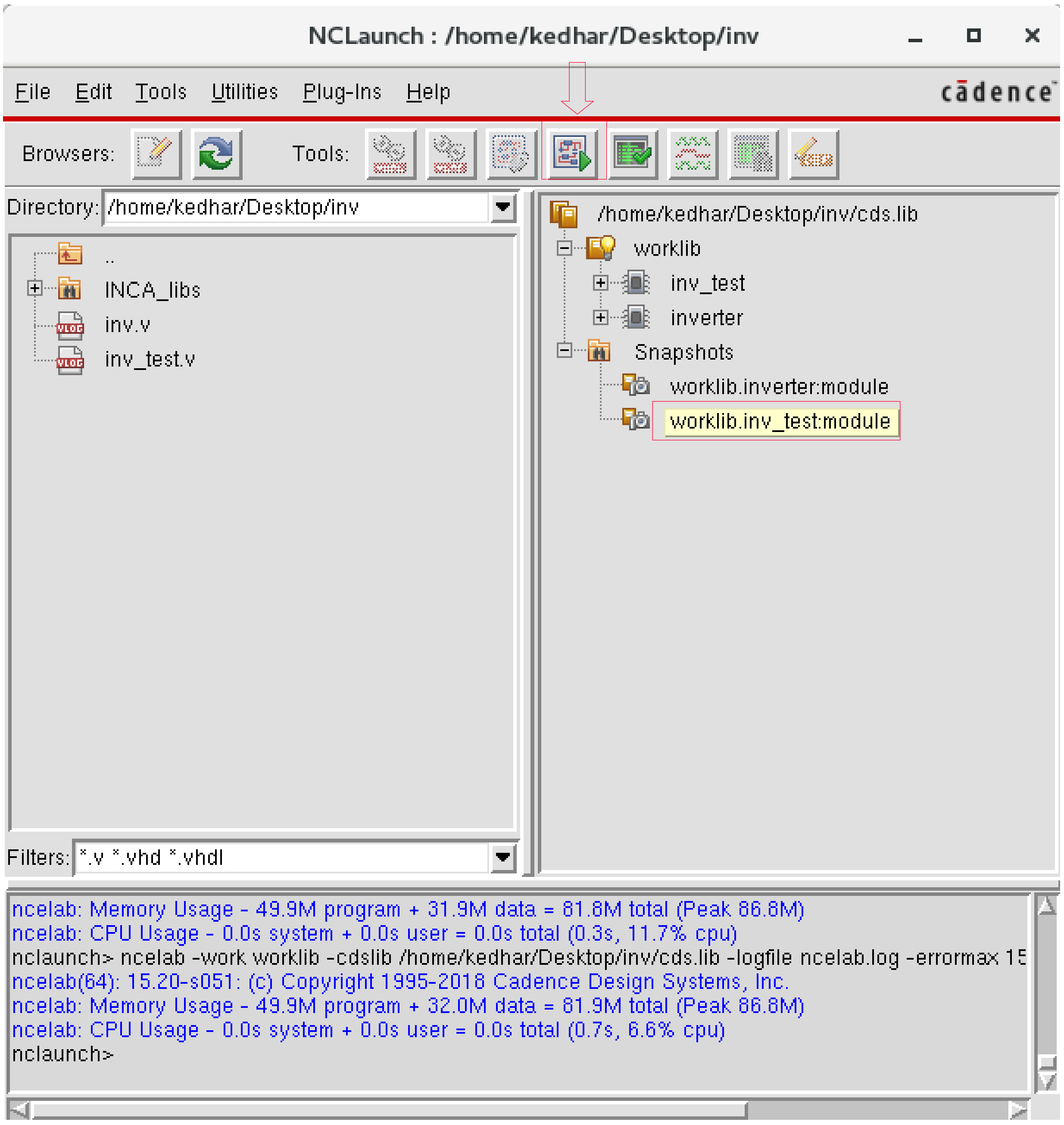
Figure : 14

Similarly, select the test bench module under worklib and do the elaboration. The snapshots will be created.

Next step will be the simulation.

Simulation :

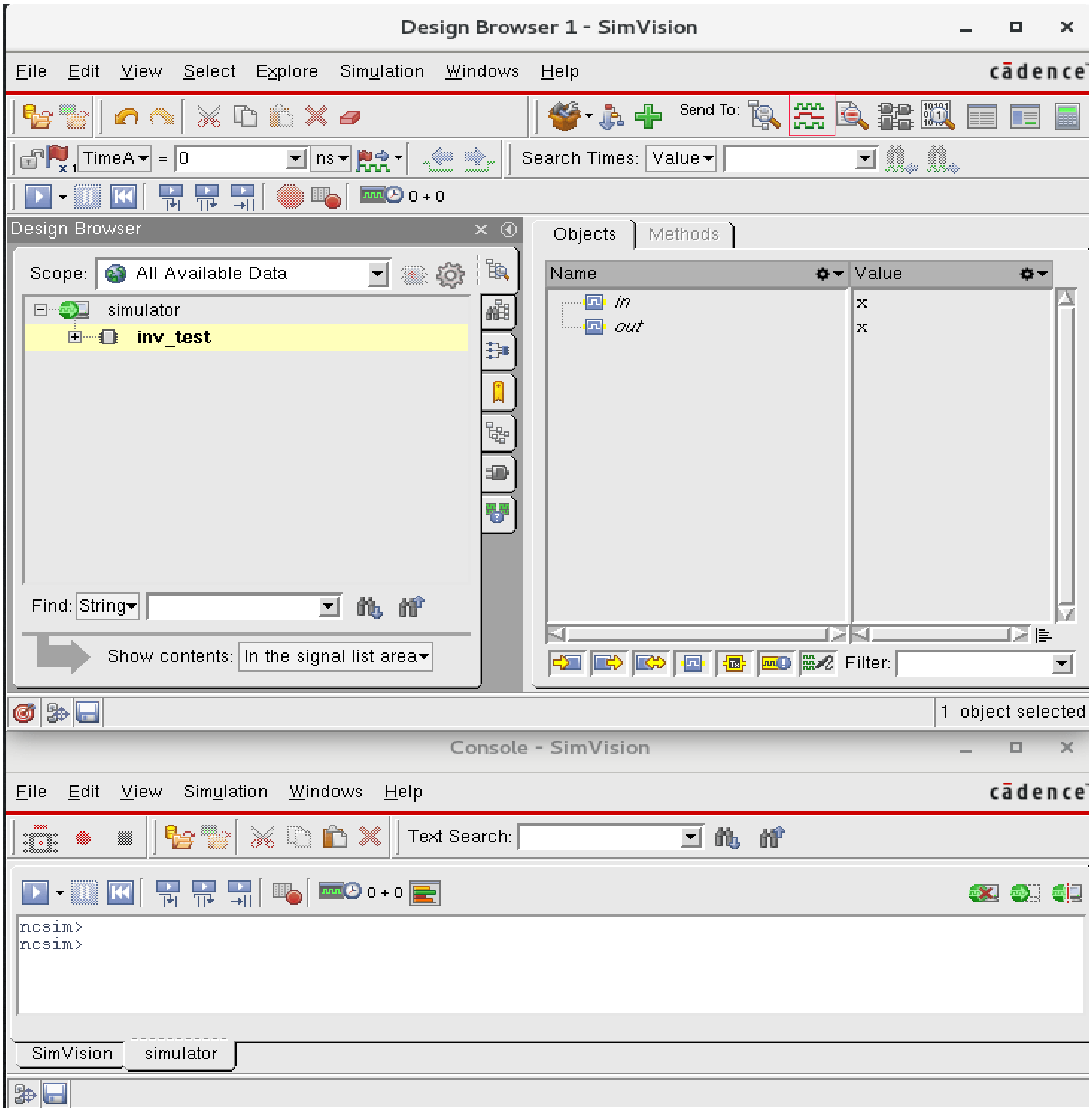
Select the test bench file under snapshot and in Tools : Launch simulator with current selection will get enable as shown in Figure 15 below.


NCLaunch : /home/kedhar/Desktop/inv
File Edit Tools Utilities Plug-Ins Help
cādence
Browsers:
Tools:
Directory: /home/kedhar/Desktop/inv
..
INCA_libs
inv.v
inv_test.v
Filters: *.v *.vhd *.vhdl
/home/kedhar/Desktop/inv/cds.lib
worklib
inv_test
inverter
Snapshots
worklib.inverter:module
worklib.inv_test:module

```
ncelab: Memory Usage - 49.9M program + 31.9M data = 81.8M total (Peak 86.8M)
ncelab: CPU Usage - 0.0s system + 0.0s user = 0.0s total (0.3s, 11.7% cpu)
nclaunch> ncelab -work worklib -cdslib /home/kedhar/Desktop/inv/cds.lib -logfile ncelab.log -errormax 15
ncelab(64): 15.20-s051: (c) Copyright 1995-2018 Cadence Design Systems, Inc.
ncelab: Memory Usage - 49.9M program + 32.0M data = 81.9M total (Peak 86.8M)
ncelab: CPU Usage - 0.0s system + 0.0s user = 0.0s total (0.7s, 6.6% cpu)
nclaunch>
```

Figure : 15

Select the simulator to simulate the design. After simulation you will get the below window as shown in Figure 16.



```
ncsim>
ncsim>
```

Figure : 16

you will get the **Design Browser** and **Simvision**. In design browser you can test bench on the left side window.

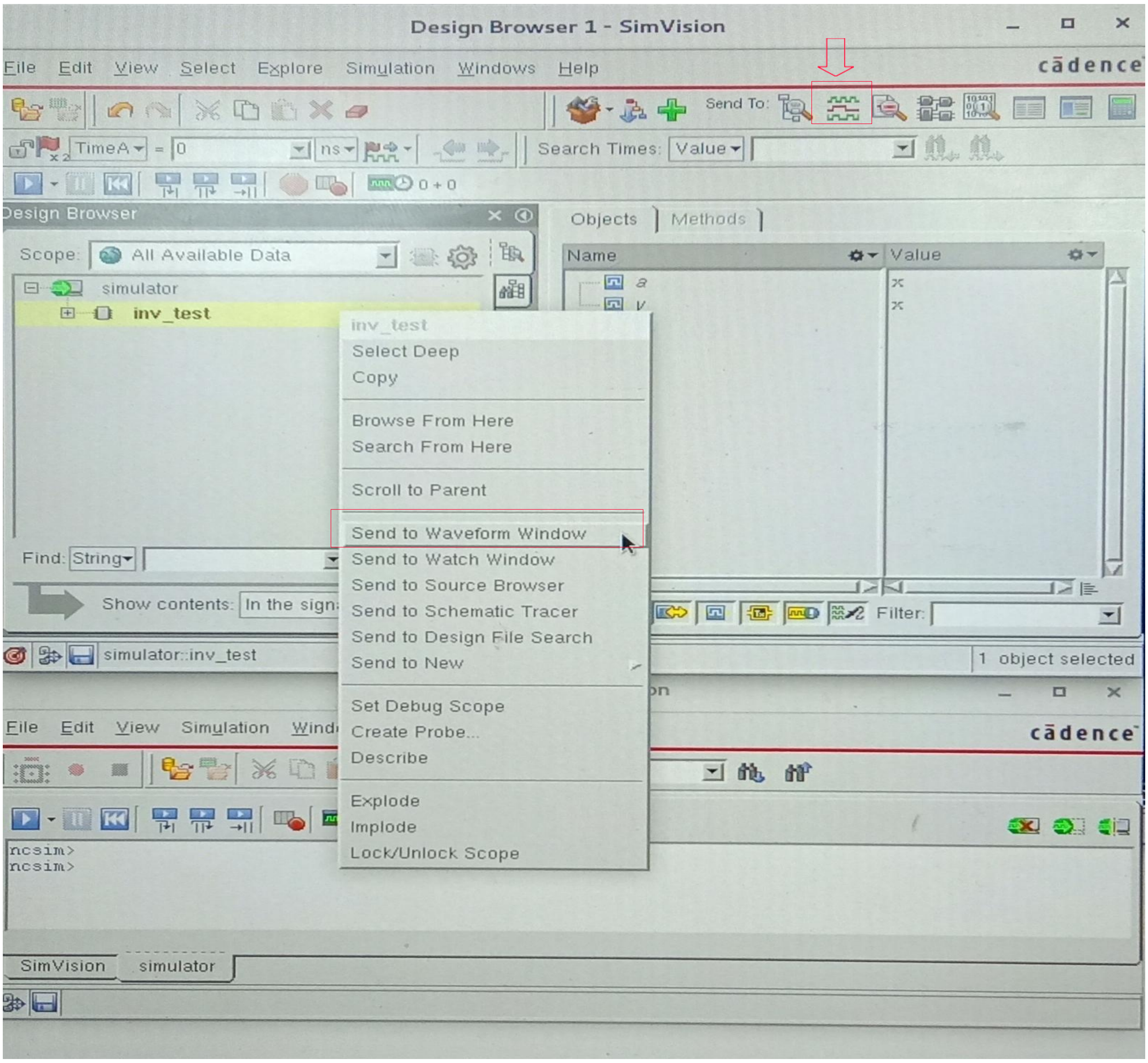


Figure : 17

Select the test bench for the inverter and **Right click** it. Select the **send to waveform window or select the waveform icon** you can see the waveform window as shown in Figure 17.

After that click the **run tool** to see the functional simulation of the inverter as shown in the figure 18.

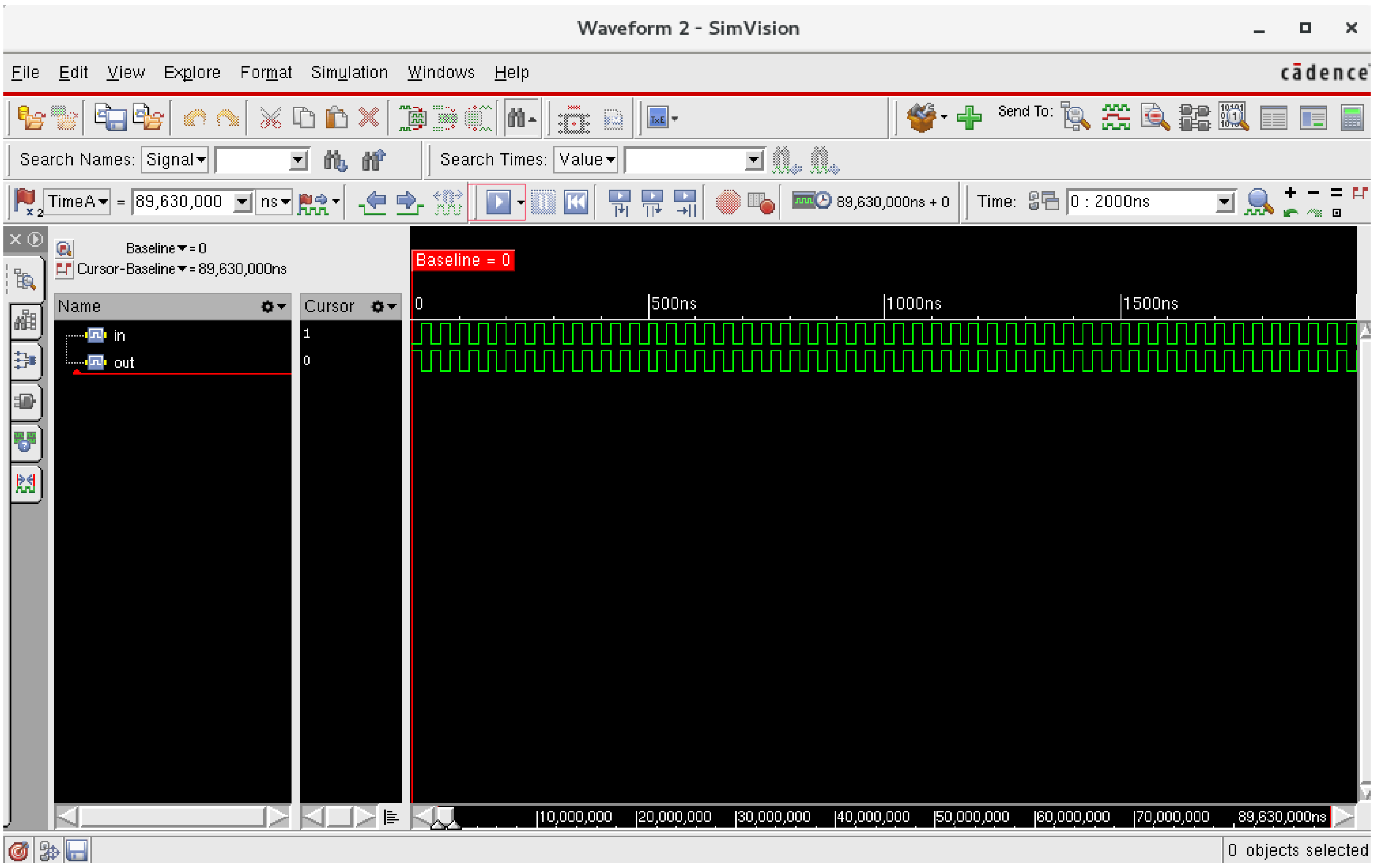


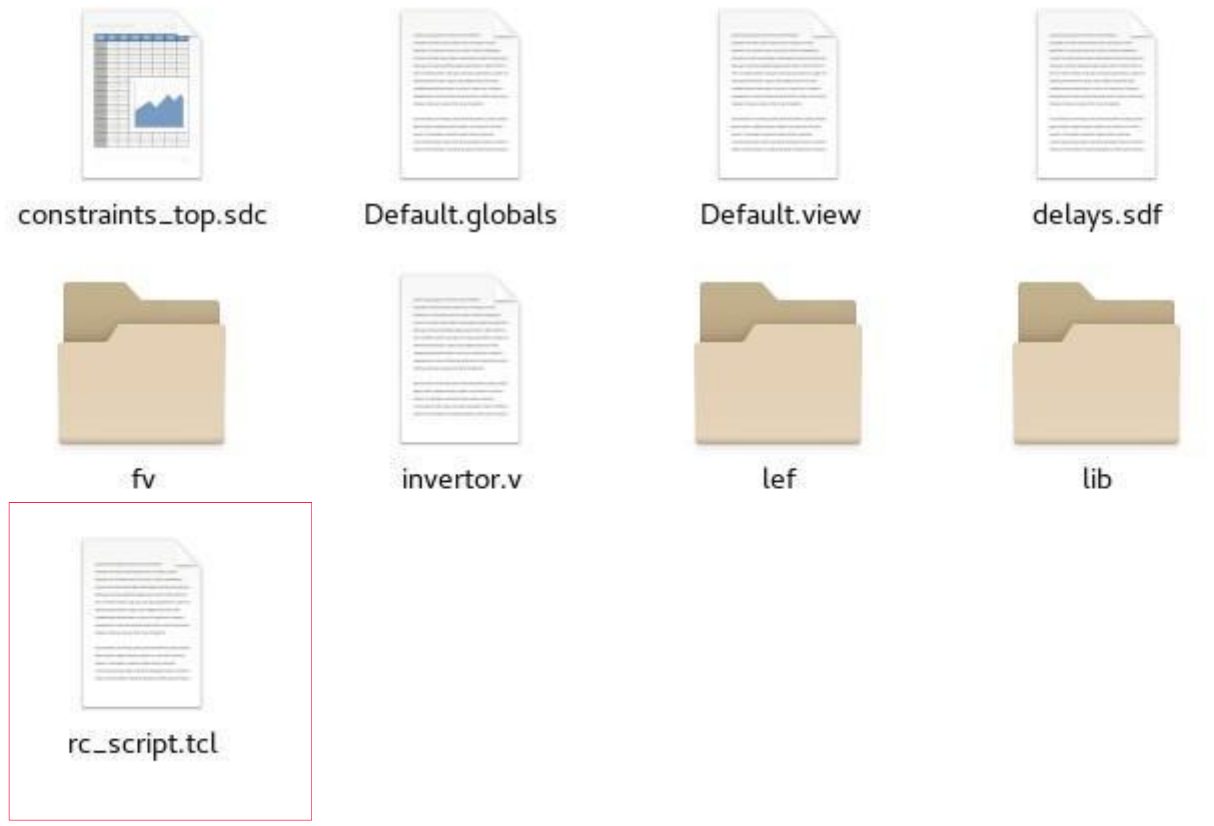
Figure : 18

The equivalent command terminal output can be observed in the Simvision console window and also in nclaunch console terminal.

Synthesis :

Synthesis is the process of converting the RTL Coding into optimized gate level netlist.

The Tool used for doing the synthesis is **GENUS.** The tcl file (**Tool Command Language**) is used for scripting.



Inside the rc_script file we have to mention the commands as shown in Figure 19 below.

Figure : 19

Script file is explained below as shown in Figure 19 :

1. Give the path of the library w.r.t to the directory you are using the command : **set_db lib_search_path**
2. Give the path of the RTL files w.r.t to the directory you are using the command : **set_db hdl_search_path**
3. Read the library file from the directory specified in giving the path for the library files in First line using the command : **set_db library** (slow.lib) is the name of the library file in the directory.
4. Read the RTL files from the directory specified in the second line. The RTL files are in the directory name : **read_hdl inverter.v**
5. Now Elaborate the design using the command : **elaborate**

Constraint File : Not Mandatory

*If you are having constraint file then you can include the constraint file like this

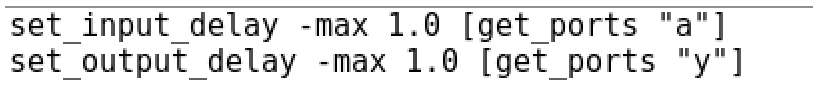
[Give the standard delay constraints using:
read_sdc./constraints_top.sdc]

- Synthesize the circuit using the commands :
  synthesize -to_mapped -effort medium

  - Write the delay file using below commands :
  write_sdf -timescale ns -nonegchecks -recrem split -edges check_edge > delays.sdf

  - Used to mention the time unit : **timescale**
  - Used to ignore the negative timing checks : **nonegchecks**
  - Used to split out the recrem(recovery-removal) timing check to separate checks for recovery and removal : **recrem**
  - Specifies the edges values : **edges**
  - Keeps edge specifiers on timing check arcs but does not add edge specifiers on combinational arcs : **check_edge**
  - Timing could be check using : **report timing.**
  - Similarly for Gates : **report gates.**
  - Check area using : **report area.**
  - Check Power dissipation using : **report power.**
  - It will generate the reports
  - Write the hdl code in terms of library components for the synthesized circuit using the command: **write_hdl > inverter_netlist.v**
  - Similarly write the constraint file using : **write_sdc > inverter_const.sdc.**

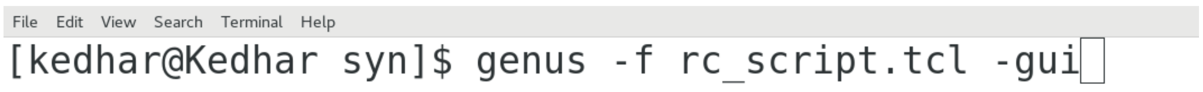
Constraints file :



```
set_input_delay -max 1.0 [get_ports "a"]
set_output_delay -max 1.0 [get_ports "y"]
```

Invoke the **Genus** tool by typing the below command on your terminal window. The below Figure 20 can be represent after typing the command.

genus -f rc_script.tcl -gui

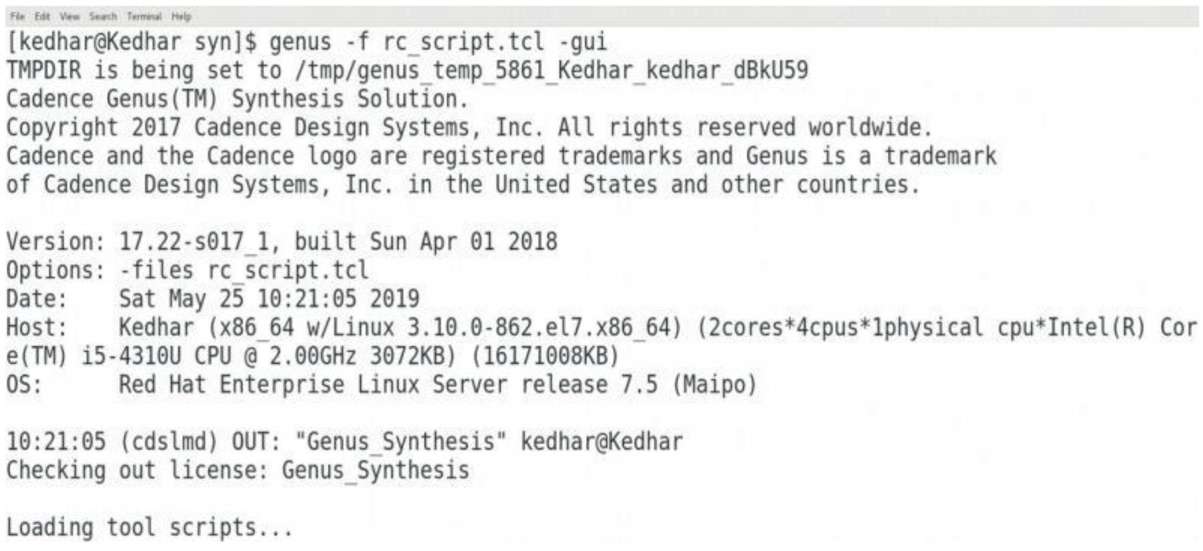


```
[kedhar@Kedhar syn]$ genus -f rc_script.tcl -gui
```

Figure : 20

Type the command in the terminal which is mentioned above as shown in Figure 20.

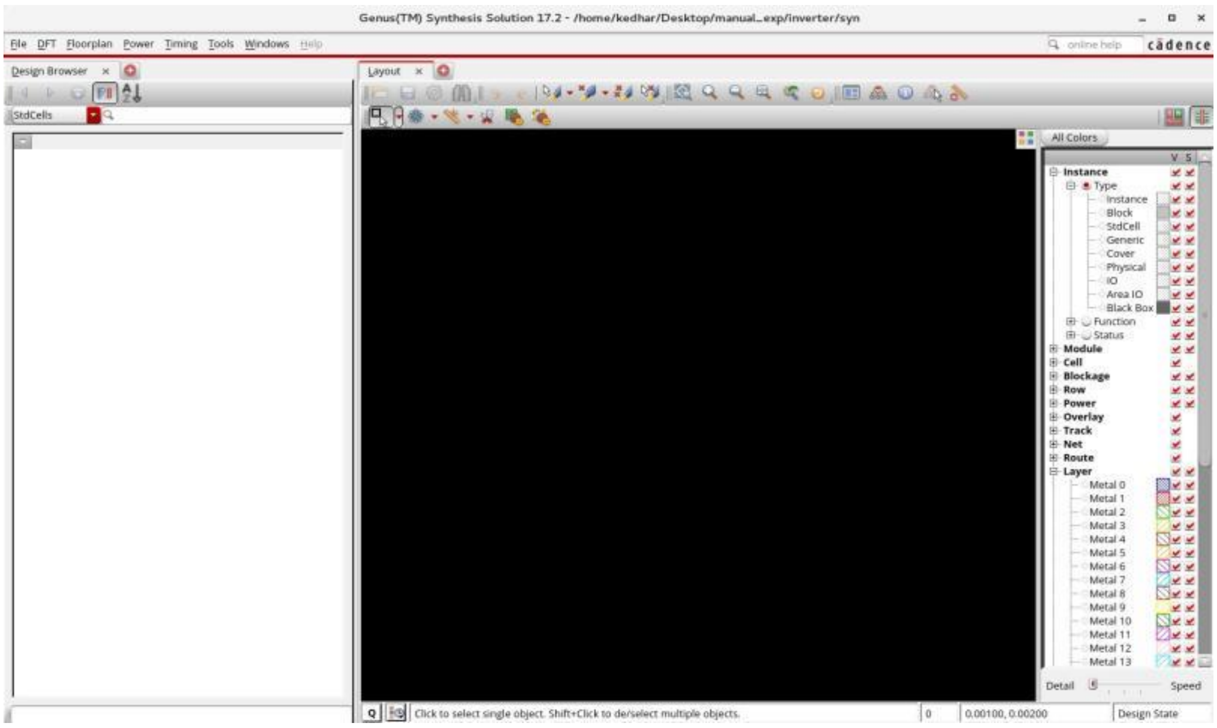
Click on **Enter** after typing the command as shown in Figure 21.



```
[kedhar@Kedhar syn]$ genus -f rc_script.tcl -gui
TMPDIR is being set to /tmp/genus_temp_5861_Kedhar_kedhar_dBkU59
Cadence Genus(TM) Synthesis Solution.
Copyright 2017 Cadence Design Systems, Inc. All rights reserved worldwide.
Cadence and the Cadence logo are registered trademarks and Genus is a trademark
of Cadence Design Systems, Inc. in the United States and other countries.

Version: 17.22-s017_1, built Sun Apr 01 2018
Options: -files rc_script.tcl
Date:    Sat May 25 10:21:05 2019
Host:    Kedhar (x86_64 w/Linux 3.10.0-862.el7.x86_64) (2cores*4cpus*1physical cpu*Intel(R) Cor
e(TM) i5-4310U CPU @ 2.00GHz 3072KB) (16171008KB)
OS:      Red Hat Enterprise Linux Server release 7.5 (Maipo)

10:21:05 (cdslmd) OUT: "Genus_Synthesis" kedhar@Kedhar
Checking out license: Genus_Synthesis

Loading tool scripts...
```

Figure : 21

The genus gui window will be shown as in Figure 22.

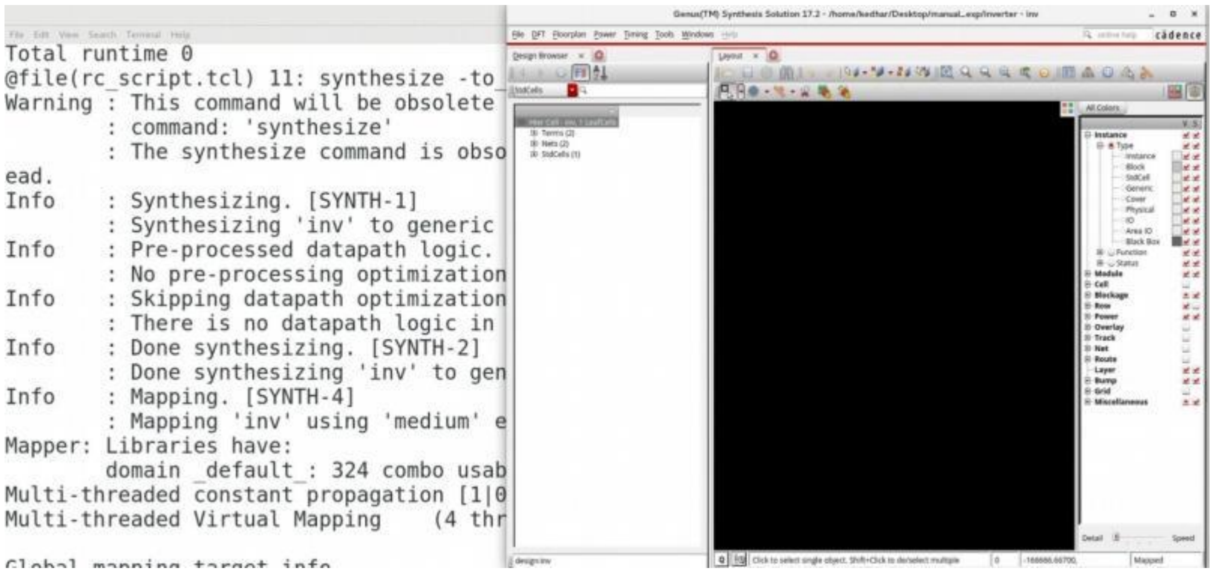


Click to select single object. Shift+Click to de/select multiple objects.

0.00100, 0.00200

Figure : 22

After running the script, you can see the log report in the terminal and the module in the gui window as shown in Figure 23.



```
Total runtime 0
@file(rc_script.tcl) 11: synthesize -to_
Warning : This command will be obsolete
        : command: 'synthesize'
        : The synthesize command is obso
ead.
Info    : Synthesizing. [SYNTH-1]
        : Synthesizing 'inv' to generic
Info    : Pre-processed datapath logic.
        : No pre-processing optimization
Info    : Skipping datapath optimization
        : There is no datapath logic in
Info    : Done synthesizing. [SYNTH-2]
        : Done synthesizing 'inv' to gen
Info    : Mapping. [SYNTH-4]
        : Mapping 'inv' using 'medium' e
Mapper: Libraries have:
        domain _default_: 324 combo usab
Multi-threaded constant propagation [1|0
Multi-threaded Virtual Mapping    (4 thr

Global mapping target info
```

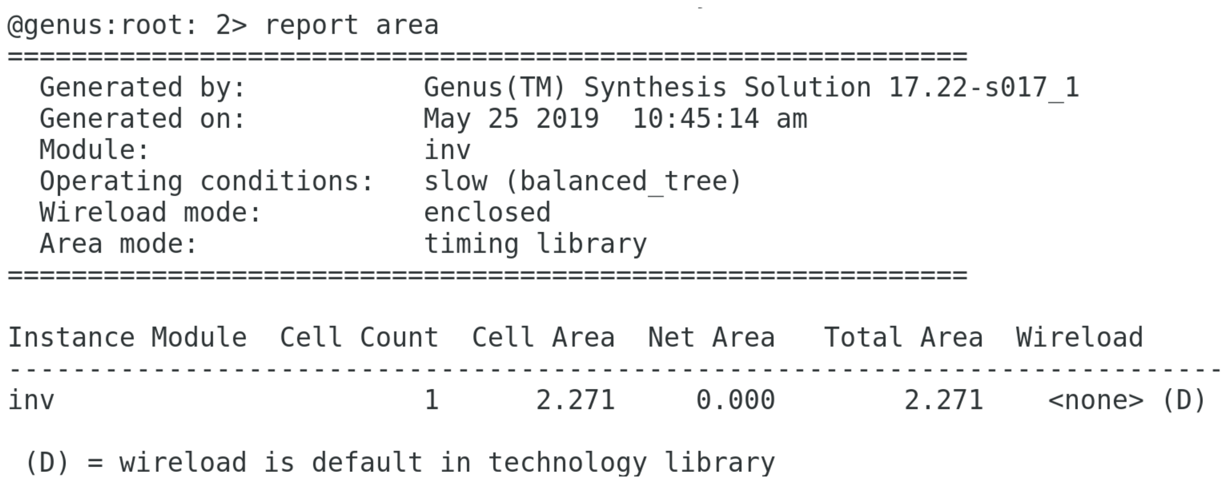
Click to select single object. Shift+Click to de/select multiple

Figure : 23

It will generate the Timing, Area, Power and Gates of the design.

Area Report :

The area report as shown in Figure 24.

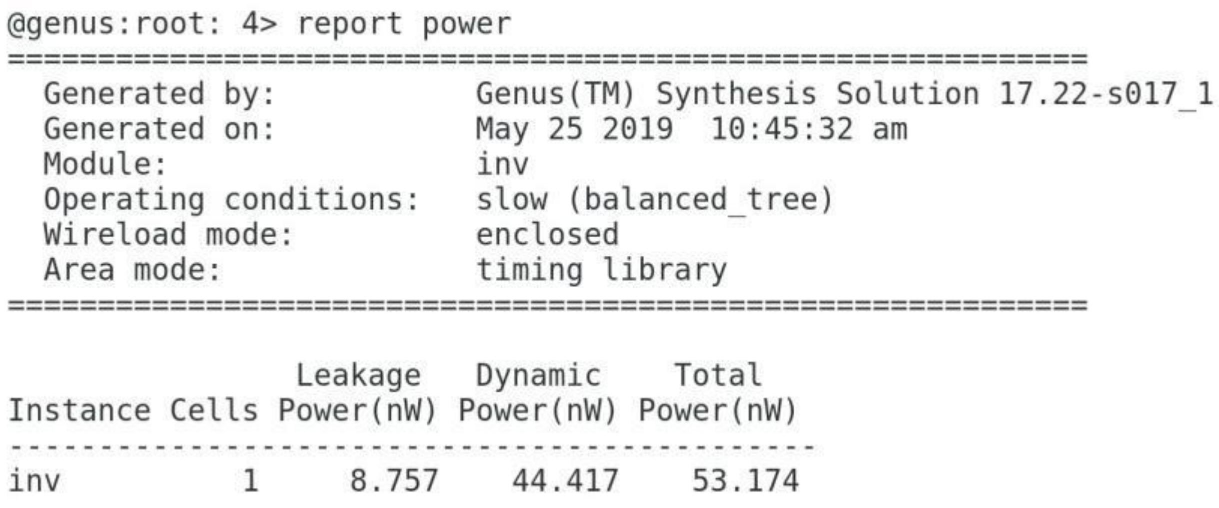


```
@genus:root: 2> report area
============================================================
  Generated by:           Genus(TM) Synthesis Solution 17.22-s017_1
  Generated on:           May 25 2019  10:45:14 am
  Module:                 inv
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
============================================================

Instance Module  Cell Count  Cell Area  Net Area   Total Area  Wireload
----------------------------------------------------------------------------
inv                    1      2.271     0.000        2.271    <none> (D)

 (D) = wireload is default in technology library
```

Figure : 24

Power Report :

The power report as shown in Figure 25.

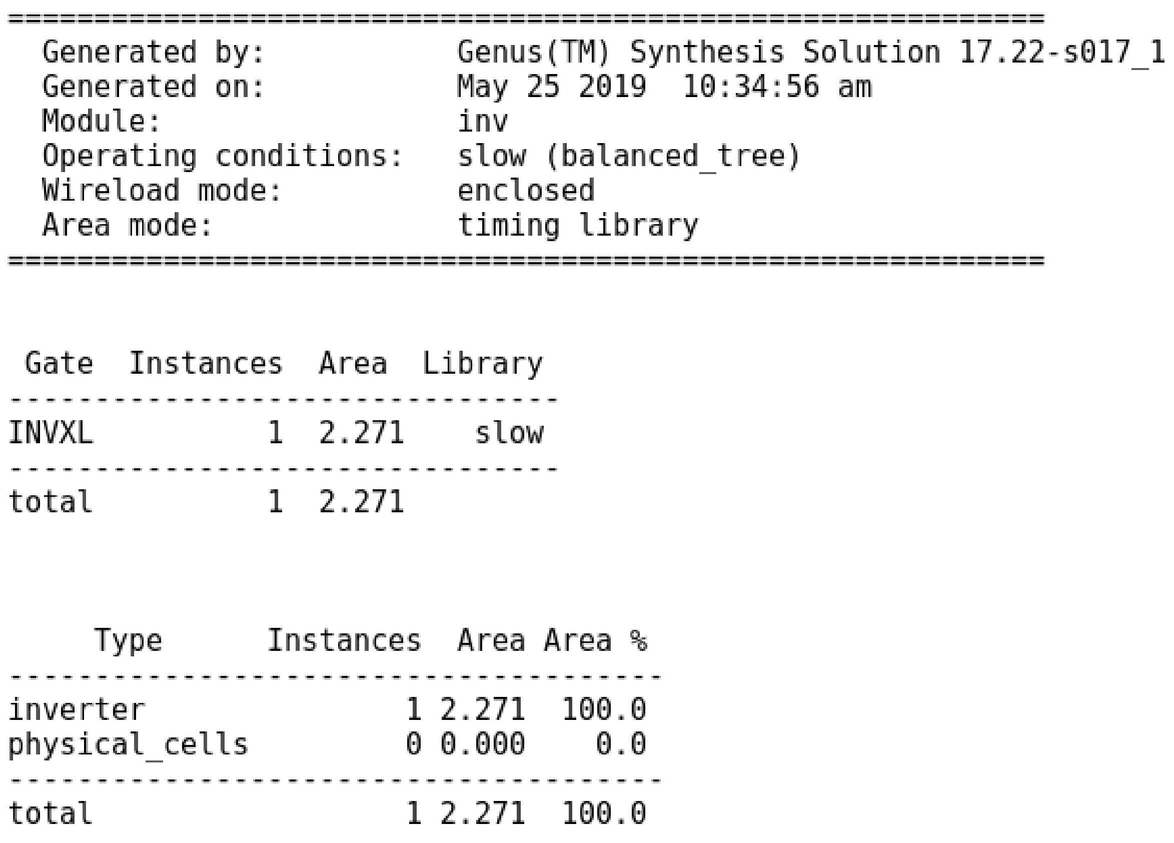


```
@genus:root: 4> report power
============================================================
  Generated by:           Genus(TM) Synthesis Solution 17.22-s017_1
  Generated on:           May 25 2019  10:45:32 am
  Module:                 inv
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
============================================================

                  Leakage   Dynamic    Total
Instance Cells Power(nW) Power(nW) Power(nW)
-------------------------------------------
inv         1     8.757    44.417    53.174
```

Figure : 25

Gate Report :

The number of gates can be displayed in the below report as shown in Figure 26.



```
============================================================
  Generated by:           Genus(TM) Synthesis Solution 17.22-s017_1
  Generated on:           May 25 2019  10:34:56 am
  Module:                 inv
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
============================================================


 Gate  Instances  Area  Library
--------------------------------
INVXL          1  2.271    slow
--------------------------------
total          1  2.271


     Type      Instances  Area Area %
--------------------------------------
inverter               1 2.271  100.0
physical_cells         0 0.000    0.0
--------------------------------------
total                  1 2.271  100.0
```

Figure : 26

GUI :

Double Click file on the side of the window and see the RTL design.

The design after doing the synthesis will be shown in Figure 27 below.

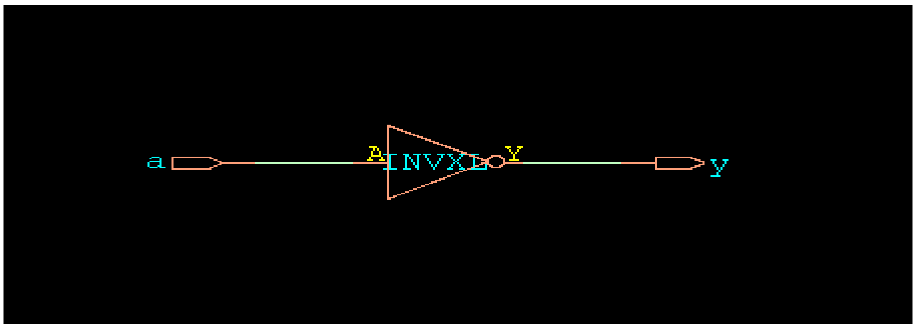


Figure : 27

Generated Files :

The files generated after doing the synthesis is shown in Figure 28 with the highlighted arrows.

Figure : 28

LAB 2 : BUFFER

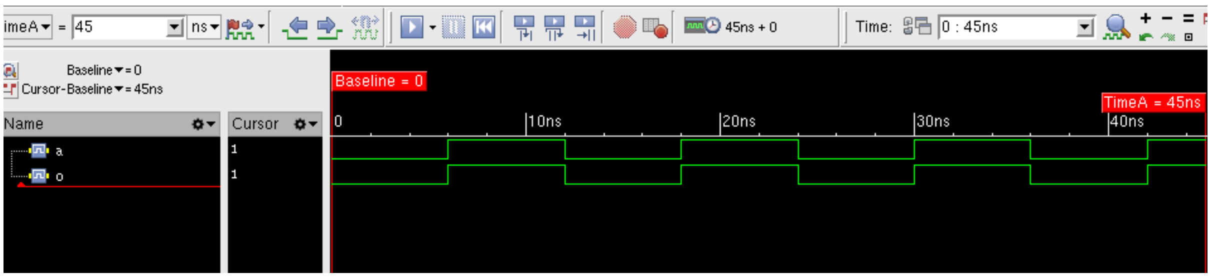
Top Module :

```
module buffer(o,a);
    output o;
    input a;
    assign o=a;
endmodule
```

Test Module :

```
module tb_buffer;

    reg a;
    wire o;

    buffer test (.a(a) ,.o(o));

    //Above style is connecting by names

    initial begin
        a =1'b0;

        #45 $finish;
    end

    always #6 a =~a;



    always @(o)
    $display( "time =%0t \tINPUT VALUES: \t a=%b \toutput value o
=%b",$time,a,o);
endmodule
```

RESULT :



Constraint file :

set_input_delay -max 1.0 [get_ports “a”]
set_output_delay -max 1.0 [get_ports “y”]

Design :

The design after doing the synthesis as shown in figure 29.

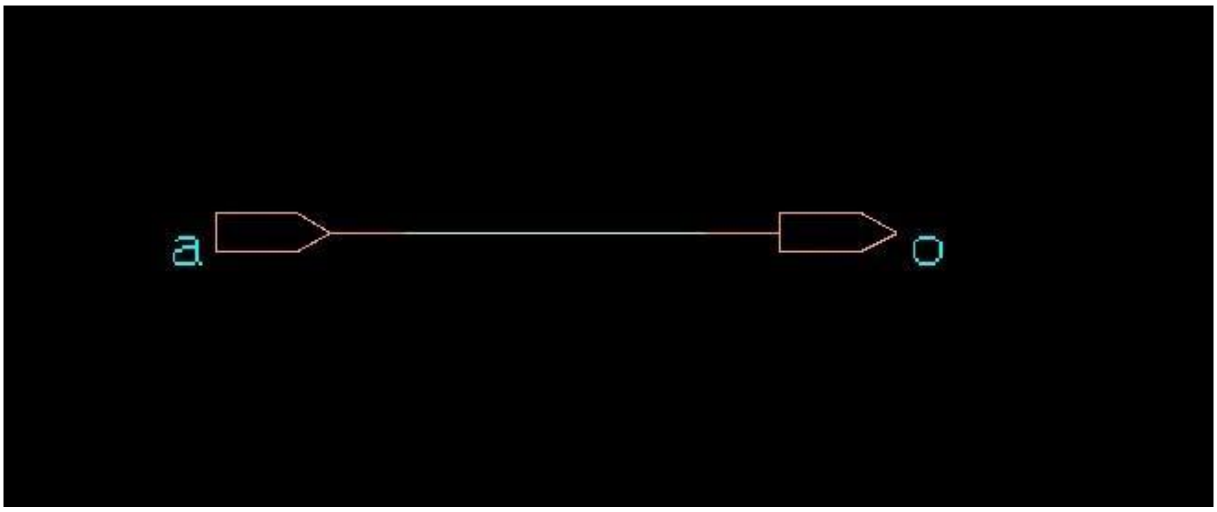


Figure : 29

LAB 3 : Transmission Gate

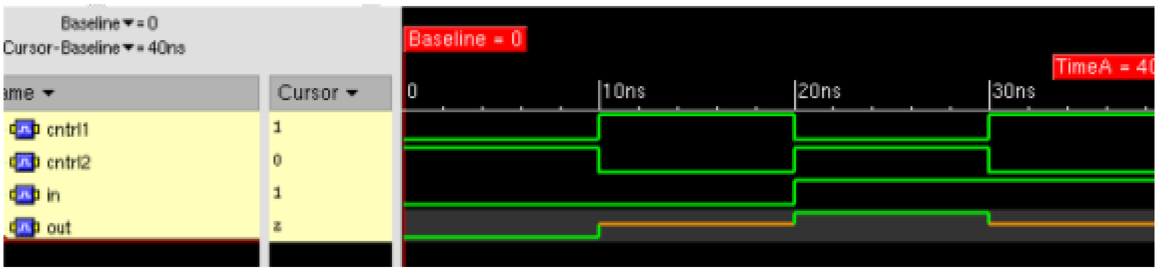
Top module :

```
module my_tran(A_in, A_out, B_in, B_out, control); input
        A_in, B_in, control;
        output A_out, B_out;

        assign A_out = A_in;
        assign A_out = control ? B_in : 1'bz;

        assign B_out = B_in;
        assign B_out = control ? A_in : 1'bz;

      endmodule
```

Test Module :

```
module trangate_test;
wire out ;
reg in ;
reg cntrl1,cntrl2;

trangate t1 ( out, in, cntrl1, cntrl2 ) ; task
display ;
begin
$display ( "time=%0d" , $time , " ns", " Input=" , in, " Output=", out, "
Control1=",cntrl1, " Control2=",cntrl2 ) ;
end
endtask
initial
begin
in = 1'b0 ; cntrl1 = 1'b0 ; cntrl2 = 1'b1 ;
in = 1'b0 ; cntrl1 = 1'b1 ; cntrl2 = 1'b0 ; in =
1'b1 ; cntrl1 = 1'b0 ; cntrl2 = 1'b1 ; in = 1'b1
; cntrl1 = 1'b1 ; cntrl2 = 1'b0 ; end
endmodule
```

RESULT :



LAB 4 : Basic / Universal Gates

4a.AND gate :

Top Module :

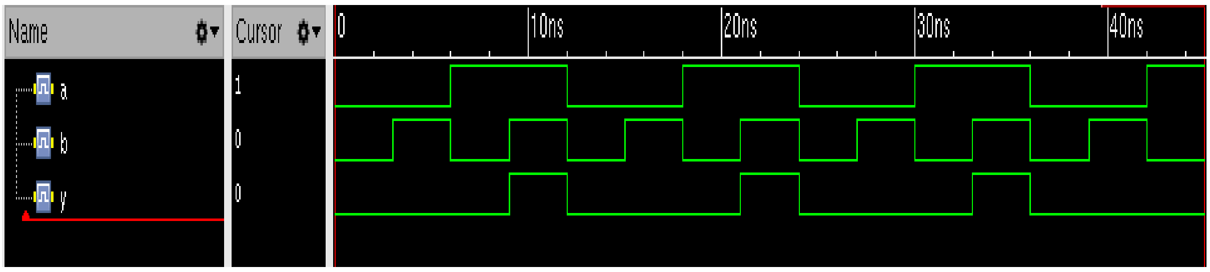
```
module andgate (a, b, y);
input a, b;
output y;
assign y = a & b;
endmodule
```

Test Module :

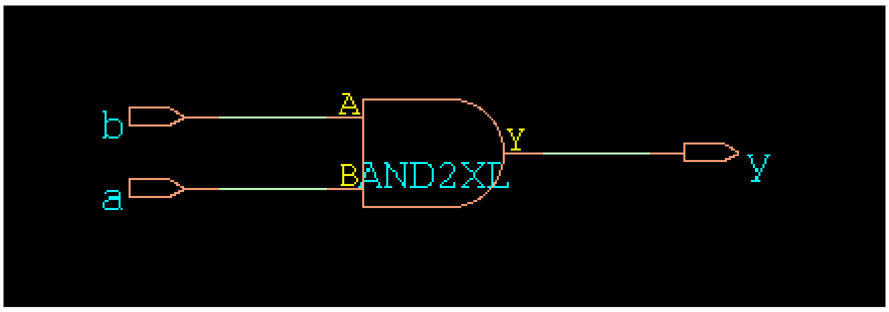
```
module tb_and_gate;

    reg a,b;
    wire y;

    andgate test (.a(a) ,.b(b),.y(y));
```

```
//Above style is connecting by names

    initial begin a
        =1'b0; b =
        1'b0;
        #45 $finish;
    end

    always #6 a =~a;
    always #3 b =~b;

    always @(y)
    $display( "time =%0t \tINPUT VALUES: \t a=%b b =%b \t output value y
=%b",$time,a,b,y);
endmodule
```

RESULT :



SCHEMATIC :



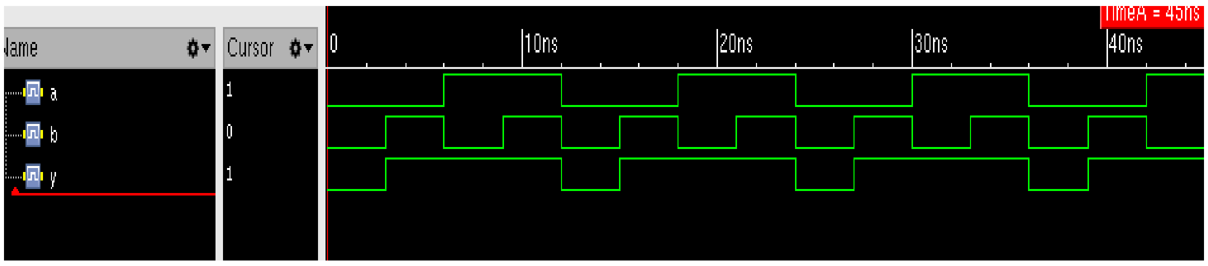
4b. OR gate :
Top Module :

```
module orgate (a, b, y);
input a, b;
output y; assign y
= a | b;
endmodule
```

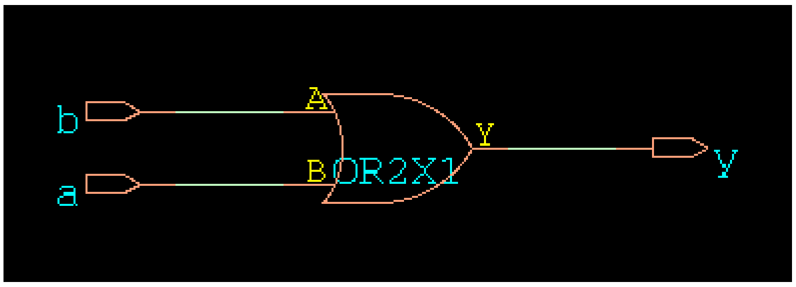
Test Module :

```
module tb_or_gate;

   reg a,b;
   wire y;

   orgate test (.a(a) ,.b(b),.y(y));

   //Above style is connecting by names

   initial begin
      a =1'b0; b
      = 1'b0;
      #45 $finish;
   end

   always #6 a =~a;
   always #3 b =~b;

   always @(y)
   $display( "time =%0t \tINPUT VALUES: \t a=%b b =%b \t output value y
=%b",$time,a,b,y);
endmodule
```

RESULT :



SCHEMATIC :



4c.NAND gate :

Top Module :

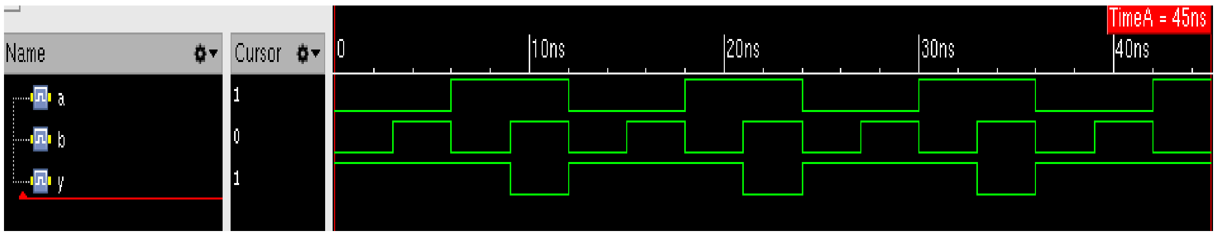
```
module nandgate (a, b, y);
input a, b;
output y;
assign y = ~(a & b);
endmodule
```

Test Module :

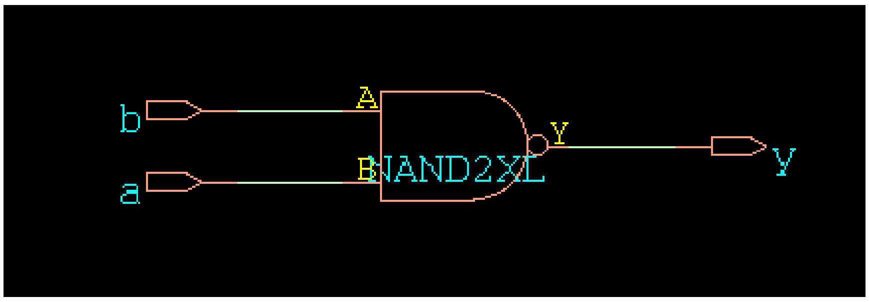
```
module tb_nand_gate;

   reg a,b;
   wire y;

   nandgate test (.a(a) ,.b(b),.y(y));
```

```
//Above style is connecting by names

   initial begin a
      =1'b0; b =
      1'b0;
      #45 $finish;
   end

   always #6 a =~a;
   always #3 b =~b;

   always @(y)
   $display( "time =%0t \tINPUT VALUES: \t a=%b b =%b \t output value y
=%b",$time,a,b,y);
endmodule
```

RESULT :



SCHEMATIC :



4d.NOR gate :

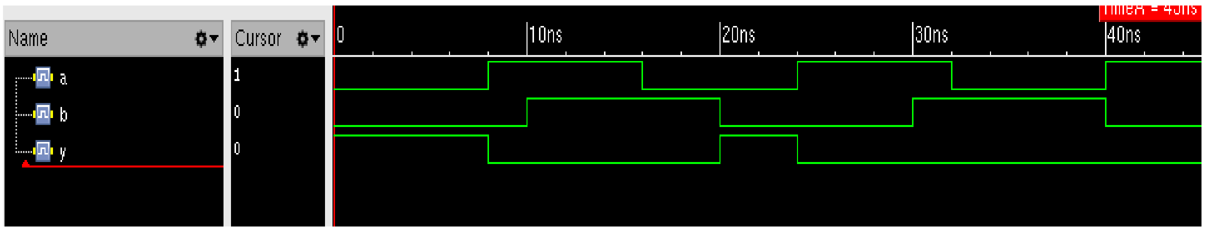
Top Module :

```
module norgate (a, b, y);
input a, b;
output y;
assign y = ~(a | b);
endmodule
```

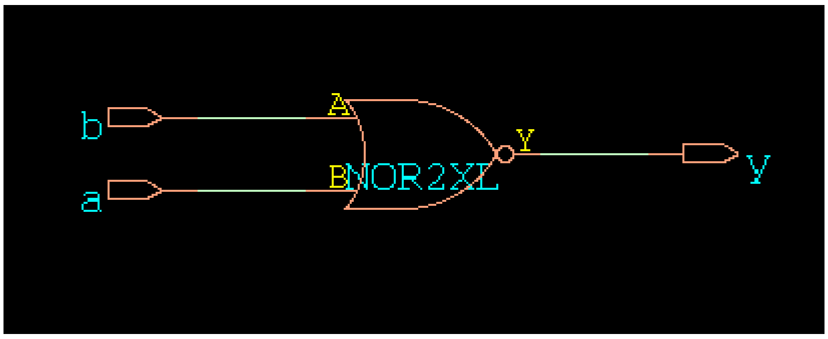
Test Module :

```
module tb_nor_gate;

   reg a,b;
   wire y;

   norgate test (.a(a) ,.b(b),.y(y));

   //Above style is connecting by names

   initial begin
      a =1'b0; b
      = 1'b0;
      #45 $finish;
   end

   always #8 a =~a;
   always #10 b =~b;

   always @(y)
   $display( "time =%0t \tINPUT VALUES: \t a=%b b =%b \t output value y
=%b",$time,a,b,y);
endmodule
```

RESULT :



SHCEMATIC :



4e. XOR gate :

Top module :

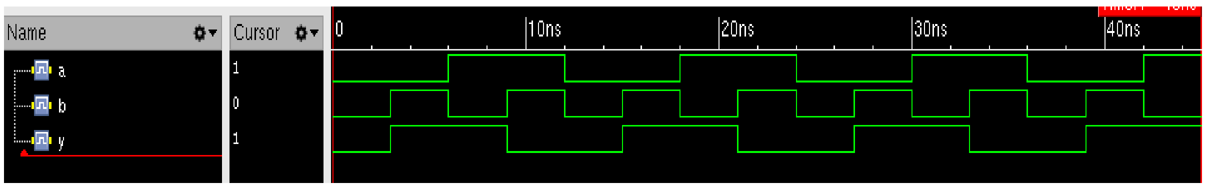
```
module xorgate (a, b, y);
input a, b;
output y;
assign y = a ^ b;
endmodule
```

Test module :

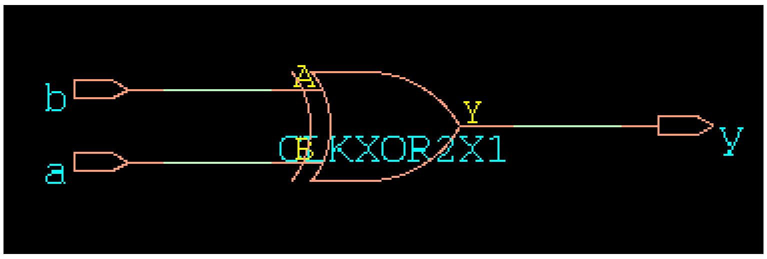
```
module tb_xor_gate;

    reg a,b;
    wire y;
```

```
xorgate test (.a(a) ,,.b(b),.y(y));

    //Above style is connecting by names

    initial begin
        a =1'b0; b
        = 1'b0;
        #45 $finish;
    end

    always #6 a =~a;
    always #3 b =~b;

    always @(y)
    $display( "time =%0t \tINPUT VALUES: \t a=%b b =%b \t output value y
=%b",$time,a,b,y);
endmodule
```

RESULT :



SCHEMATIC :



4f. XNOR gate :

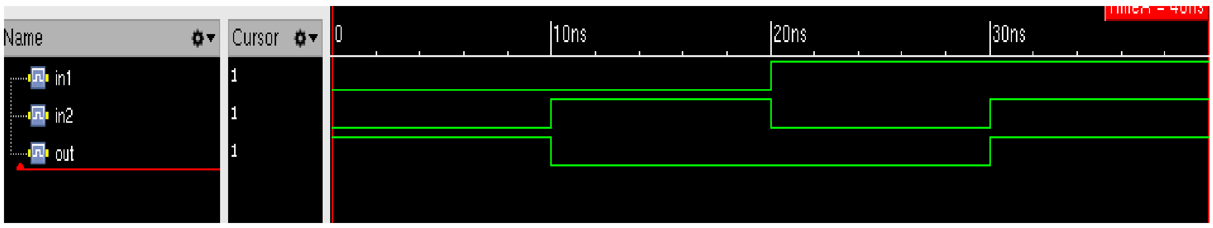
Top module :

```
//Define our own XNOR Gate, module
xnorgate ( out , in1 , in2 );

// Declarations of I/O ports
output out;
input in1,in2;
wire in2bar;
assign in2bar = ~in2;

// Instantiate pmos and nmos switches :
pmos (out,in2bar,in1);
nmos (out,in2,in1); pmos
(out,in1,in2bar); nmos
(out,in1,in2); endmodule
```

Test module :

```
module xnor_test;
wire out ;
reg in1,in2 ;
`uselib view = vlog
// Instantiate Xnor gate Module
xnorgate x1 ( out, in1, in2 ) ;
`nouselib
// Display task
display ; begin
$display ( "time=%0d" , $time , " ns"," Input1=" , in1 ," Input2=" , in2 ,
" Output=" , out ) ;
end
endtask
// Apply Stimulus initial
begin
in1 = 1'b0 ; in2 = 1'b0 ; #10 ; display ; in1 =
1'b0 ; in2 = 1'b1 ; #10 ; display ; in1 = 1'b1 ;
in2 = 1'b0 ; #10 ; display ; in1 = 1'b1 ; in2 =
1'b1 ; #10 ; display ; end
endmodule
```

RESULT :



ALTERNATIVE CODE :

```
module gates(
input a,b;
output not_out, and_out, or_out, nand_out, nor_out, xor_out,
xnor_out);
assign not_out=!a;
assign and_out=a&b;
assign or_out=a|b;
assign nand_out=~(a&b);
assign nor_out=~(a|b);
assign xor_out=a^b;
assign xnor_out=~(a^b);
endmodule
```

LAB 5 : FLIP FLOPS

5a. D-Flip Flop :

Top Module :

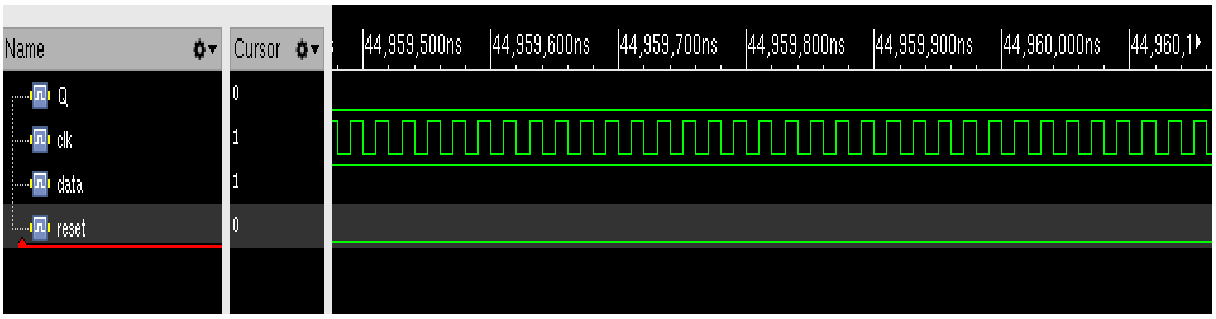
```
module dff_sync_reset (data,clk,reset,q);
input data, clk, reset ;
output q;
reg q;
always @ ( posedge clk) if
(~reset) begin
q <= 1'b0;
end else begin q
<= data;
end
endmodule
```

Test Module :

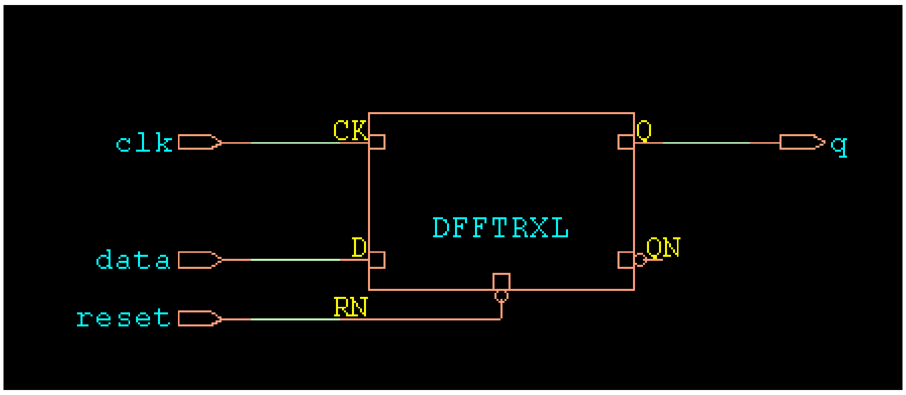
```
module tb_DFF();
reg data;
reg clk;
reg reset;
wire Q;

dff_sync_reset dut(data,clk,reset,Q);

initial begin
  clk=0;
     forever #10 clk = ~clk;
```

```
end
initial begin
 reset=1;
 data <= 0;
 #100;
 reset=0;
 data <= 1;
 #100;
 data <= 0;
 #100;
 data <= 1;
end
endmodule
```

RESULT :



SCHEMATIC :



5b. JK-FLIP FLOP :

Top Module :

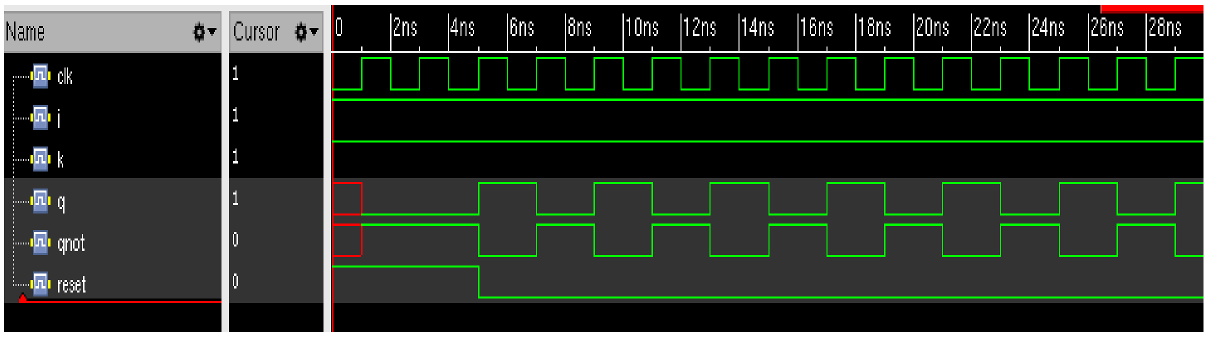
```
module jkff(input reset, input clk, input j, input k, output reg q, output qnot);
assign qnot=~q; always
@(posedge clk)
  if (reset) q<=1'b0; else
  case ({j, k})
 2'b00: q<=q;
 2'b01: q<=1'b0;
 2'b10: q<=1'b1;
 2'b11: q<=~q;
endcase
endmodule
```

Test Module :

```
module test;

reg clk=0;
reg j=0; reg
k=0;
reg reset=1;
wire q, qnot;

  jkff dut(reset, clk,j,k,q,qnot);

initial
  begin
    $dumpfile("dump.vcd");
    $dumpvars(1);
  j=1'b1; // set your JK here

  k=1'b1;
#5  reset=1'b0;
#25 $finish;
  end

always #1 clk=~clk;

endmodule
```

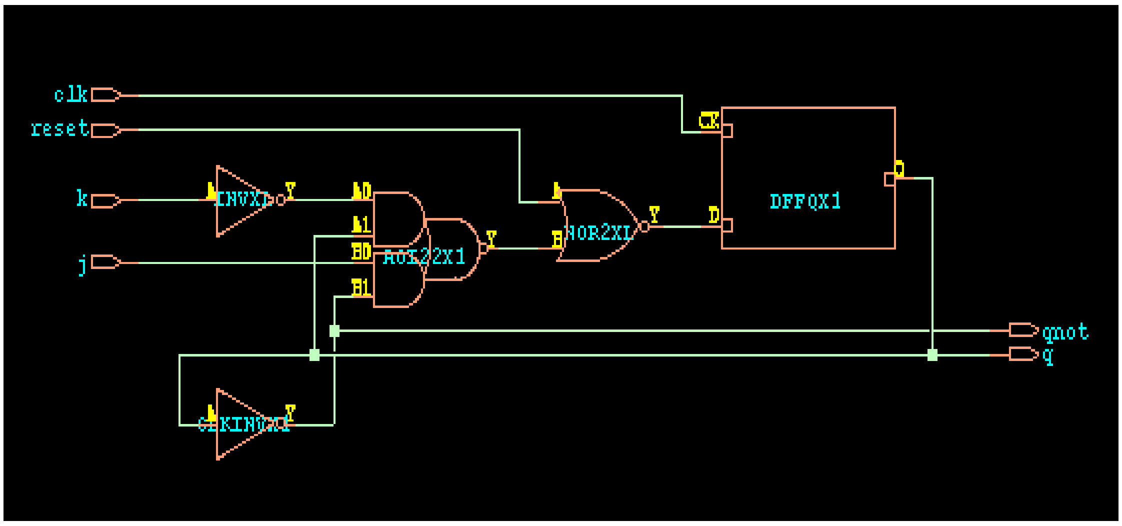
RESULT

:



Name	Cursor
clk	1
j	1
k	1
q	1
qnot	0
reset	0

SCHEMATIC :



5c. Master-Slave Flip Flop :

Top Module :

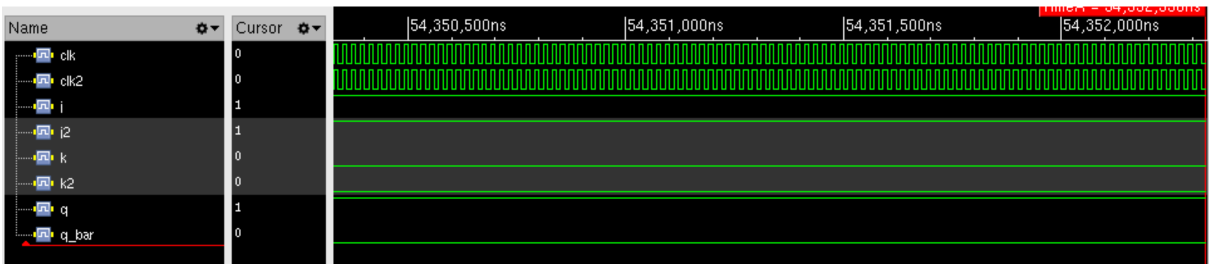
```
module ms_jkff(q,q_bar,clk,j,k);
output q,q_bar;
input clk,j,k; reg
tq,q,q_bar;
always @(clk)
begin
if (!clk)
begin
if (j==1'b0 && k==1'b1) tq
<= 1'b0;
else if (j==1'b1 && k==1'b0) tq <=
1'b1;
else if (j==1'b1 && k==1'b1) tq
<= ~tq;
end
if (clk)
begin
q <= tq; q_bar
<= ~tq; end
end
endmodule
```

Test Module :

```
module tb_ms_jkff;
reg clk,j,k;
wire q,q_bar;
wire clk2,j2,k2;
ms_jkff inst(q,q_bar,clk,j,k);
```

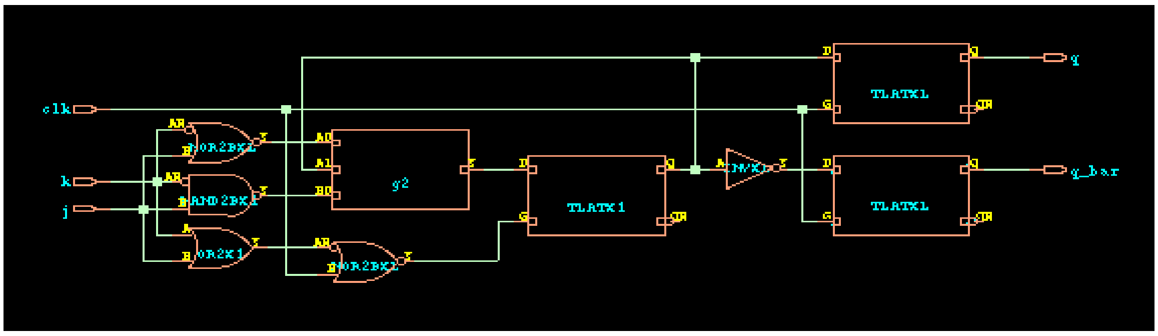
```
assign clk2=clk;
assign j2=j; assign
k2=k; initial
clk = 1'b0;
always #10
clk = ~clk;
initial begin
j = 1'b0; k = 1'b0;
#60 j = 1'b0; k = 1'b1; #40 j
= 1'b1; k = 1'b0; #20 j =
1'b1; k = 1'b1; #40 j = 1'b1;
k = 1'b0; #5 j = 1'b0; #20 j =
1'b1;
end
endmodule
```

RESULT :



```
Name      Cursor
clk       0
clk2      0
j         1
j2        1
k         0
k2        0
q         1
q_bar     0
```

SCHEMATIC :



5d. T- Flip Flop :

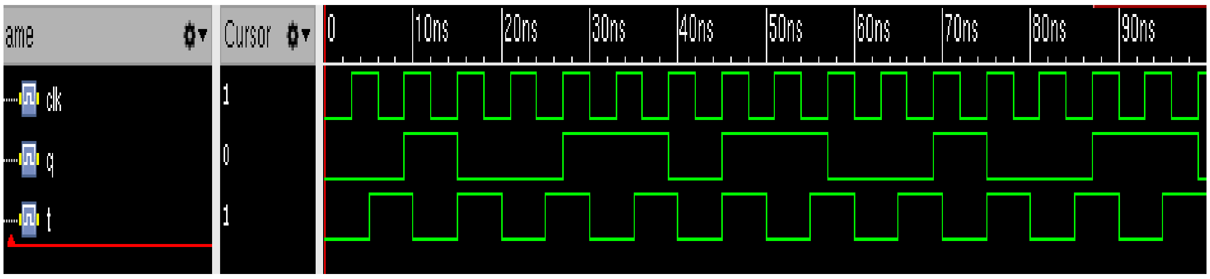
Top Module :

```
module tffmod(t, clk, q);
input t;
input clk;
output q;
reg q;
initial
q <= 0;
always @(posedge clk) q
<= q^t;
endmodule
```

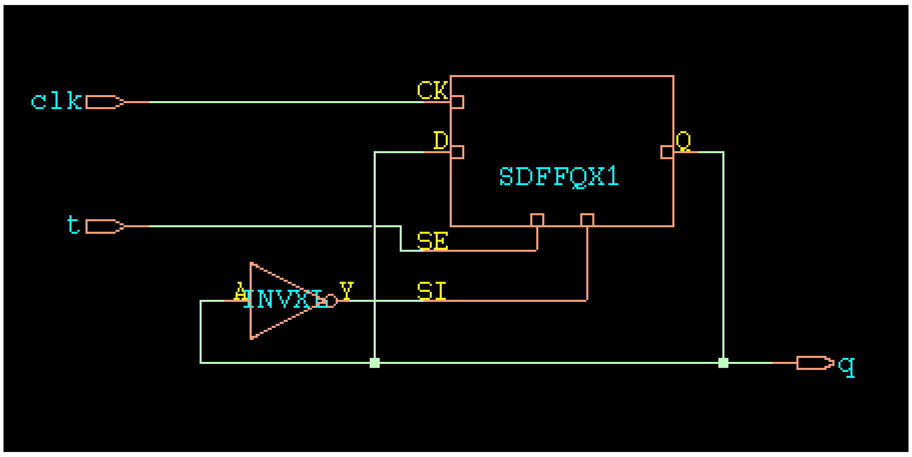
Test Module :

```
module tflipflopt_b;
reg t;
reg clk;
wire q;
tffmod uut (.t(t), .clk(clk),.q(q));
initial begin
t = 0;
clk = 0;
#100;
end
always #3 clk=~clk;
always #5 t=~t;
initial
#100 $stop;
endmodule
```

RESULT :



SCHEMATIC :



5e. SR-Flip Flop ;

Top Module :

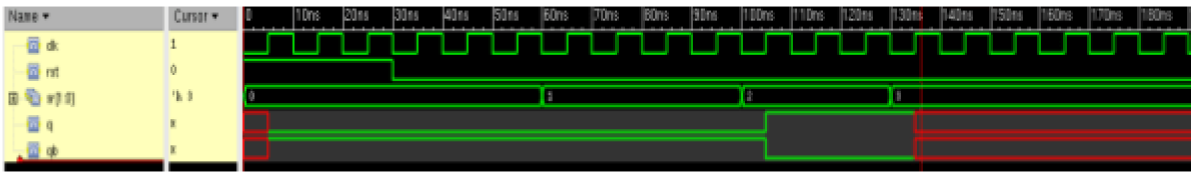
```
module SR_flipflop(q,q1,r,s,clk);
output q,q1;
input r,s,clk;
reg q,q1;
initial
begin
q=1'b0; q1=1'b1;
end
always @(posedge clk)
begin
```

```
case({s,r})
{1'b0,1'b0}: begin q=q; q1=q1; end
{1'b0,1'b1}: begin q=1'b0; q1=1'b1; end
{1'b1,1'b0}: begin q=1'b1; q1=1'b0; end
{1'b1,1'b1}: begin q=1'bx; q=1'bx; end
endcase
end
endmodule
```

Test Module :

```
module test;

reg clk=0;
reg s=0;
reg r=0;

wire q, qnot;

  jkff dut(reset, clk,j,k,q,qnot);

initial
  begin

    s=1'b1;
    r=1'b1;

#25 $finish;
  end
always #1 clk=~clk;
endmodule
```

RESULT :



LAB 6 : SERIAL AND PARALLEL ADDER

6a. Parallel Adder :

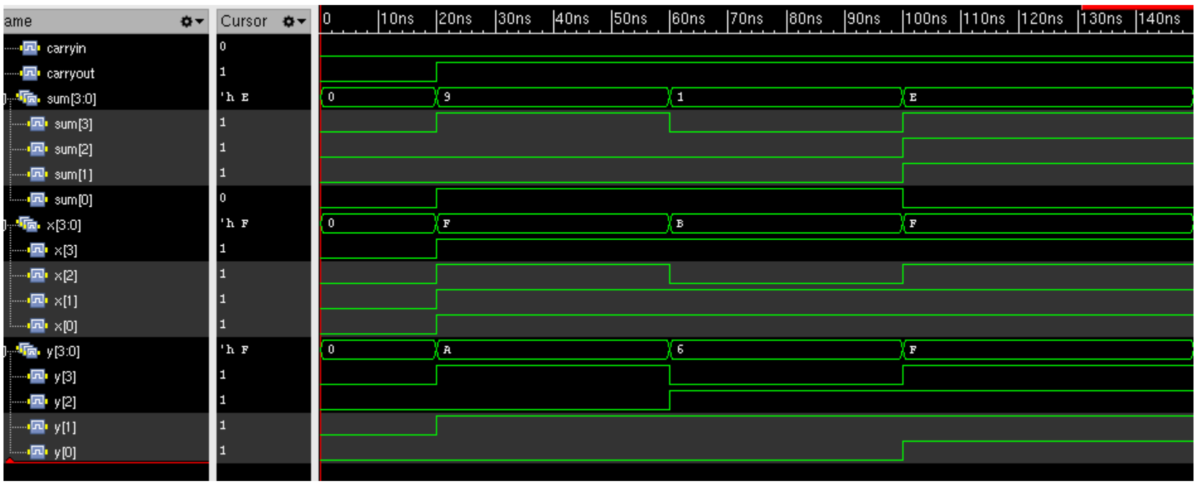
Top Module :

```
module adder4 ( carryin,x,y,sum,carryout); input
carryin;
input [3:0] x,y;
output [3:0] sum;
output carryout;
fulladd stage0 (carryin,x[0],y[0],sum[0],c1);
fulladd stage1 (c1,x[1],y[1],sum[1],c2);
fulladd stage2 (c2,x[2],y[2],sum[2],c3); fulladd
stage3 (c3,x[3],y[3],sum[3],carryout); endmodule

module fulladd (cin,x,y,s,cout);
input cin,x,y;
output s,cout; assign
s = x^y^cin;
assign cout =( x & y) | (x & cin) |( y & cin);
endmodule
```

Test Module :

```
module adder4_t ;
reg [3:0] x,y;
reg carryin; wire
[3:0] sum; wire
carryout;
adder4 a1 ( carryin,x,y,sum,carryout);
initial
begin
x = 4'b0000; y= 4'b0000;carryin = 1'b0; #20 x
=4'b1111; y = 4'b1010;
#40 x =4'b1011; y =4'b0110; #40
x =4'b1111; y=4'b1111;
#50 $finish;
end
endmodule
```

RESULT :



6b. Serial Adder :

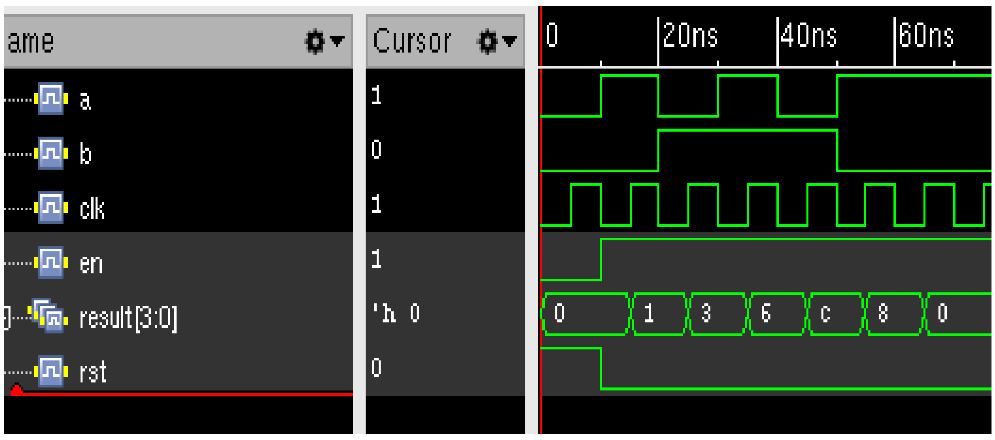
Top Module :

```
module adder_serial(
input clk,rst,
input en, // on Enable, addition will start input a,
// 4-bit adder
input b,
output [3:0] result
);
reg [3:0] y;
reg carry;

always@(posedge rst or posedge clk)
begin
if (rst)
begin
y = 4'b0;
carry = 1'b0;
end
else if (en)
begin
y[3] = y[2];
y[2] = y[1];
y[1] = y[0];
{carry,y[0]} = a + b + carry; end
end
assign result = y;
endmodule
```

Test Module :

```
module serial_adder_test;
reg clk,rst,en,a,b;
wire [3:0] result;
adder_serial U1 (clk,rst,en,a,b,result); //instantiation initial
clk=1'b0;
always
#5 clk=~clk;
initial
begin
rst =1'b1;en=1'b0;a=0;b=0;
#10 rst =1'b0;en=1'b1;a=1;b=0;
#10 rst =1'b0;en=1'b1;a=0;b=1;
#10 rst =1'b0;en=1'b1;a=1;b=1;
#10 rst =1'b0;en=1'b1;a=0;b=1;
#10 rst =1'b0;en=1'b1;a=1;b=0;
#200 $finish;
end
endmodule
```

RESULT :



LAB 7 : 4-BIT COUNTERS

7a. Asynchronous Counter :

Top Module :

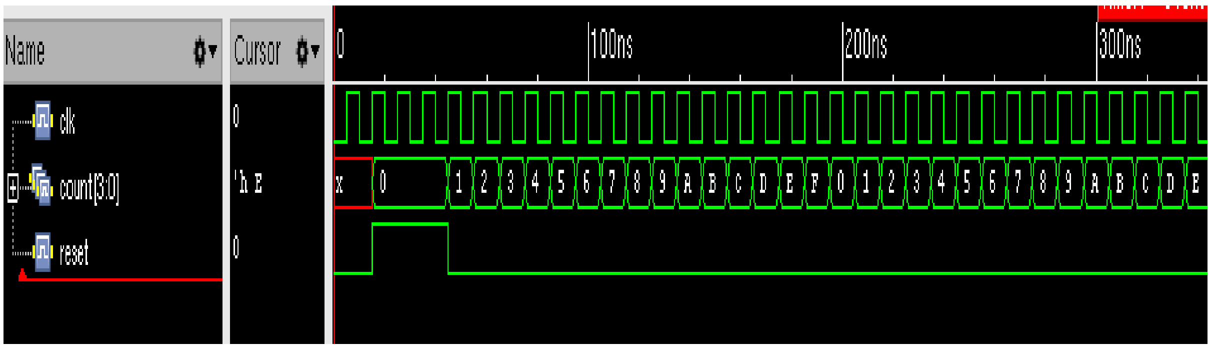
```
module counter_behav ( count,reset,clk);
input wire reset, clk;
output reg [3:0] count;
always @(posedge clk or posedge reset) begin
if (reset)
count <= 4'b0000;
else
count <= count + 4'b0001;
end
endmodule
```

Test Module :

```
module mycounter_t ;
wire [3:0] count;
reg reset,clk;
initial
clk = 1'b0;
always
#5 clk = ~clk;
counter_behav m1 ( count,reset,clk);
initial
begin
reset = 1'b0 ;
```

```
#15 reset =1'b1;
#30 reset =1'b0;
#300 $finish;
end
initial
$monitor ($time, "Output count = %d ",count );
endmodule
```

RESULT :



7b. Synchronous Counter :

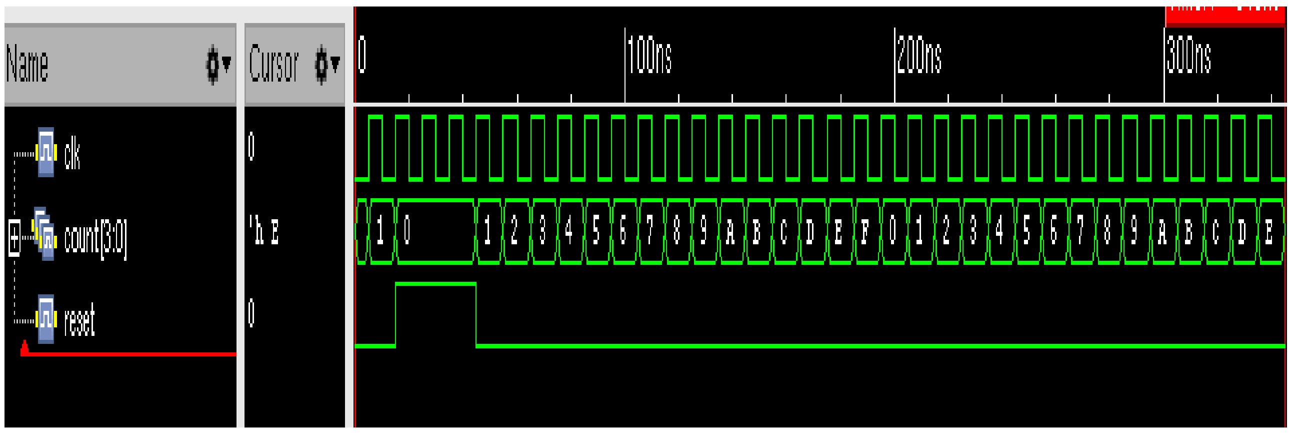
Top Module :

```
module counter_behav ( count,reset,clk);
input wire reset, clk;
output reg [3:0] count;
initial
count =4'b0000; always
@(posedge clk) if (reset)
count <= 4'b0000;
else
count <= count + 4'b0001;
endmodule
```

Test Module :

```
module mycounter_t ;
wire [3:0] count;
reg reset,clk;
initial
clk = 1'b0;
always
#5 clk = ~clk;
counter_behav m1 ( count,reset,clk);
initial
begin
reset = 1'b0 ; #15
reset =1'b1;
#30 reset =1'b0;
#300 $finish;
end
initial
$monitor ($time, "Output count = %d ",count );
endmodule
```

RESULT :



Name
Cursor
0
100ns
200ns
300ns
clk
count[3:0]
reset
0
'h E
0
1 0 1 2 3 4 5 6 7 8 9 A B C D E F 0 1 2 3 4 5 6 7 8 9 A B C D E

LAB 8 : SUCCESSIVE APPROXIMATION REGISTER

Top Module :

```
module sar (digitalout,done,comp,start,reset,clk);
output [3:0] digitalout;
output done;
input clk, start, reset, comp;
reg [3:0]ring_count;
reg [3:0]digital;
wire      D4,set0,set1,set2,set3;
assign    D4    =    ring_count[0];
assign done = !D4;
always @(posedge clk or negedge reset)
begin
if (~reset)
ring_count <= 4'b1000;
else
begin
if (start)
ring_count <= 4'b1000;
else
ring_count <= (ring_count>>1); end
end
assign   set3   =   ring_count[3];
assign   set2   =   ring_count[2];
assign   set1   =   ring_count[1];
assign set0 = ring_count[0];
always @(posedge clk or negedge reset)
begin
if(~reset)
```

```
digital[3] <= 1'b1;
else
if(start)
digital[3] <= 1'b1; else
if(set3) digital[3] <=
comp; end
always @(posedge clk or negedge reset)
begin
if(~reset) digital[2]
<= 1'b1; else
if(start)
digital[2] <= 1'b1; else
if(set2) digital[2] <=
comp; end
always @(posedge clk or negedge reset)
begin
if(~reset) digital[1]
<= 1'b1; else
if(start)
digital[1] <= 1'b1; else
if(set1) digital[1] <=
comp; end
always @(posedge clk or negedge reset)
begin
if(~reset) digital[0]
<= 1'b1; else
```

```
if(start)
digital[0] <= 1'b1; else
if(set0) digital[0] <=
comp; end
assign digitalout = (digital) | (ring_count);
endmodule
```

Sub Module :

```
module dac (comp,sar_out,vref_d,vin_d,clk,start);
output comp;
input clk,start; input
[3:0]sar_out; input
[63:0]vref_d; input
[63:0]vin_d; reg
comp;
real v_dac,vref,vin; always
@ (vin_d or start) begin
vref = $bitstoreal(vref_d); vin
= $bitstoreal(vin_d); end
always @*
begin if(start)
comp = 1'b0;
else
begin
v_dac = (vref/15)*(sar_out); if
(vin<v_dac)
comp = 1'b0;
```

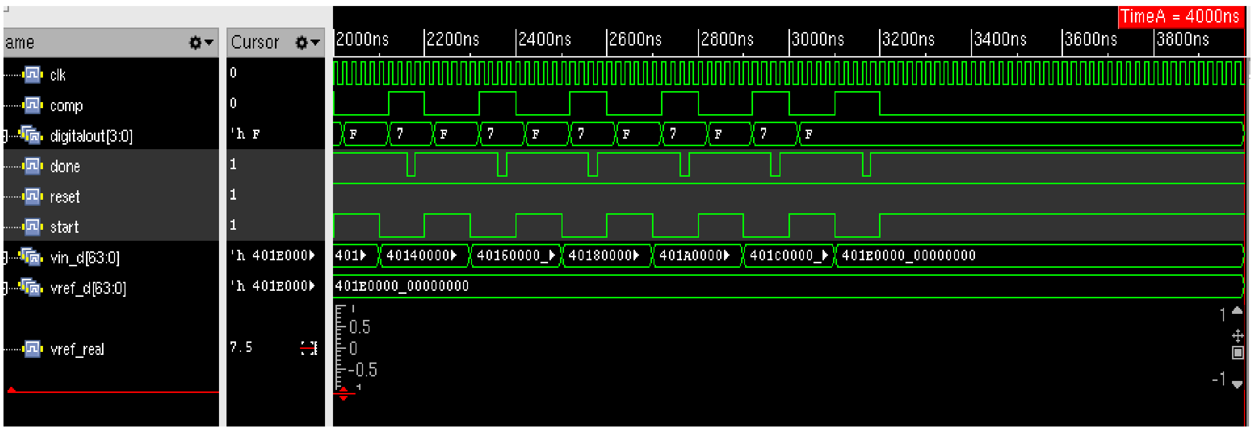
```
else
comp = 1'b1;
end
end
endmodule
```

Test Module :

```
module sar_tb; reg
clk,reset,start;
reg [63:0] vref_d,vin_d;
wire done, comp;
wire [3:0] digitalout;
real vref_real = 7.5;
sar s1 (digitalout,done,comp,start,reset,clk); dac d1
(comp,digitalout,vref_d,vin_d,clk,start); initial
begin
clk = 1'b1; start
= 1'b1; #4000
$finish; end
always #10 clk = ~clk; initial
begin
#1;reset = 1'b1;
#10; reset = 1'b0;
#1; reset = 1'b1;
end
initial
begin
#10 ;
```

```
stimulus (0.0,0.5,vref_real,8'd5);
end
task stimulus (input analog, input step, input reference, input [7:0]delay);
real
analog,step;
real reference;
begin
while(analog <= reference)
begin
repeat(delay)
@(posedge
clk); start <=
1'b0;
vref_d = $realtobits (reference);
vin_d = $realtobits (analog);
@(posedge done)
analog = analog + step;
@(posedge clk);
start <=
1'b1; end
end
endtask
endmodu
le
```

RESULT :

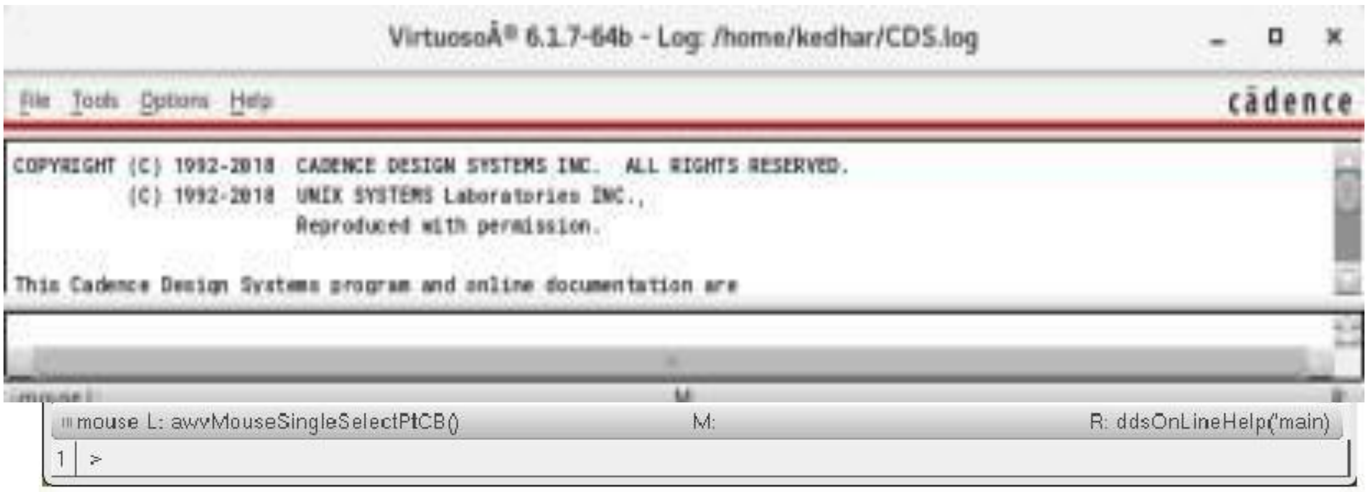


CUSTOM IC TOOL FLOW

INVERTER

Creating a new Library:

- Create a new folder for example as test
- Open the terminal inside the folder and source the cshrc file by using a command **source /home/intall/cshrc**
- After that invoke the tool by using a command **virtuoso &**
- Then the below window will be popup.

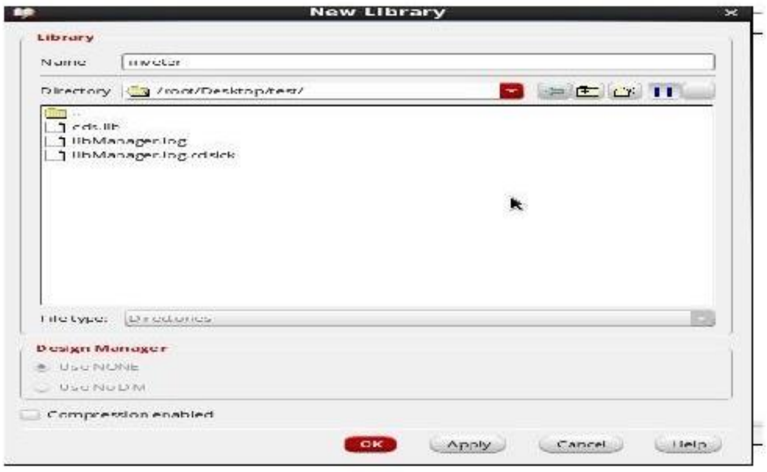


```
COPYRIGHT (C) 1992-2018  CADENCE DESIGN SYSTEMS INC.  ALL RIGHTS RESERVED.
          (C) 1992-2018  UNIX SYSTEMS Laboratories INC.,
                         Reproduced with permission.

This Cadence Design Systems program and online documentation are
```

```
mouse L: awvMouseSingleSelectPtCB()          M:          R: ddsOnLineHelp('main')
1 | >
```

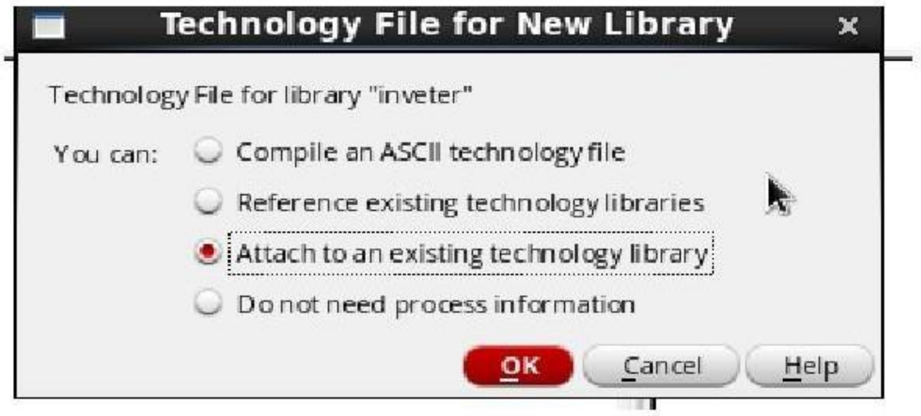
- In the above window Go to **Tools -->Library manger-->new library**

Give the library name and click ok , then a tab opens to attach the technology library as shown below



Name: inveter
Directory: /root/Desktop/test/
..
cds.lib
libManager.log
libManager.log.cdslck
File type: Directories
Design Manager: [✓ Use NONE] Use NoDM
Compression enabled: ☐

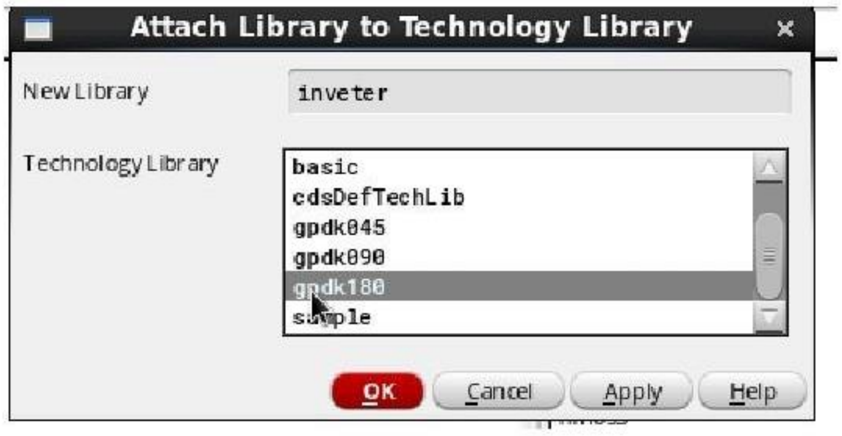
Here Select the option as **Attach to an existing technology library** and click on OK.



Technology File for library "inveter"

You can:
- Compile an ASCII technology file
- Reference existing technology libraries
- [✓ Attach to an existing technology library]
- Do not need process information

• In the "Attach Library to Technology Library" window, select gpdk180 from the Technology Library filed and click OK.

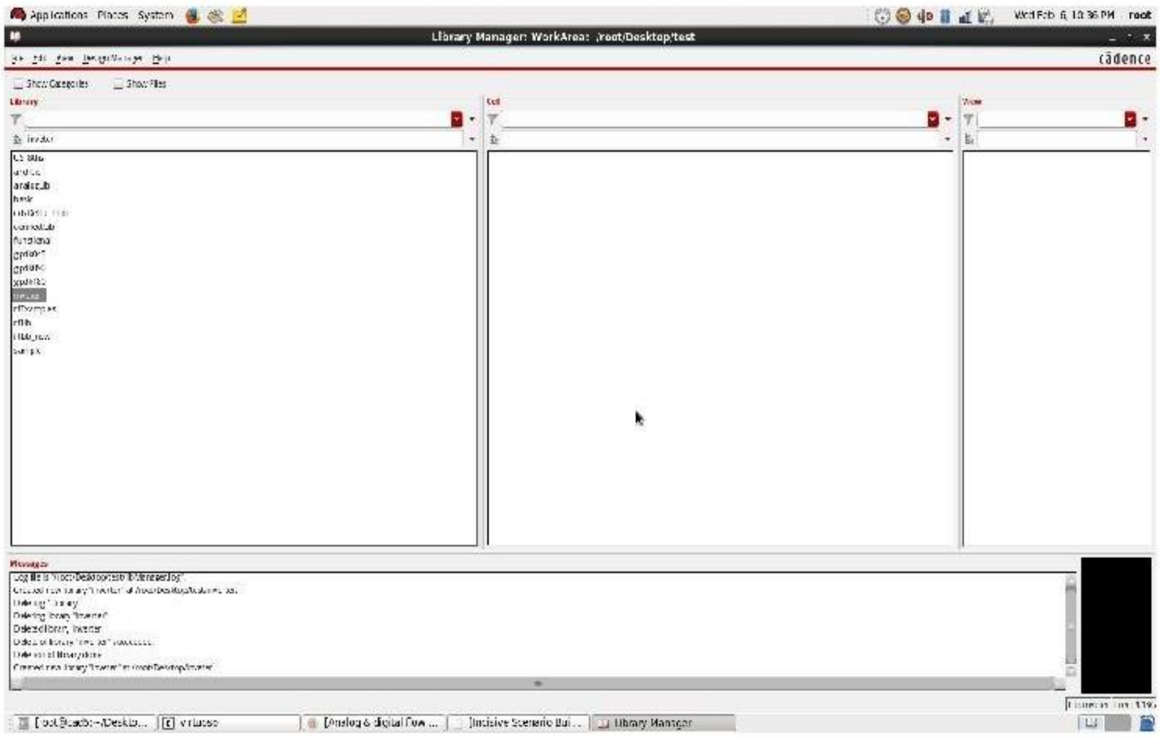


New Library: inveter

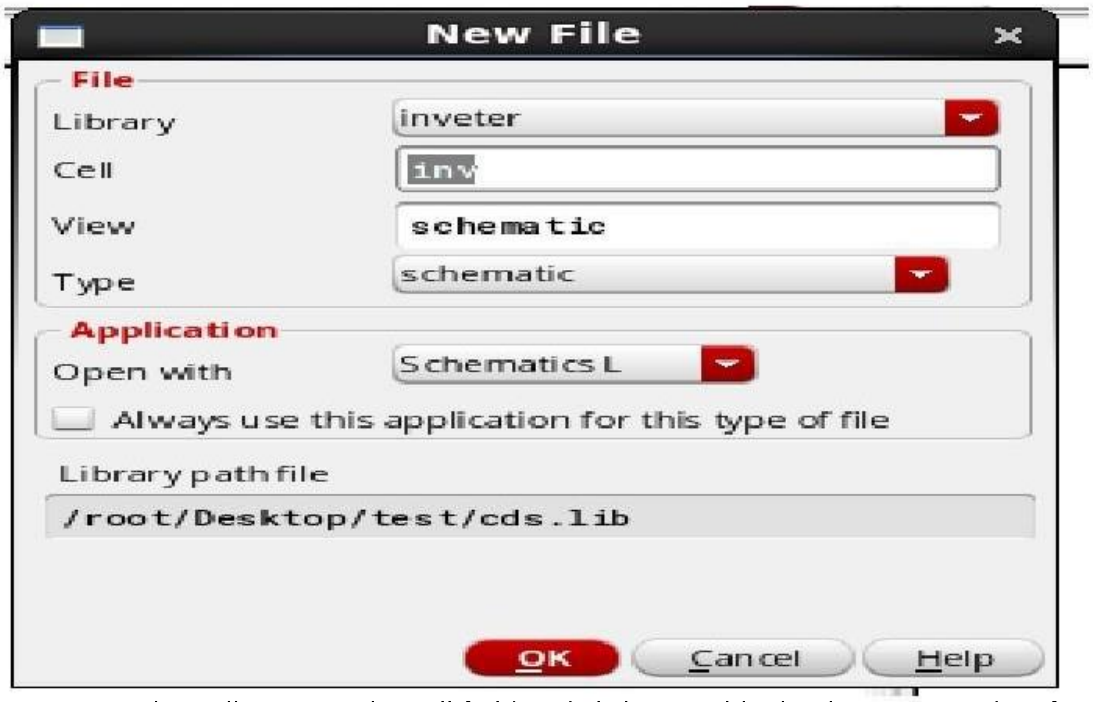
Technology Library:
- basic
- cdsDefTechLib
- gpdk045
- gpdk090
- [✓ gpdk180]
- sample

Creating a schematic cell view:

• In the Library manager, select the **library**(inverter)**, File --> New --> Cellview** Then the below Window will be pop-up.

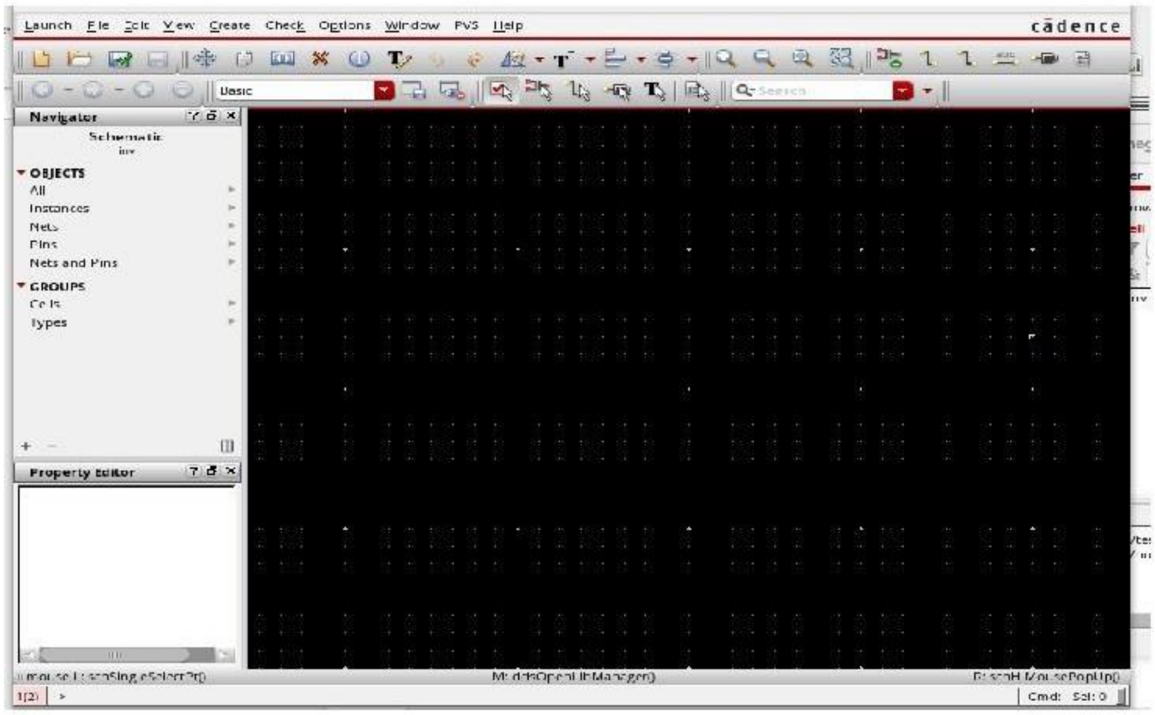


```
Log file is "/root/Desktop/test/libManager.log".
[illegible]
Deleting library "inverter"
Deleted library inverter
[illegible]
Created new library "inverter" in /root/Desktop/inverter
```

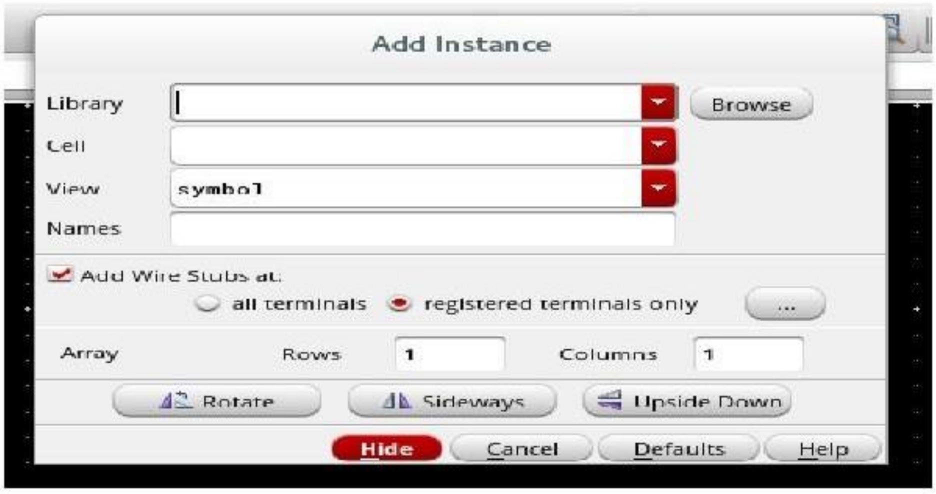


Library: inveter
Cell: inv
View: schematic
Type: schematic
Open with: Schematics L
Always use this application for this type of file
Library path file: /root/Desktop/test/cds.lib

- Give the Cell name in the cell field and click OK. A blank schematic window for the Inverter design appears as shown in below.



- In the inverter schematic window, press letter " **i** " (Insert) to get the components like pmos, nmos, etc or go to create in top menu and select **Create --> Instance**
- If we press ' **i** ' then a window opens as shown as below.


Add Instance

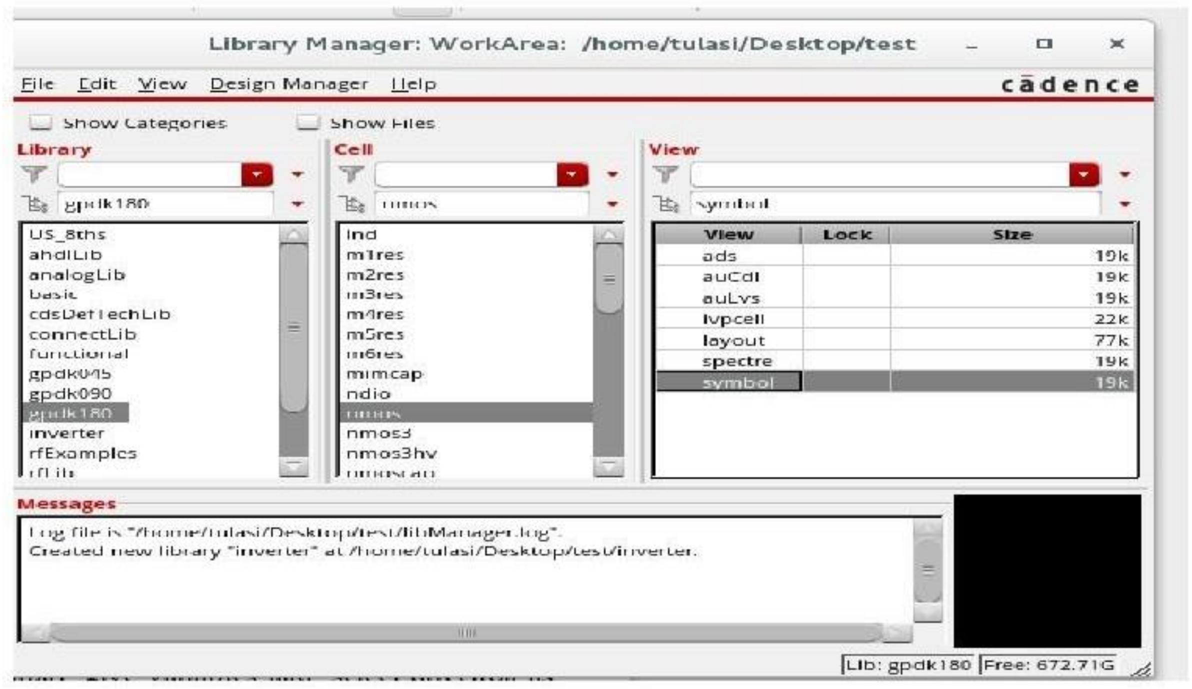
Library
Cell
View symbol
Names

[✓] Add Wire Stubs at:
all terminals [● registered terminals only]

Array Rows 1 Columns 1

In the above window we need to select **Browse** option to get the components list.

Now, Browse **gpdk180 --> cell (component) --> symbol (view) .**



Library Manager: WorkArea: /home/tulasi/Desktop/test

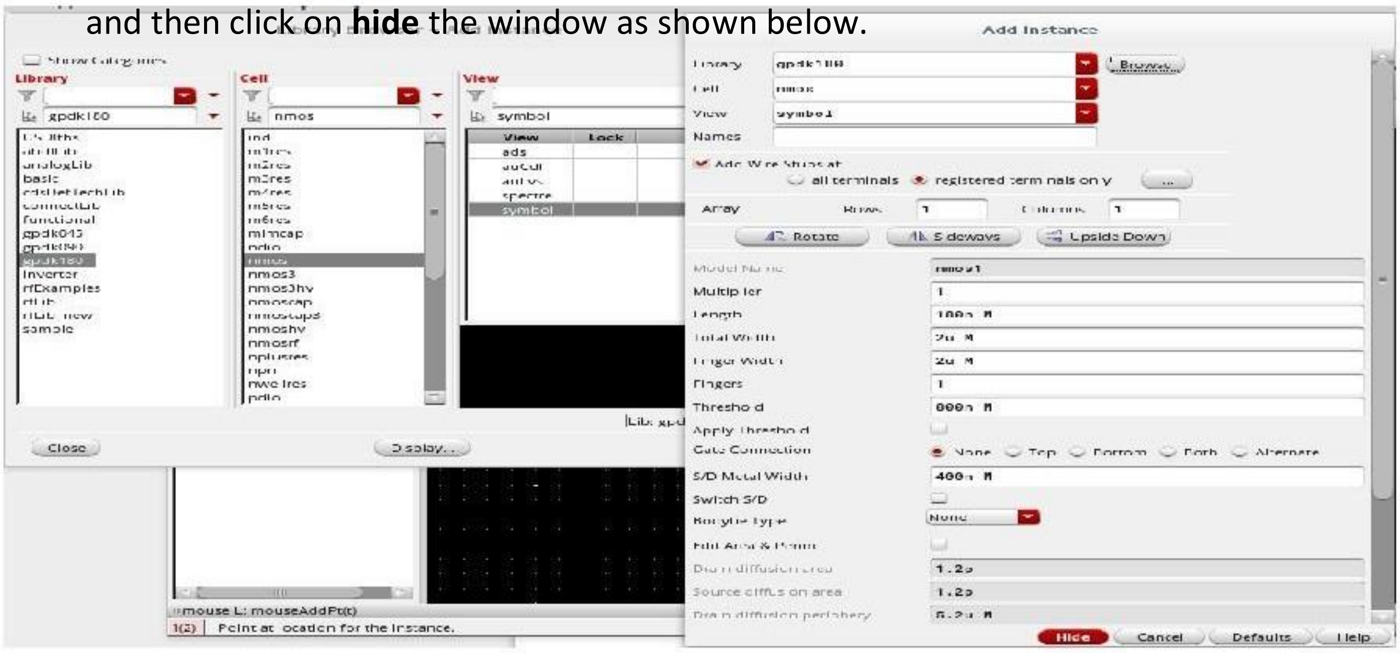
View	Lock	Size
ads		19k
auCdl		19k
auLvs		19k
ivpcell		22k
layout		77k
spectre		19k
symbol		19k

Messages

```
Log file is "/home/tulasi/Desktop/test/libManager.log".
Created new library "inverter" at /home/tulasi/Desktop/test/inverter.
```

Lib: gpdk180 Free: 672.71G

After selecting the symbol we need to specify the values of the selected component and then click on **hide** the window as shown below.

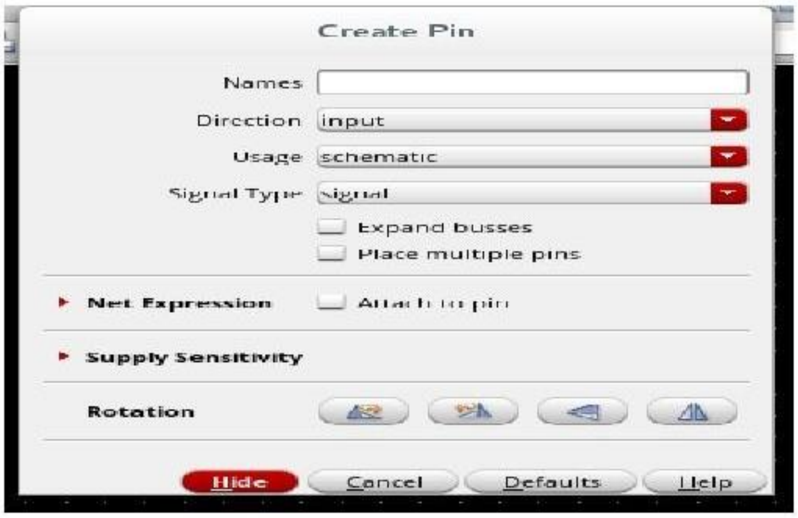


Add Instance

Library	gpdk180
Cell	nmos
View	symbol
Names	
Model Name	nmos1
Multiplier	1
Length	180n M
Total Width	2u M
Finger Width	2u M
Fingers	1
Threshold	800n M
Gate Connection	[✓ None] Top Bottom Both Alternate
S/D Metal Width	400n M
Bodytie Type	None
Drain diffusion area	1.2p
Source diffusion area	1.2p
Drain diffusion periphery	5.2u M

```
mouse L: mouseAddPt()
1(3) | Point at location for the instance.
```

- After placing the components select the pins by pressing letter ‘ **p** ’ or go to create in top menu by **Create --> Pin**
- After pressing P a Create Pin window will be pop-up.



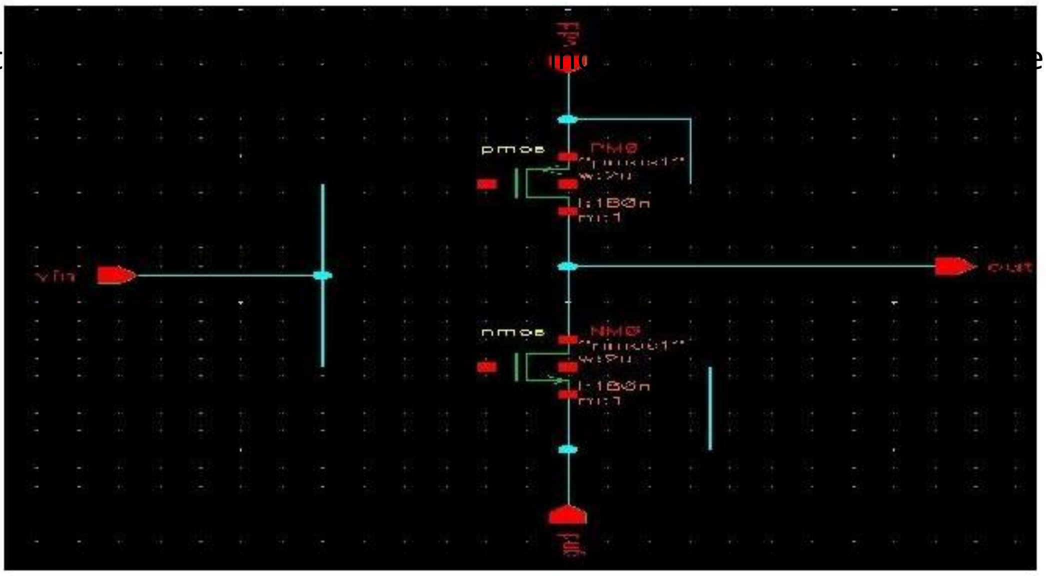
Create Pin

Names:
Direction: input
Usage: schematic
Signal Type: signal
☐ Expand busses
☐ Place multiple pins
Net Expression ☐ Attach to pin
Supply Sensitivity
Rotation

In the Create Pin Window Specify the name of the pins(input, output, VDD, VSS) and the Direction of the pins, then select Hide.

- For rotation of pins we can press letter “ **r** “ .
- After placing the pins, press letter “ **f** “ so that the schematic will fit to the screen.
- After placing all the components and pins, press **‘w’** to give the connections or go to create in top menu and select **Create --> Wire.**
- Click the Check and Save icon in the schematic editor window to Save the Design.

Go back t ematic design.

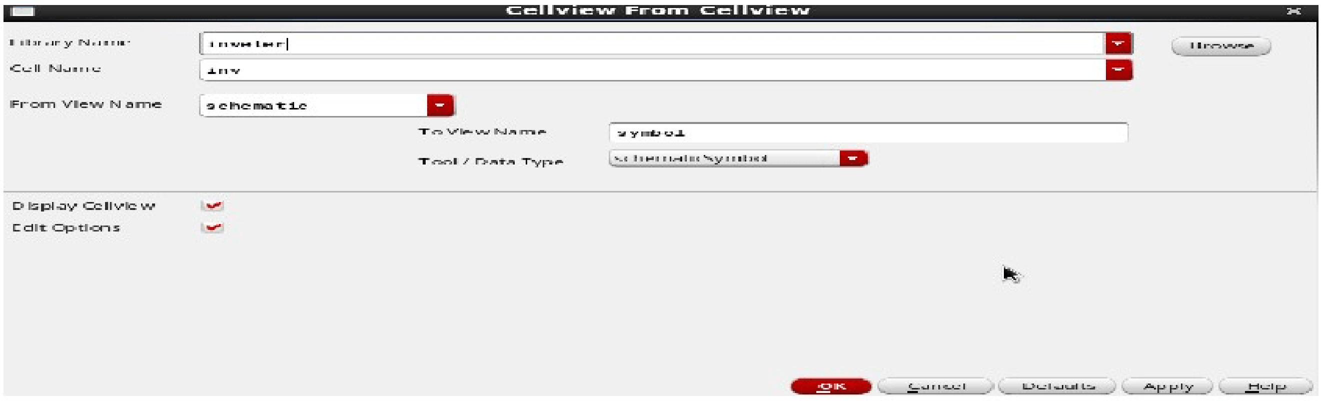


Symbol creation:

1. In the Schematic window, select Create from top menu

Create —> Cellview—> From Cellview.

2. The "**Cellview From Cellview**" form appears. With the Edit Options function active, you can control the appearance of the symbol to generate.



Cellview From Cellview

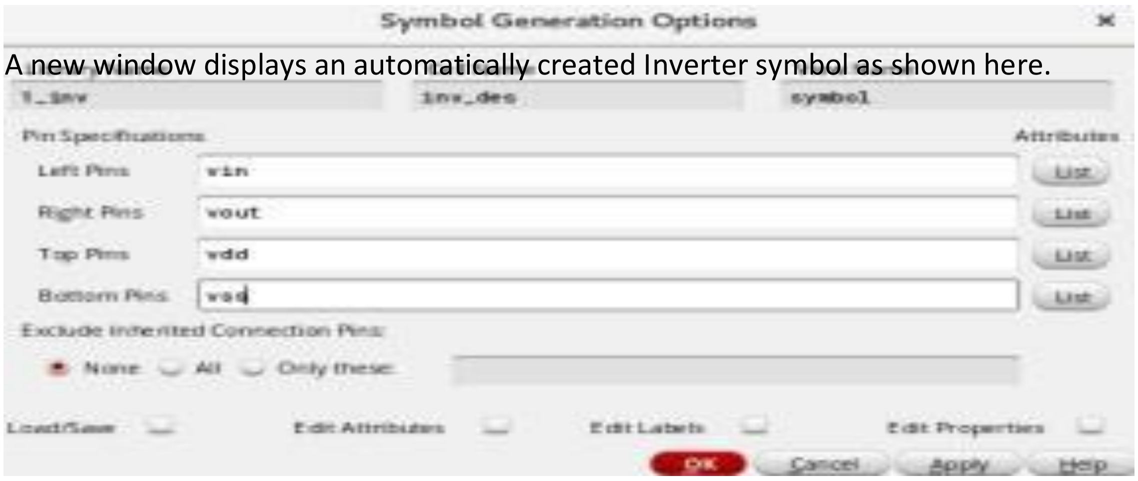
Library Name	inverter	Browse
Cell Name	inv	
From View Name	schematic	
To View Name	symbol	
Tool / Data Type	SchematicSymbol	
Display Cellview	✓	
Edit Options	✓	

3. Verify that the "**From View Name**" field is set to schematic, and the "To View Name" field is set to symbol, with the "Tool/Data Type" set as Schematic Symbol.

4. Click OK in the "**Cellview From Cellview**" form. The Symbol Generation Form appears. Modify the Pin Specifications as follows:

5. Click OK in the Symbol Generation Options form.

6. A new window displays an automatically created Inverter symbol as shown here.



Symbol Generation Options

1_inv	inv_des	symbol

Pin Specifications

Left Pins	vin	List
Right Pins	vout	List
Top Pins	vdd	List
Bottom Pins	vss	List

Exclude Inherited Connection Pins: [✓ None] All Only these:

Editing a symbol:

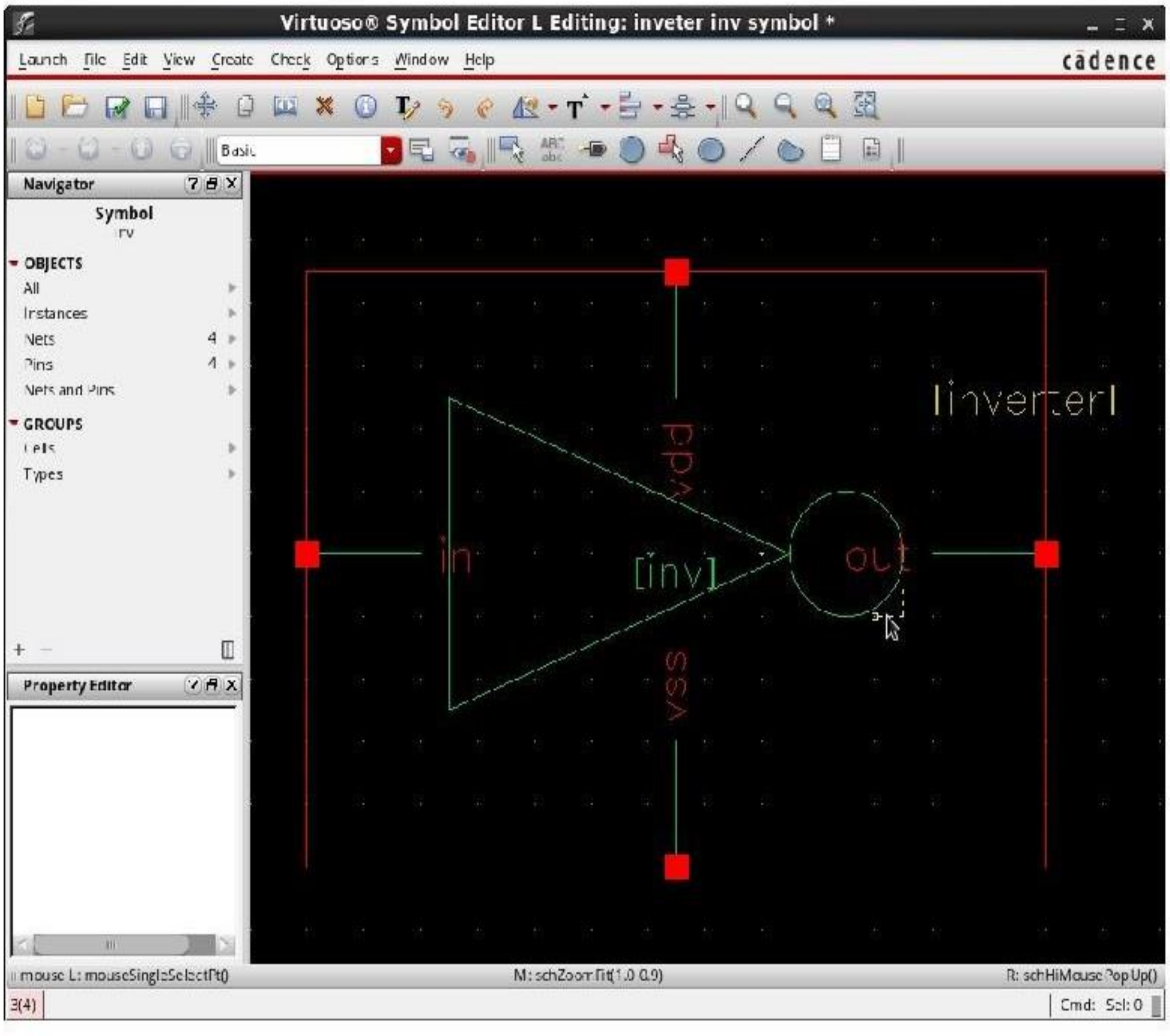
It is optional, we can modify it or not because the functionality will not change. we will modify the inverter symbol as look like an Inverter gate symbol.

1. Move the cursor over the automatically generated symbol, until the green rectangle is highlighted, click left to select it.

2. Click Delete icon in the symbol window, similarly select the red rectangle and delete that.

3. Execute “**Create –->Shape –->polygon”**, and draw a shape similar to triangle ass shown below.

4. After creating the triangle press **ESC** key. Execute “**Create –->Shape –-> Circle”** to make a circle at the end of triangle.



mouse L: mouseSingleSelectPt() M: schZoomFit(1.0 0.9) R: schHiMousePopUp()
3(4) Cmd: Sel: 0

5. You can move the pin names according to the location.

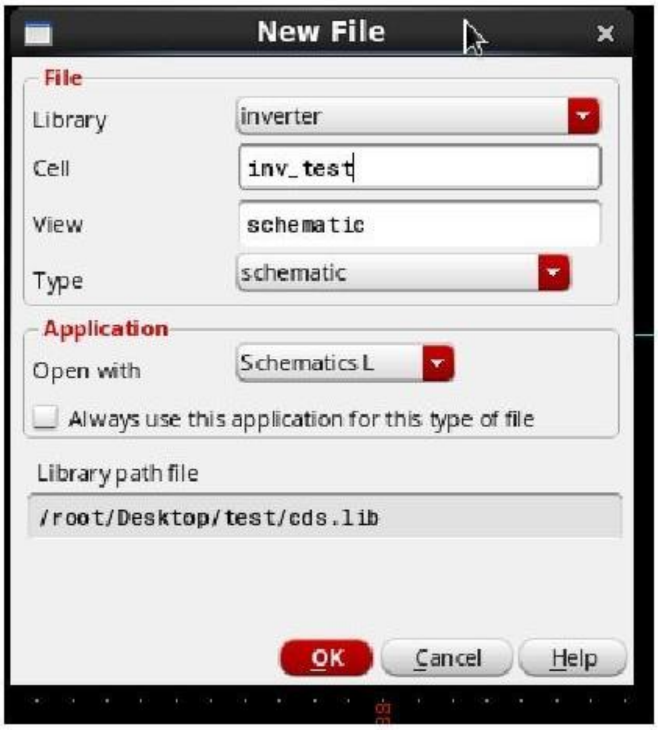
6. Execute **“Create —>Selection Box”**. In the Add Selection Box form, click Automatic. A new red selection box is automatically added.

7. After creating symbol, click on the save icon in the symbol editor window to save the symbol. In the symbol editor, execute **File —>Close** to close the symbol view window.

Inverter test design:

1. In the CIW or Library Manager, execute “**File—>New—> Cellview”**.

2. Set up the New File form as follows:



File

Library	inverter
Cell	inv_test
View	schematic
Type	schematic

Application

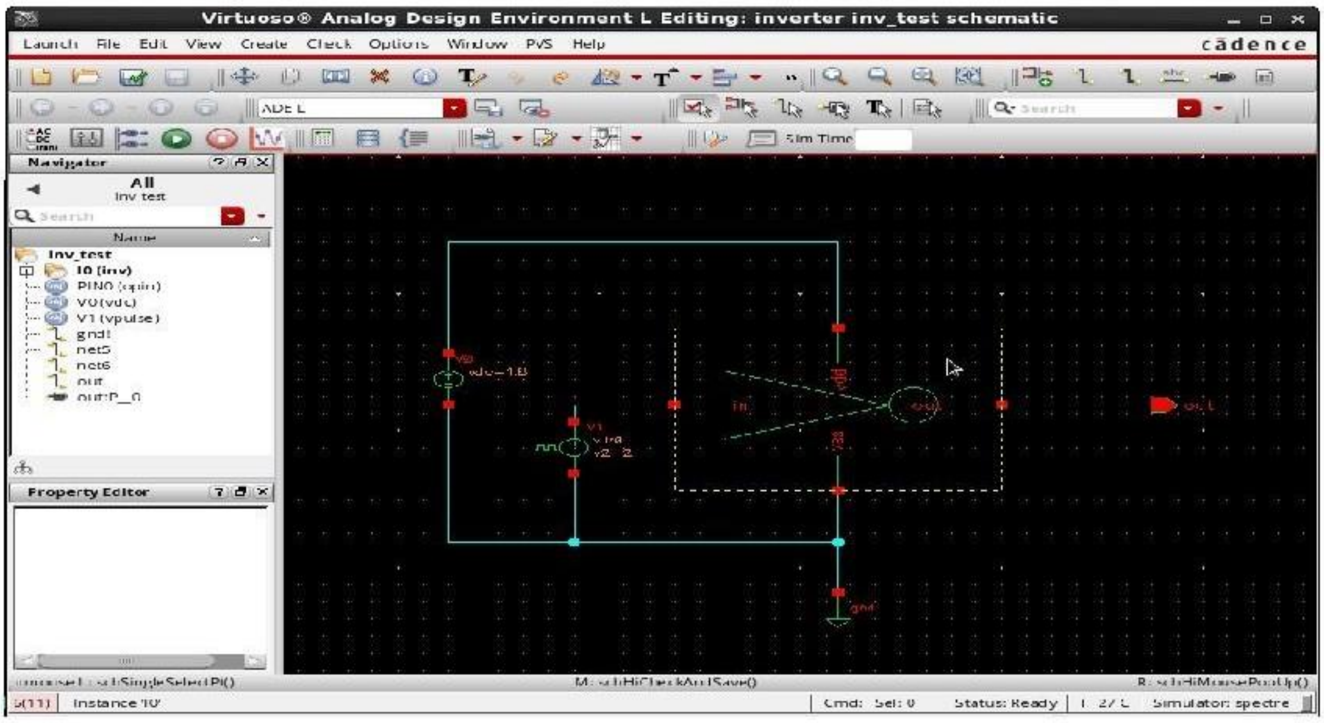
Open with	Schematics L

☐ Always use this application for this type of file

Library path file

/root/Desktop/test/cds.lib

3. Click **OK** when done. A blank schematic window for the Inverter_Test design appears.


Virtuoso® Analog Design Environment L Editing: inverter inv_test schematic
Launch File Edit View Create Check Options Window PVS Help
cadence
ADE L
Navigator
Property Editor
Status: Ready
Simulator: spectre

4. Add the components using **Create —>Instance** or by pressing letter " **I** ".

5. **Click the Wire** (narrow) icon and wire your schematic or you can also press the " **w** " key.

6. Go to **Create—>Wire** (narrow) in top menu and make the necessary connections. 7.Click

on the **Check and Save** icon to save the design.

Simulation:

In the Inverter_Test schematic window, select **Launch -->ADE L** in top menu. The Virtuoso Analog Design Environment (ADE) simulation window appears.

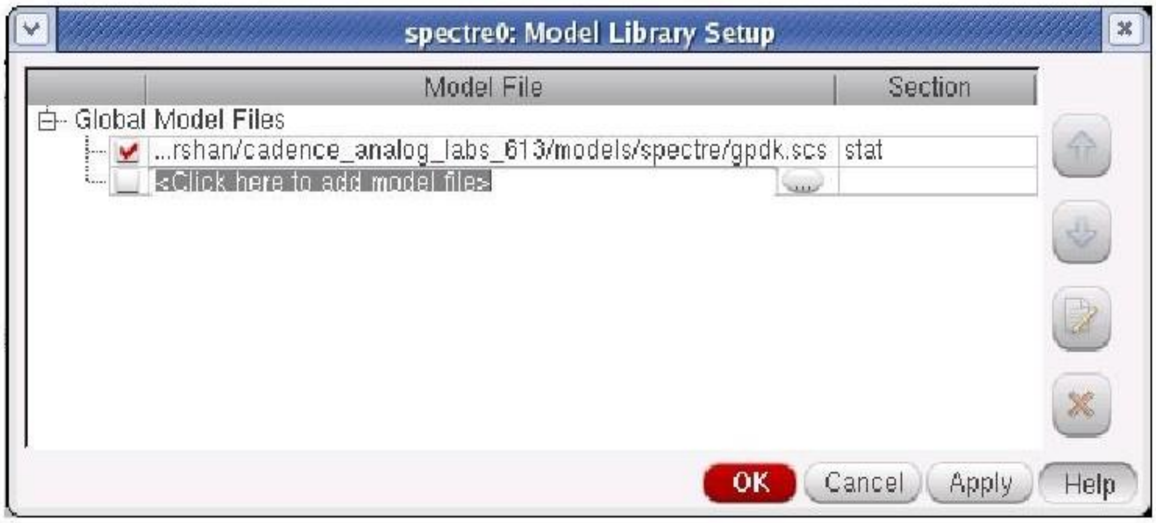
1. In the simulation window, go to top menu and select **Setup—>Simulator/Directory/Host**.

2. In the Choosing Simulator form, set the Simulator field to **spectre** and click **OK**.

3. In the simulation window, go to top menu and select **Setup --->Model Libraries**.

The Model Library Setup form appears. Click the browse button to add **gpdk.scs** if not added by default as shown in the Model Library Setup form.

Remember to select the section type as **stat** in front of the **gpdk.scs** file. The Model Library Setup window will now looks like the below figure.


spectre0: Model Library Setup

Model File	Section
Global Model Files	
✓ ...rshan/cadence_analog_labs_613/models/spectre/gpdk.scs	stat
<Click here to add model file>	

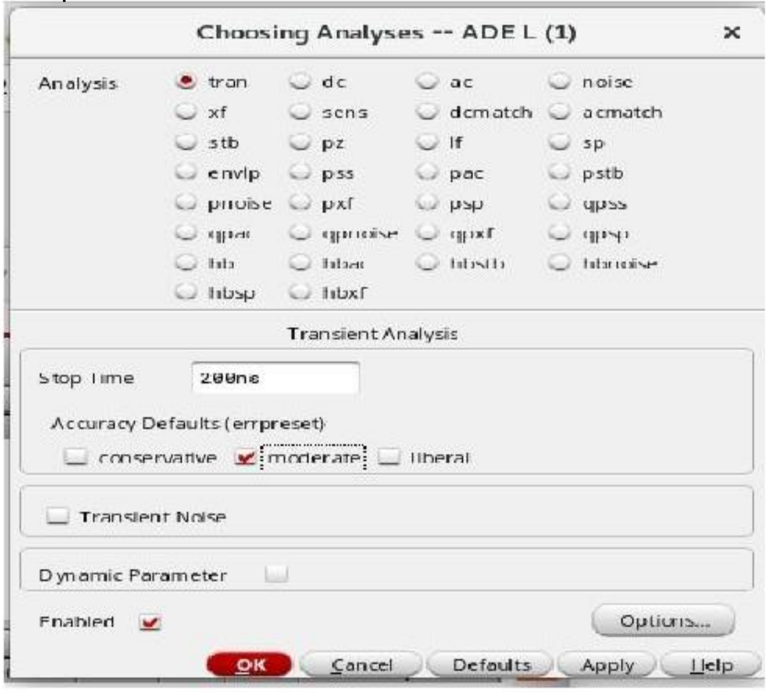
Choosing Analysis:

1. In the Simulation window (ADE), click the **Choose -->Analyses** icon or you can also execute **Analyses --> Choose.**

The Choosing Analysis window appears. This is a dynamic form, the bottom of the form changes based on the selection above.

2. To setup for transient analysis

- In the Analysis section select **tran**
- Set the stop time.


Choosing Analyses -- ADE L (1)

Analysis [✓ tran]

Transient Analysis

Stop Time 200ns

Accuracy Defaults (errpreset)

conservative [✓ moderate] liberal

Transient Noise

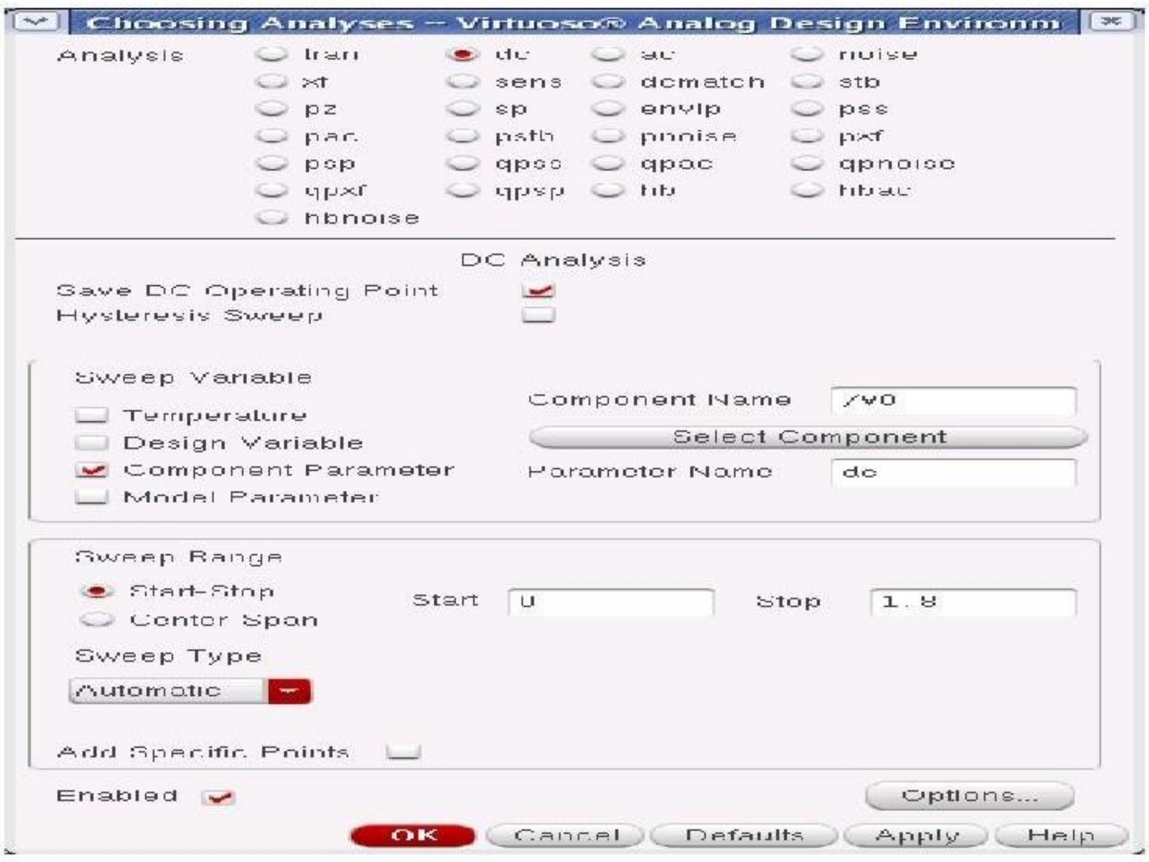
Dynamic Parameter

Enabled ✓

Select the Accuracy Defaults by click on the **moderate or Enabled** option at the bottom, and then click Apply.

To setup for DC Analysis:

- In the Analysis section, select **dc**.
- In the DC Analysis section, enable Save DC Operating Point.
- Select the Component Parameter from Sweep Variable section.
- Click the Select Component, Which takes you to the schematic window.
- Select the input signal vpulse source in the test schematic window.
- Select " DC Voltage " in the Select Component Parameter form and click OK.
- In the sweep range section, type start and stop voltages as 0 to 1.8 respectively.
- Check the enable button and then click Apply.



Choosing Analyses -- Virtuoso® Analog Design Environm

Analysis: ○ tran ○ xf ○ pz ○ pac ○ psp ○ qpxf ○ hbnoise [✓ ● dc] ○ sens ○ sp ○ pstb ○ qpss ○ qpsp ○ ac ○ dcmatch ○ envlp ○ pnoise ○ qpac ○ hb ○ noise ○ stb ○ pss ○ pxf ○ qpnoise ○ hbac

DC Analysis

Save DC Operating Point ✓
Hysteresis Sweep ☐

Sweep Variable
☐ Temperature
☐ Design Variable
✓ Component Parameter
☐ Model Parameter

Component Name: /V0
Select Component
Parameter Name: dc

Sweep Range
● Start-Stop
○ Center Span
Start: 0 Stop: 1.8

Sweep Type: Automatic

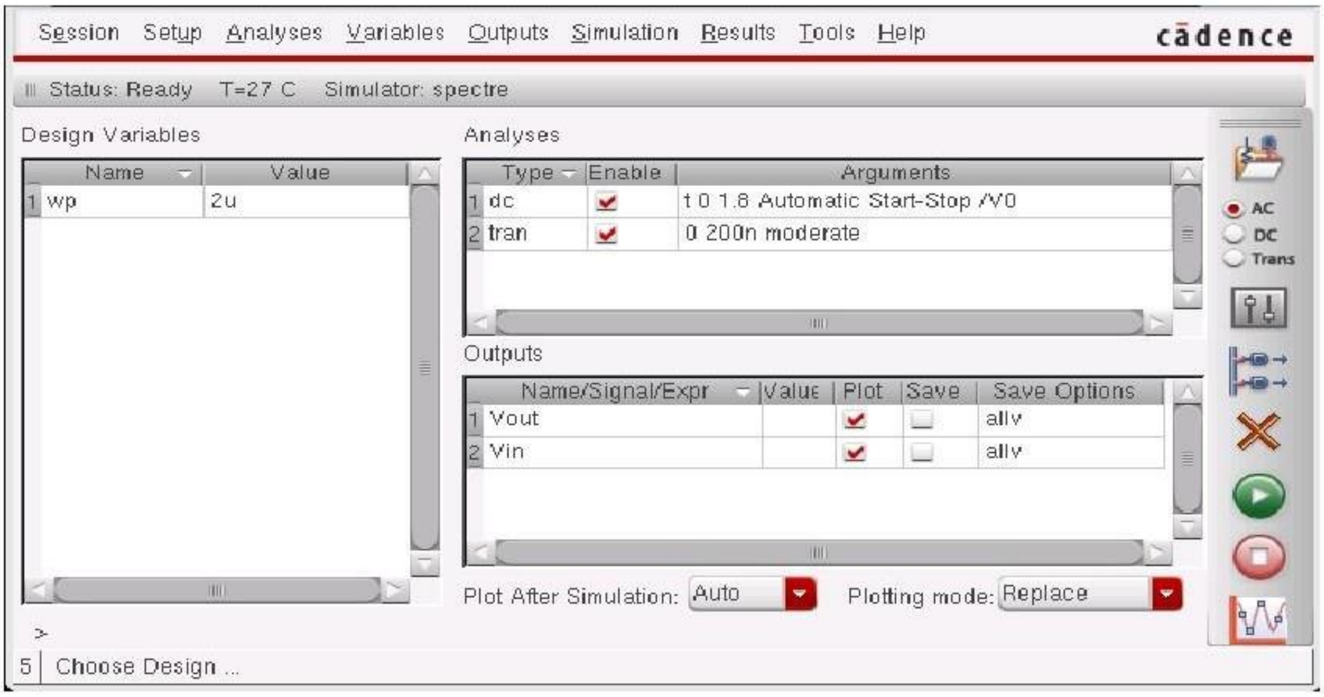
Add Specific Points ☐

Enabled ✓

Options... OK Cancel Defaults Apply Help

Selecting outputs for plotting:

1. Select **Outputs –->To be plotted –->Select on Design(**Schematic) in the simulation window.



Status: Ready T=27 C Simulator: spectre

Design Variables

	Name	Value
1	wp	2u

Analyses

	Type	Enable	Arguments
1	dc	✓	t 0 1.8 Automatic Start-Stop /V0
2	tran	✓	0 200n moderate

Outputs

	Name/Signal/Expr	Value	Plot	Save	Save Options
1	Vout		✓	☐	allv
2	Vin		✓	☐	allv

Plot After Simulation: Auto Plotting mode: Replace

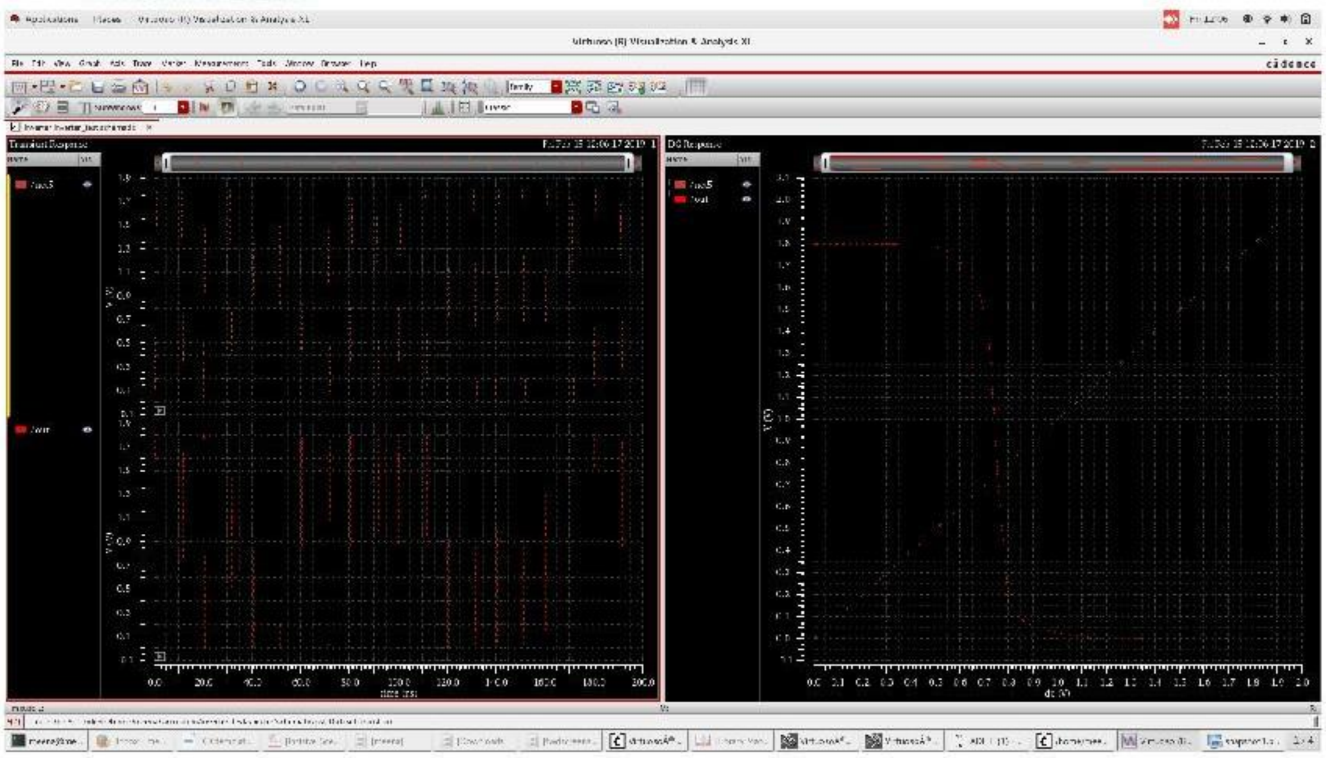
5 Choose Design ...

2. Follow the prompt at the bottom of the schematic window, Click on **output net** Vout, **input net** Vin of the Inverter. Press **Esc** with the cursor in the schematic after selecting it.

Running the simulation:

Execute **Simulation ---> Netlist and Run** in the simulation window(**Green** colour button icon) to start the simulation, this will create the netlist as well as run the simulation.

When simulation finishes, the Transient plots automatically will be popped up along with log file.



Saving the simulator state:

1. In the simulation window, go to **session ---> save state** option**.** The saving state form appears.

2. Set the **save as** field to **State1_inv** and make sure all options are selected under what to save field.

3. Click ok in the saving state form. The simulator state is saved.

Creating a Layout view of inverter:

1. From the Inverter schematic window menu execute **Launch --> Layout XL**. A Start-up Option form appears.

2. Select **Create New** option. This gives a New Cell View Form

3. Check the Library name(**Inverter**), Cell name (**Inv-des**) and View name (**layout**).

4. Click **OK** from the New Cellview form and a blank layout window appear along with schematic window.

Adding components to Layout:

1.Go to **Connectivity –> Generate –> All from Source** or click the icon

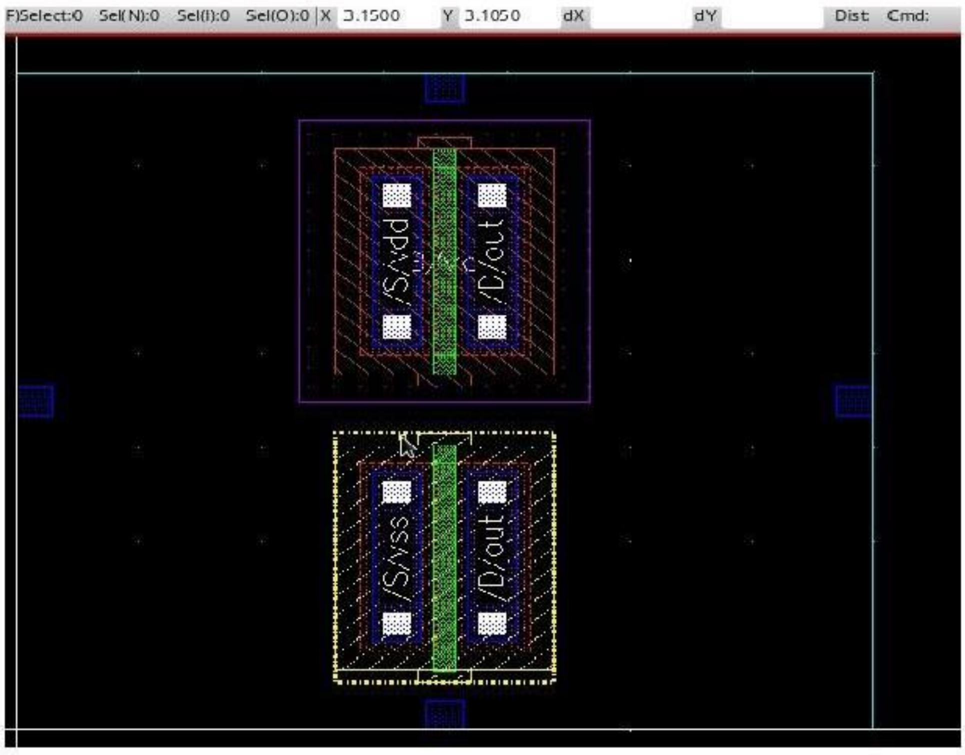
In the layout editor window, **Generate Layout** form appears. Click OK which imports the schematic components in to the Layout window automatically.

2.Re arrange the components with in the PR-Boundary.

3.To rotate a component, Select the component and click **Edit –>Properties**. Now select the degree of rotation from the property edit form.

4.Press ‘ **f** ’ to fit the layout on display.

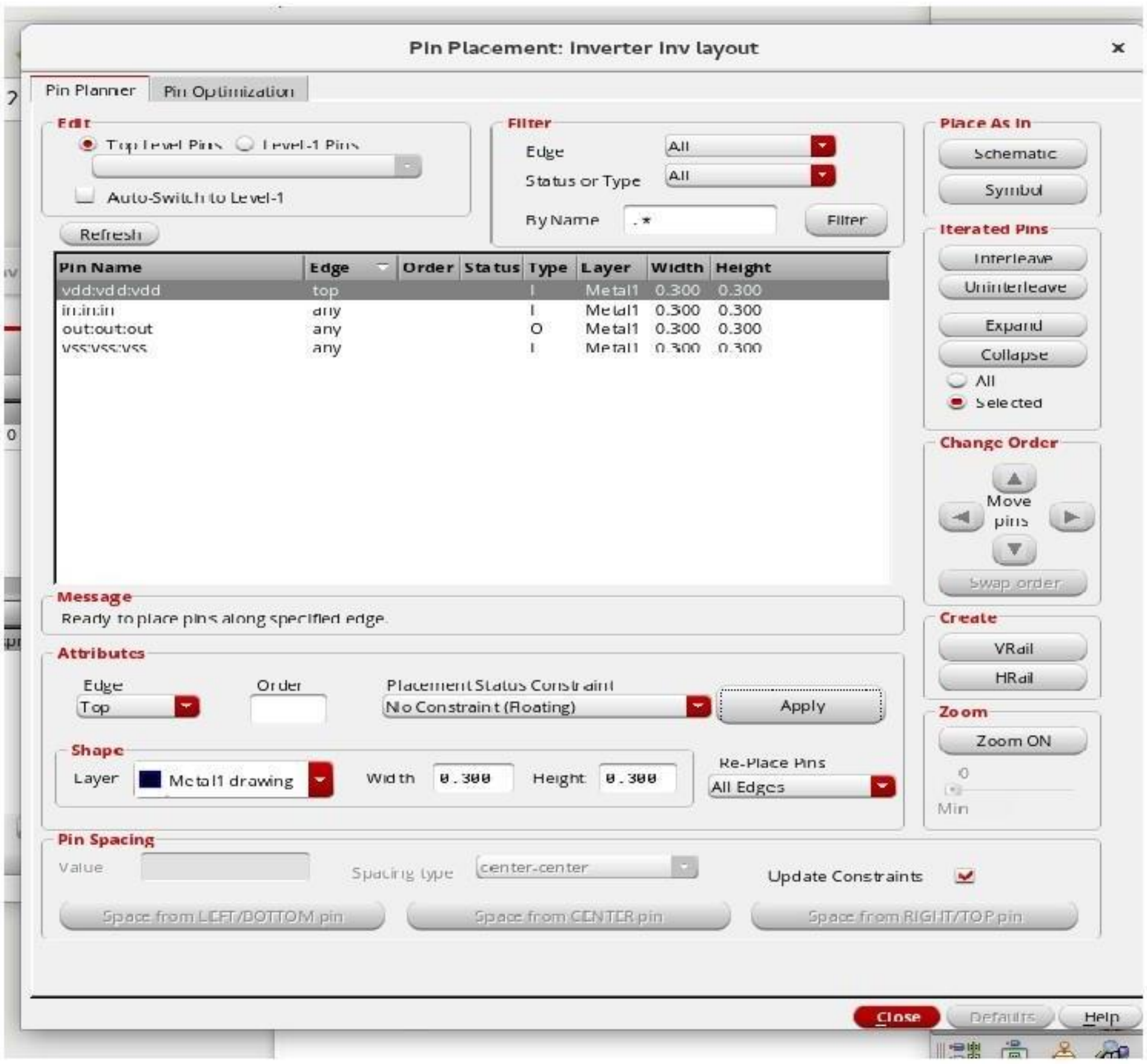
5.Press **‘ Shift+f ’** key, to enable the terminals of a transistor


F)Select:0 Sel(N):0 Sel(I):0 Sel(O):0 X 3.1500 Y 3.1050 dX dY Dist Cmd:
/S/vdd
/D/out
/S/vss
/D/out

6.After enabling all the terminals of a transistor, press ‘P’ in order to make the connectivity.

7.To do HRail for vdd and vss pins,

Right click on the **vdd/vss** pin select ‘**pin placement’** option or select **Place --> Pin Placement** from top menu below window will be pop-up.



Pin Placement: Inverter Inv layout

Edit: [✓ Top Level Pins] Level-1 Pins; Auto-Switch to Level-1

Filter: Edge: All; Status or Type: All; By Name: .*

All [✓ Selected]

Pin Name	Edge	Order	Status	Type	Layer	Width	Height
vdd:vdd:vdd	top			I	Metal1	0.300	0.300
in:in:in	any			I	Metal1	0.300	0.300
out:out:out	any			O	Metal1	0.300	0.300
vss:vss:vss	any			I	Metal1	0.300	0.300

Message: Ready to place pins along specified edge.

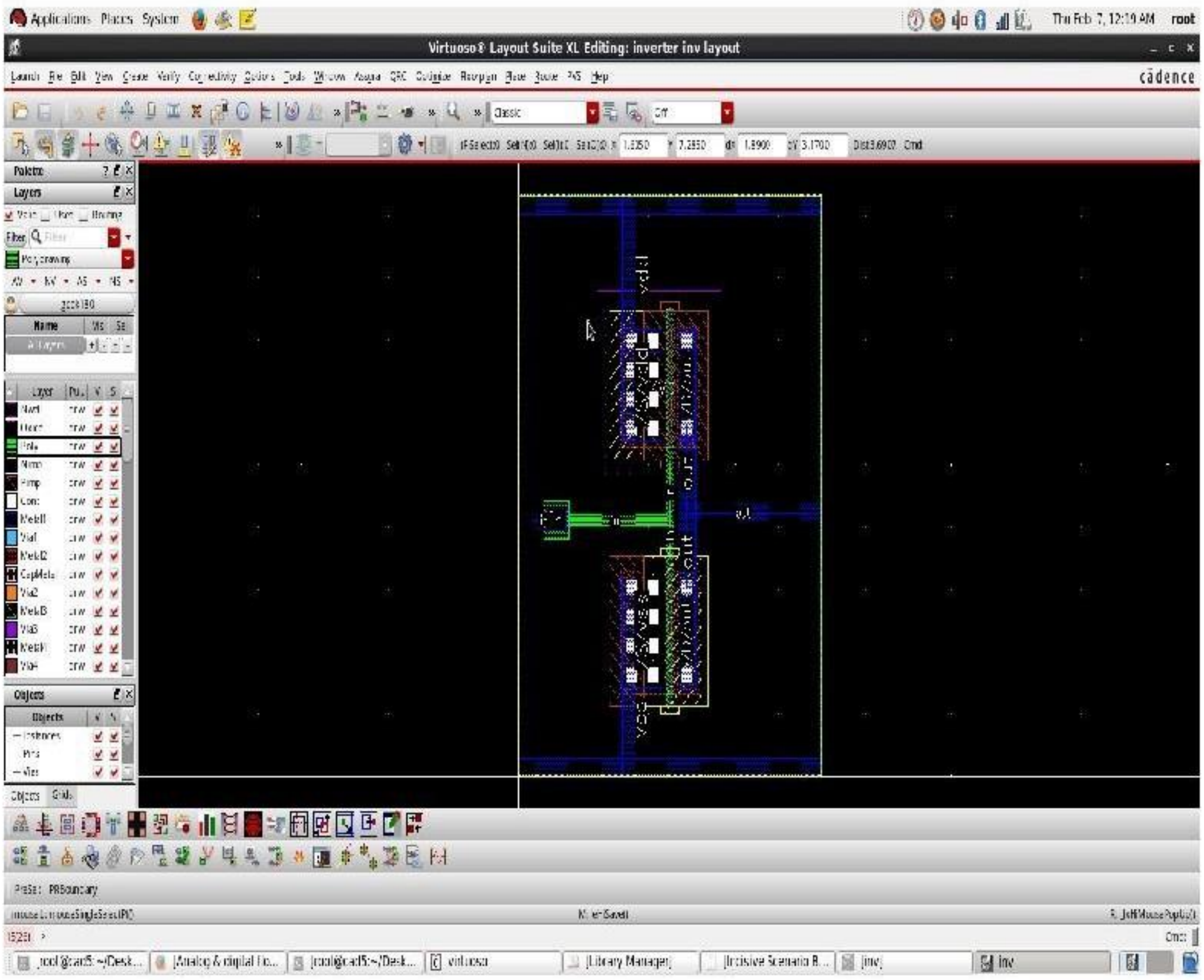
Attributes: Edge: Top; Order: ; Placement Status Constraint: No Constraint (Floating)

Shape: Layer: Metal1 drawing; Width: 0.300; Height: 0.300; Re-Place Pins: All Edges

Pin Spacing: Value: ; Spacing type: center-center; Update Constraints ✓

• Select the necessary pin like **vdd/vss** and select the required **Edge** in **Attributes** session and click on **Apply** option by selecting **HRail** which is available in **Create** session as shown in the image .

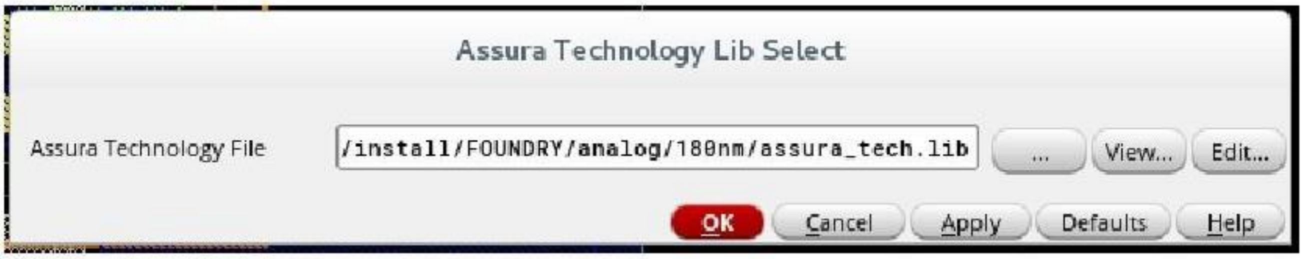
• Make the necessary connections as per the schematic in layout window.



- Save the design by selecting **File —>Save** or click on '**save'** icon.

Physical Verification :

- For doing Physical verification we will go with **ASSURA** tool.
- Add the Assura technology lib file path, Go to **Assura ---> Technology**, then a assura technology lib window will appear. Browse the Assura tech lib.

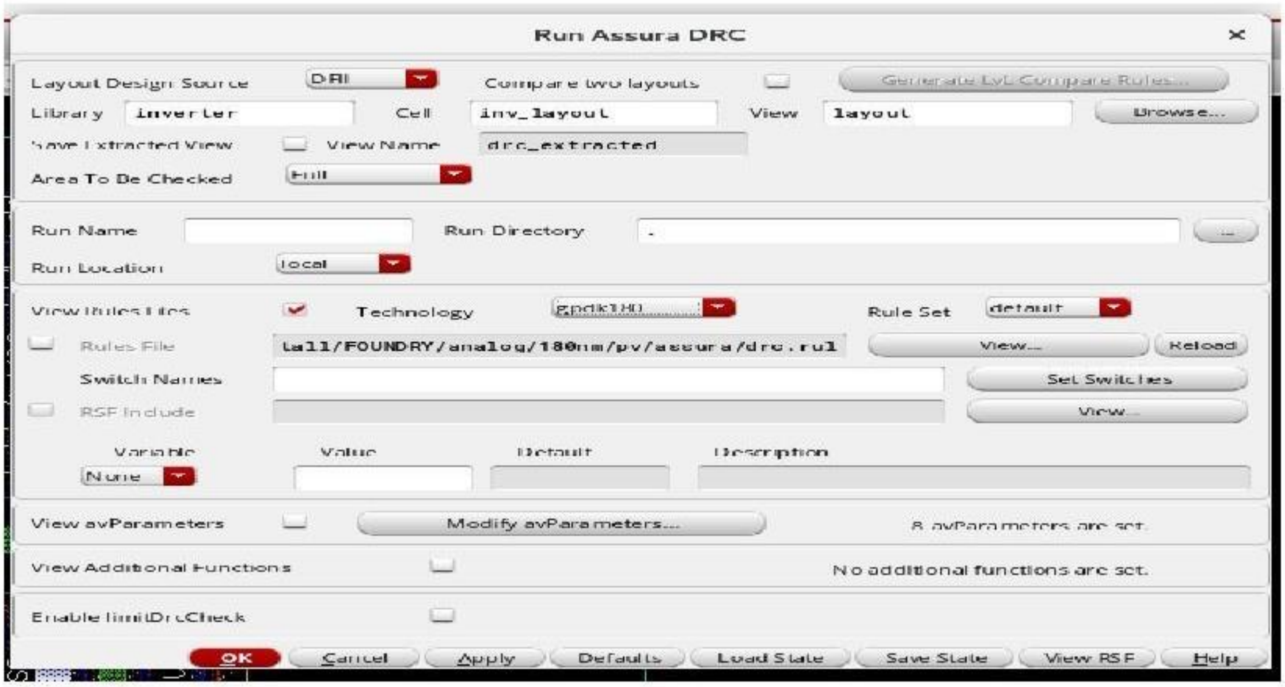


Assura Technology Lib Select

Assura Technology File: /install/FOUNDRY/analog/180nm/assura_tech.lib

Running a DRC:

- Once the Technology library is added Select **Assura --> Run DRC** from layout window.
- The **Run Assura DRC window** will pop up. Now enable **View Rule files**, select the **technology** like gpdk180, 90 or 45 and also provide the **Run Name** while running the DRC as shown in the below figure.



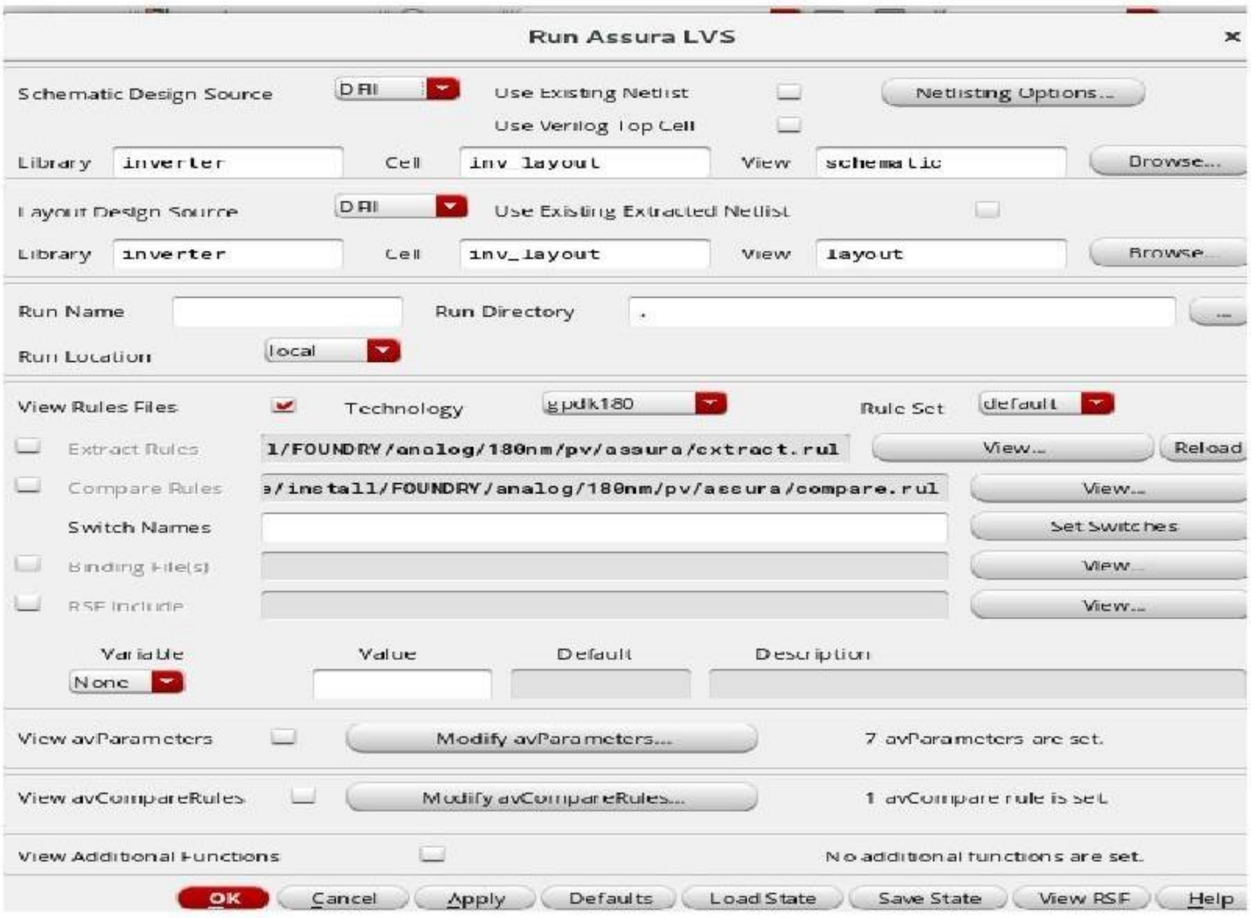
Run Assura DRC

Layout Design Source: DFII
Compare two layouts: ☐
Library: inverter Cell: inv_layout View: layout
Save Extracted View: ☐ View Name: drc_extracted
Area To Be Checked: Full
Run Name: Run Directory: .
Run Location: local
View Rules Files: ✓ Technology: gpdk180 Rule Set: default
☐ Rules File: tall/FOUNDRY/analog/180nm/pv/assura/drc.rul
Switch Names:
☐ RSF Include:
Variable: None Value: Default: Description:
View avParameters: ☐ 8 avParameters are set.
View Additional Functions: ☐ No additional functions are set.
Enable limitDrcCheck: ☐

- Click on Ok and new **Progress File** Window pop up so click on W**atch log file** so log file will appear.
- If there are any DRC errors exist in the design **View Layer Window (VLW)** the **Error Layer Window (ELW) appears** by double clicking on the errors in ELW the errors will be highlighted in the layout window.
- Re-correct all the errors and re-run the DRC again.
- If there are no DRC errors it will pop up with the window showing no **DRC errors.**

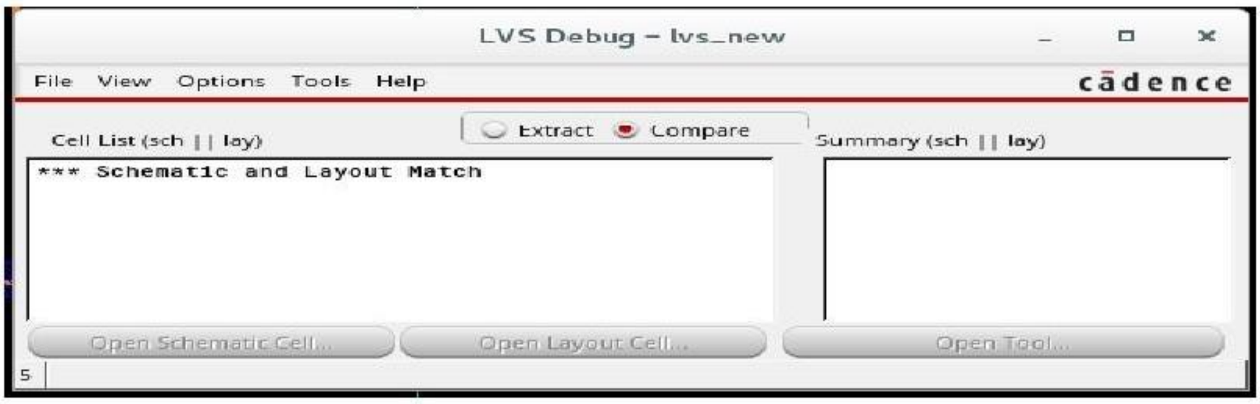
Running LVS:

- **LVS** (Layout versus schematic)
- Select **Assura --> Run LVS** from the layout window to run LVS check.
- Below window will pop up Select the **Technology File** and specify the run name make all the necessary changes as shown in the figure and hit on OK.


Run Assura LVS

Schematic Design Source: DFII
Use Existing Netlist
Use Verilog Top Cell
Library: inverter Cell: inv_layout View: schematic
Layout Design Source: DFII
Use Existing Extracted Netlist
Library: inverter Cell: inv_layout View: layout
Run Name: Run Directory: .
Run Location: local
View Rules Files: ✓ Technology: gpdk180 Rule Set: default
Extract Rules: 1/FOUNDRY/analog/180nm/pv/assura/extract.rul
Compare Rules: s/install/FOUNDRY/analog/180nm/pv/assura/compare.rul
Switch Names:
Binding File(s)
RSF Include
Variable: None Value Default Description
View avParameters — 7 avParameters are set.
View avCompareRules — 1 avCompare rule is set.
View Additional Functions — No additional functions are set.

- Now watch the log file by clicking on **Watch Log file** in Progress File Window.
- If Layout and schematic matches one window will pop and notify schematic and layout match as shown in the figure.



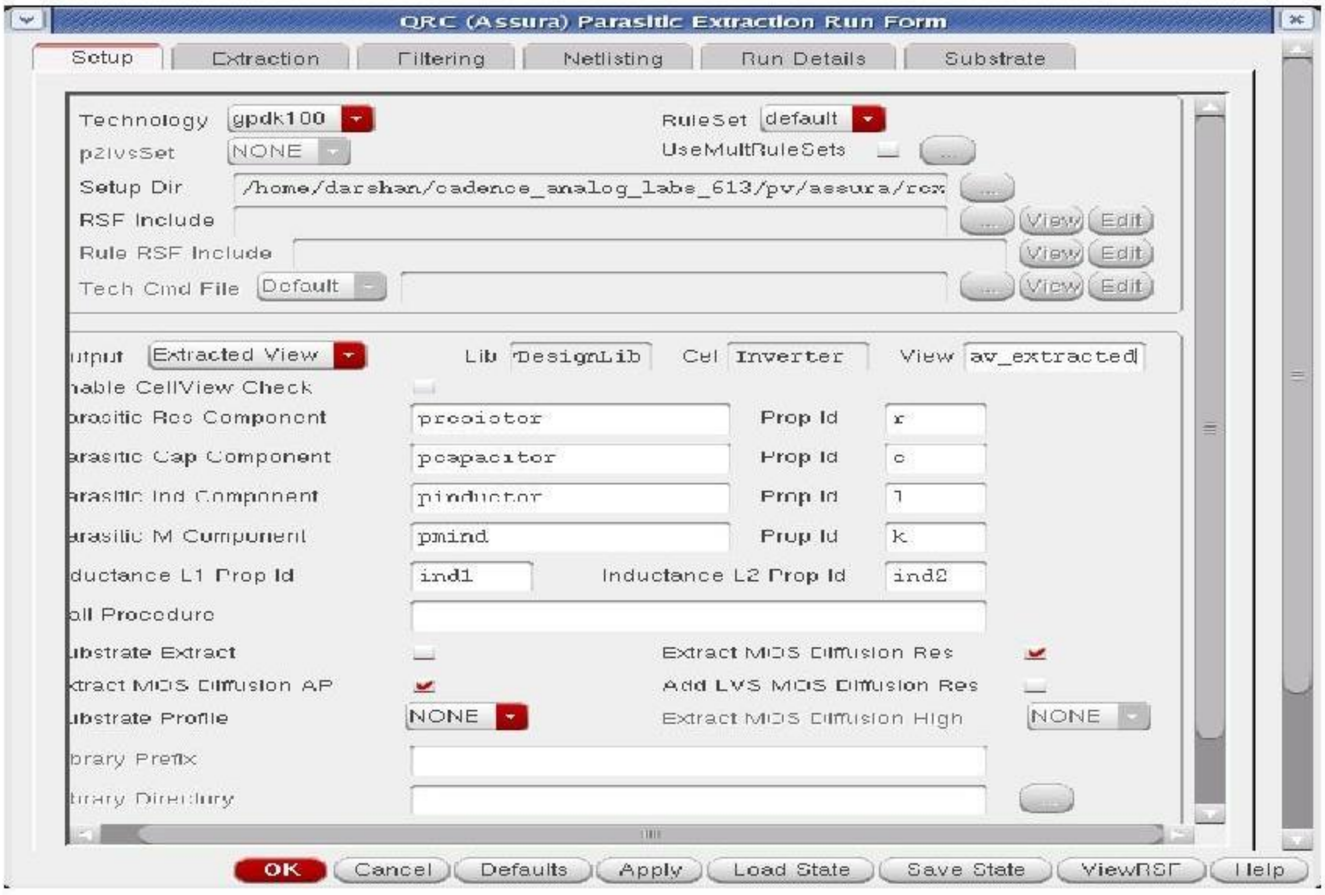
```
*** Schematic and Layout Match
```

If the schematic and layout do not matches, a form informs that the LVS completed successfully and asks if you want to see the results of this run.

- Click **Yes** in the form.
- LVS debug form appears, and you are redirected to LVS debug environment.

In the LVS debug form you can find the details of mismatches and you need to correct all those mismatches and Re – run the LVS till you will be able to match the schematic and layout.

Running Quantus QRC:

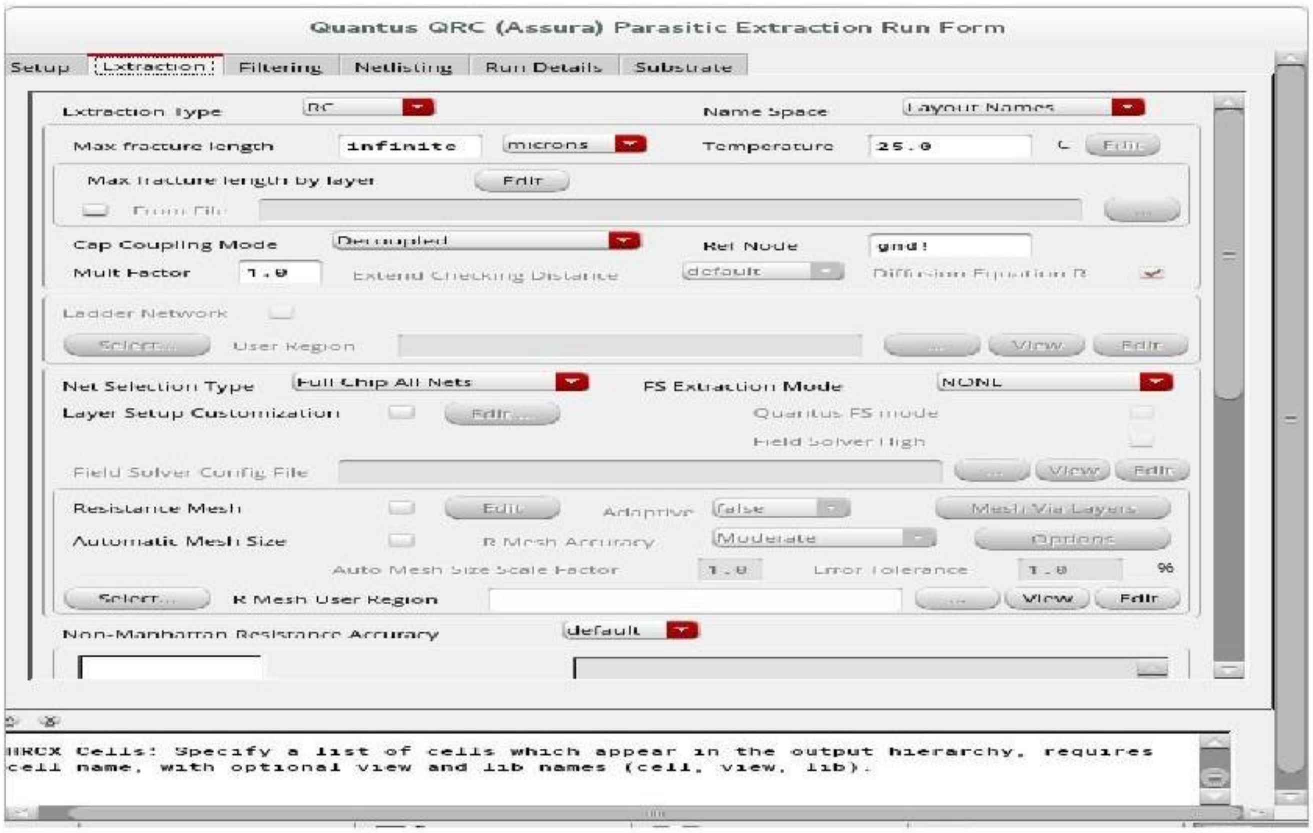

QRC (Assura) Parasitic Extraction Run Form

Technology	gpdk100	RuleSet	default
p2lvsSet	NONE	UseMultRuleSets	
Setup Dir	/home/darshan/cadence_analog_labs_613/pv/assura/rcx		
RSF Include			
Rule RSF Include			
Tech Cmd File	Default		

utput	Extracted View	Lib	DesignLib	Cel	Inverter	View	av_extracted

nable CellView Check			
arasitic Res Component	presistor	Prop Id	r
arasitic Cap Component	pcapacitor	Prop Id	c
arasitic Ind Component	pinductor	Prop Id	l
arasitic M Component	pmind	Prop Id	k
ductance L1 Prop Id	ind1	Inductance L2 Prop Id	ind2
ell Procedure			
ubstrate Extract		Extract MOS Diffusion Res	✓
xtract MOS Diffusion AP	✓	Add LVS MOS Diffusion Res	
ubstrate Profile	NONE	Extract MOS Diffusion High	NONE
brary Prefix			
brary Directory			

- From the layout window, select **Assura --> Run Quantus QRC** for RC extraction.
- Click on **setup** and set the **output** to **Extracted View** and select the **Technology** as shown in the above image**.**
- Click on **Extraction** from the top of the window and give the **Ref Node** as **gnd!** And also select the **Extraction Type** to **RC .**
- Do the necessary changes as shown in the below image and click on **ok**.



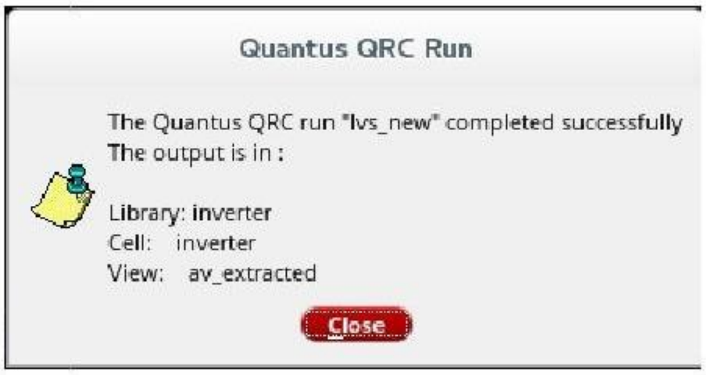
Quantus QRC (Assura) Parasitic Extraction Run Form

Setup | Extraction | Filtering | Netlisting | Run Details | Substrate

Extraction Type: RC
Name Space: Layout Names
Max fracture length: infinite microns
Temperature: 25.0 C
Max fracture length by layer
From File
Cap Coupling Mode: Decoupled
Ref Node: gnd!
Mult Factor: 1.0
Extend Checking Distance: default
Diffusion Equation R
Ladder Network
User Region
Net Selection Type: Full Chip All Nets
FS Extraction Mode: NONE
Layer Setup Customization
Quantus FS mode
Field Solver High
Field Solver Config File
Resistance Mesh
Adaptive: false
Automatic Mesh Size
R Mesh Accuracy: Moderate
Auto Mesh Size Scale Factor: 1.0
Error Tolerance: 1.0 %
R Mesh User Region
Non-Manhattan Resistance Accuracy: default

```
HRCX Cells: Specify a list of cells which appear in the output hierarchy, requires
cell name, with optional view and lib names (cell, view, lib).
```

- The QRC progress will appears, click on **Watch log file** from **Progress File** window.
- When QRC completes, a dialogue box appears, informs you that Quantus QRC run completed successfully or not.
- Open the av_extracted view from the library manager and view the parasitics.

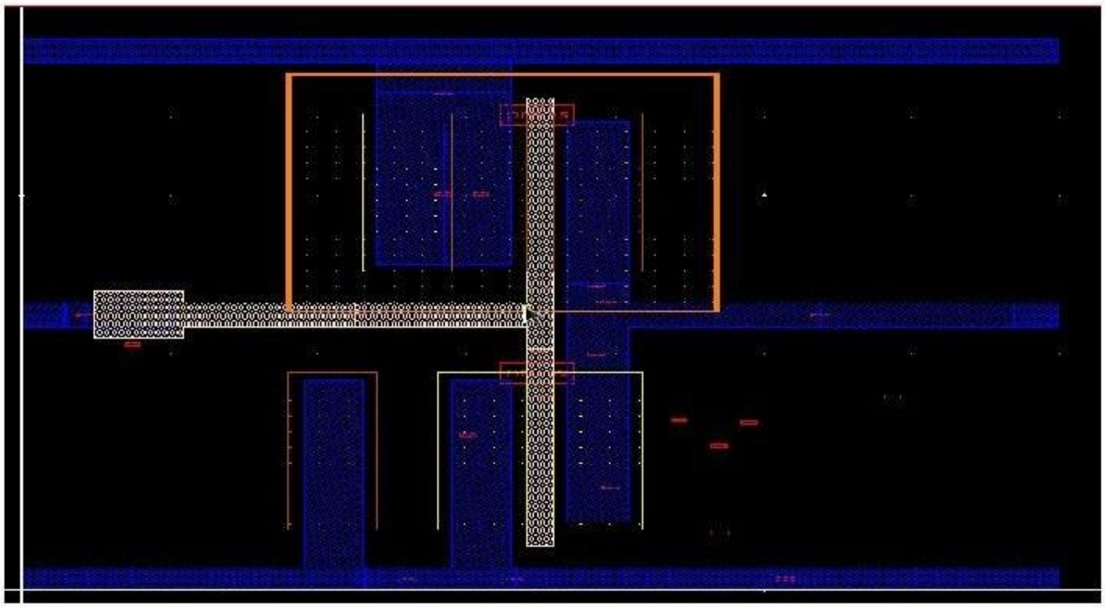


Quantus QRC Run

The Quantus QRC run "lvs_new" completed successfully
The output is in :

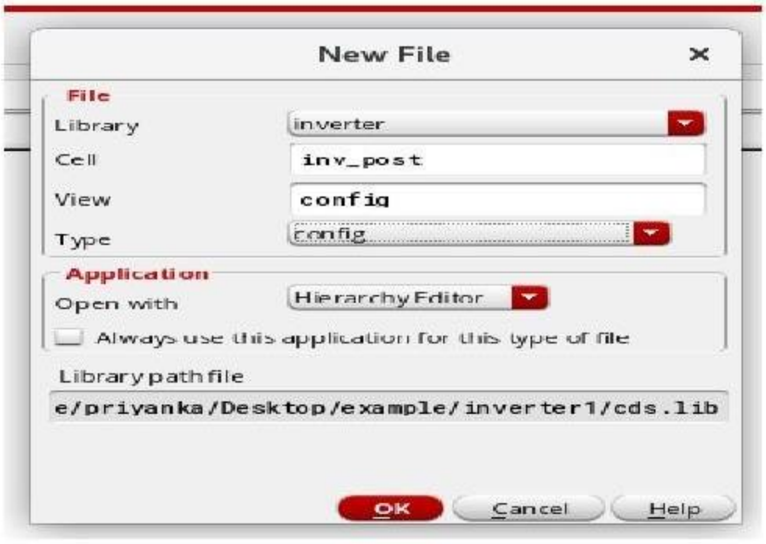
Library: inverter
Cell: inverter
View: av_extracted

- The av_extracted view will appear as shown in the below figure.



To Create the configuration view

- In the CIW or Library Manager, go to File --> New --> Cellview.
- In the Create New file form by setting **View** as **config** as shown in the image and click on ok.



New File

File

Library: inverter

Cell: inv_post

View: config

Type: config

Application

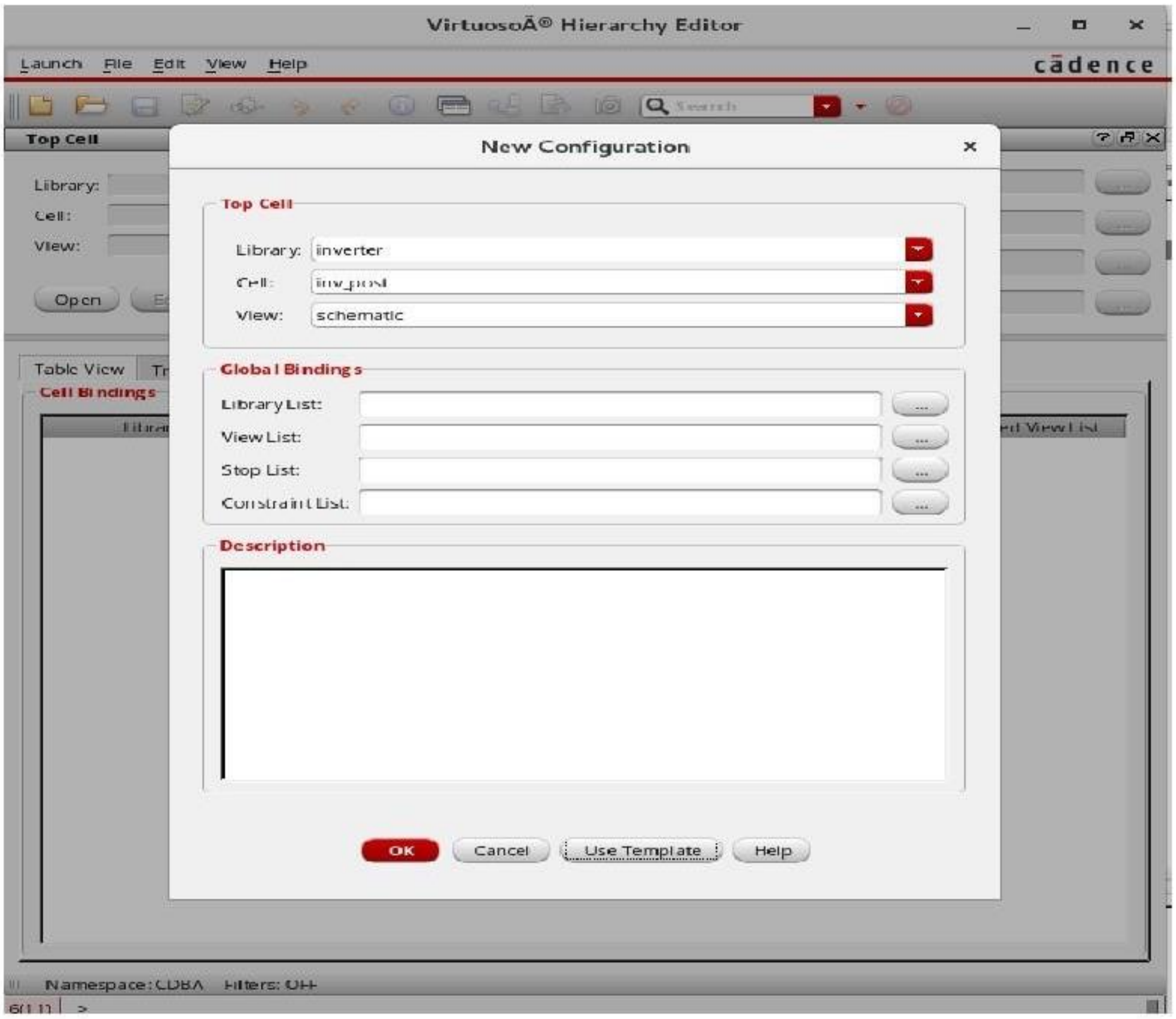
Open with: Hierarchy Editor

Always use this application for this type of file

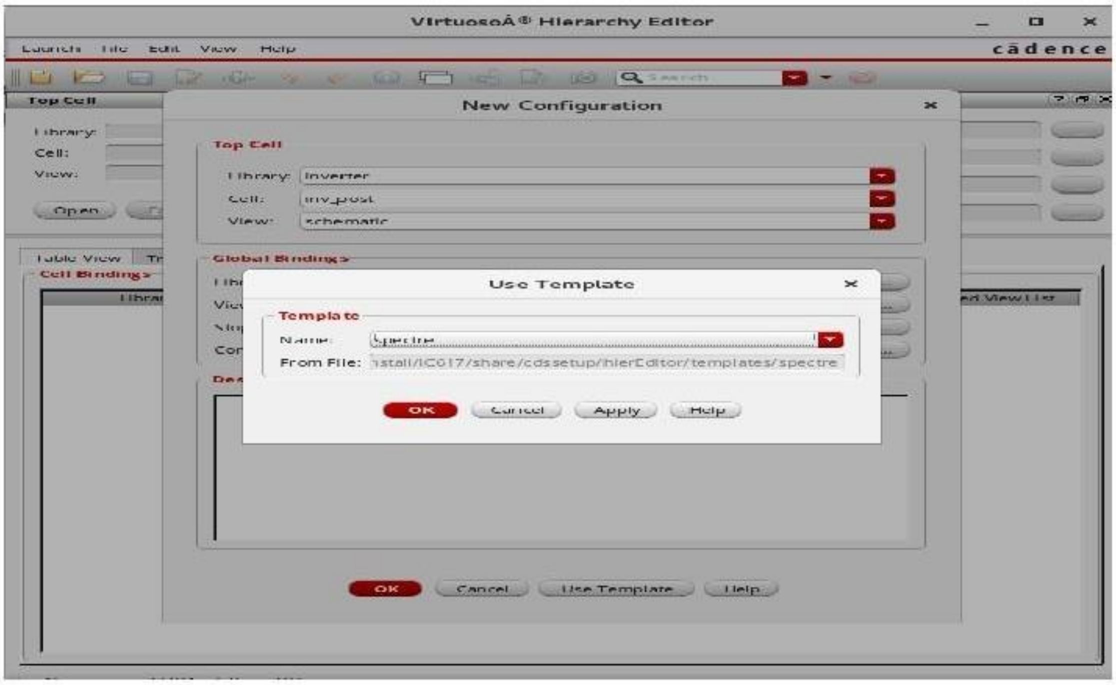
Library path file

e/priyanka/Desktop/example/inverter1/cds.lib

- Now the new window will pop up click on **use template** option which will be on bottom of the New configuration window.



- Select **Spectre** option and click on **ok** as shown in the below image .


Virtuoso® Hierarchy Editor

New Configuration

Top Cell

Library: Inverter

Cell: inv_post

View: schematic

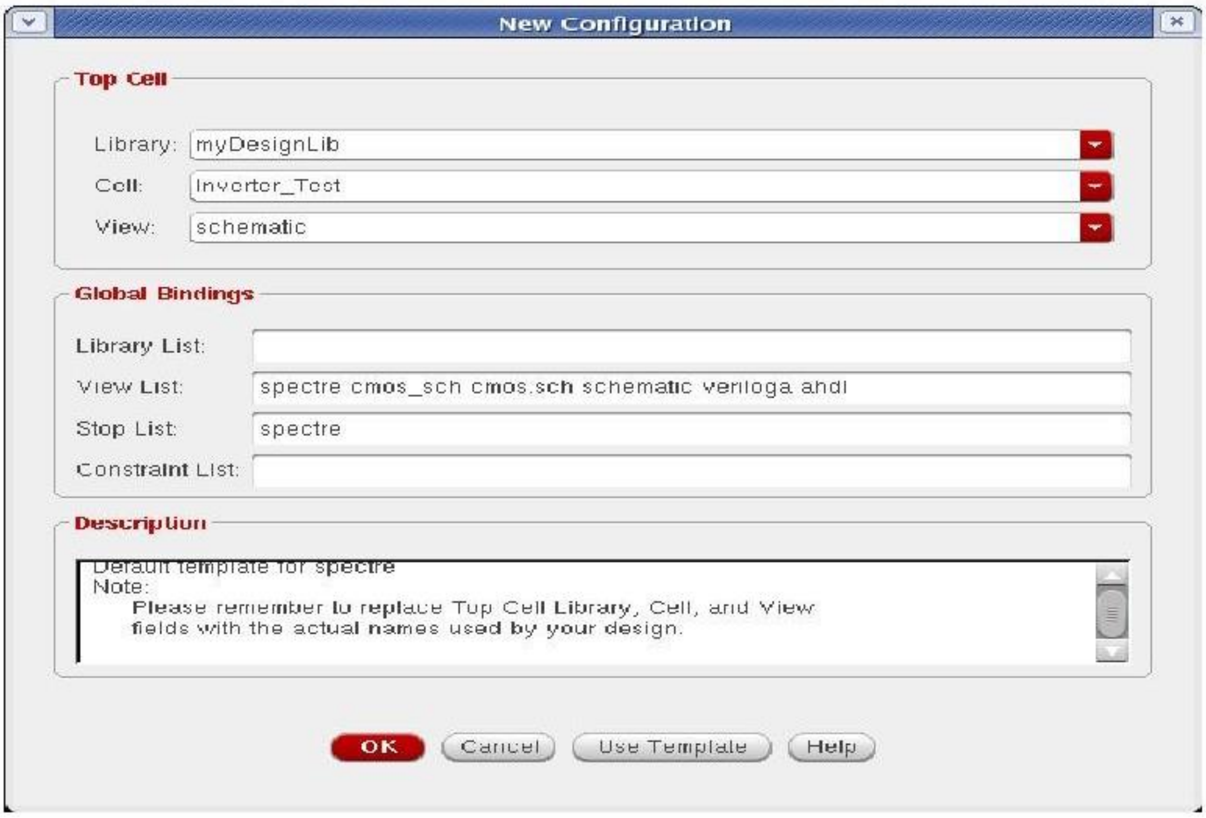
Use Template

Template

Name: spectre

From File: nstall/IC617/share/cdssetup/hierEditor/templates/spectre

- Change the Top Cell View to schematic and remove the default entry from the Library List field and the top cell must be test schematic design.


New Configuration

Top Cell

Library: myDesignLib

Cell: Inverter_Test

View: schematic

Global Bindings

Library List:

View List: spectre cmos_sch cmos.sch schematic veriloga ahdl

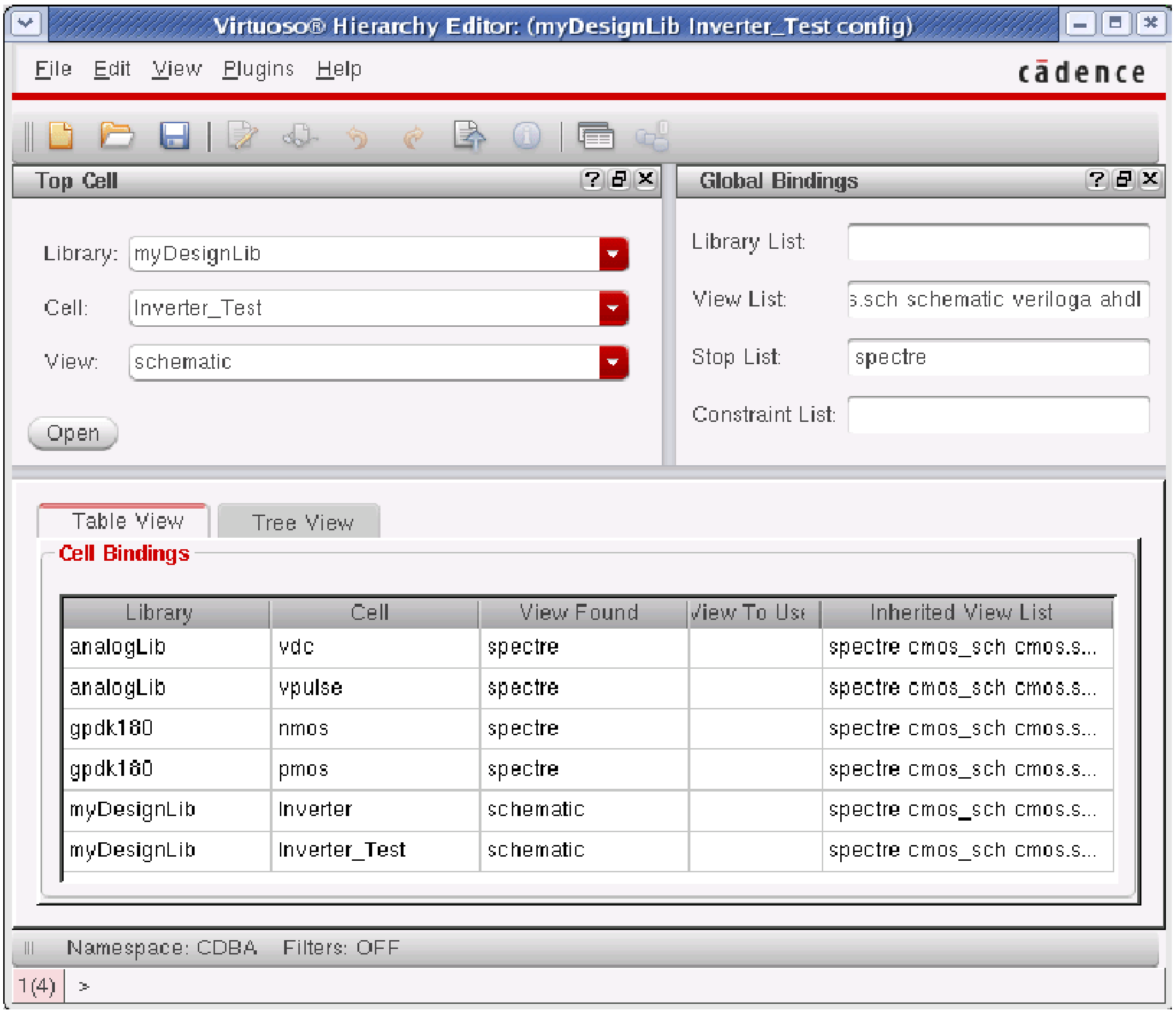
Stop List: spectre

Constraint List:

Description

Default template for spectre
Note:
    Please remember to replace Top Cell Library, Cell, and View
    fields with the actual names used by your design.

- Click on **Ok** by doing the necessary changes as shown in the image.
- The Hierarchy will be **hierarchy window** will be displayed as shown in the below image.


Virtuoso® Hierarchy Editor: (myDesignLib Inverter_Test config)
File Edit View Plugins Help
cādence
Top Cell
Global Bindings
Open
Table View
Tree View
Cell Bindings

Top Cell

Library:	myDesignLib
Cell:	Inverter_Test
View:	schematic

Global Bindings

Library List:	
View List:	s.sch schematic veriloga ahdl
Stop List:	spectre
Constraint List:	

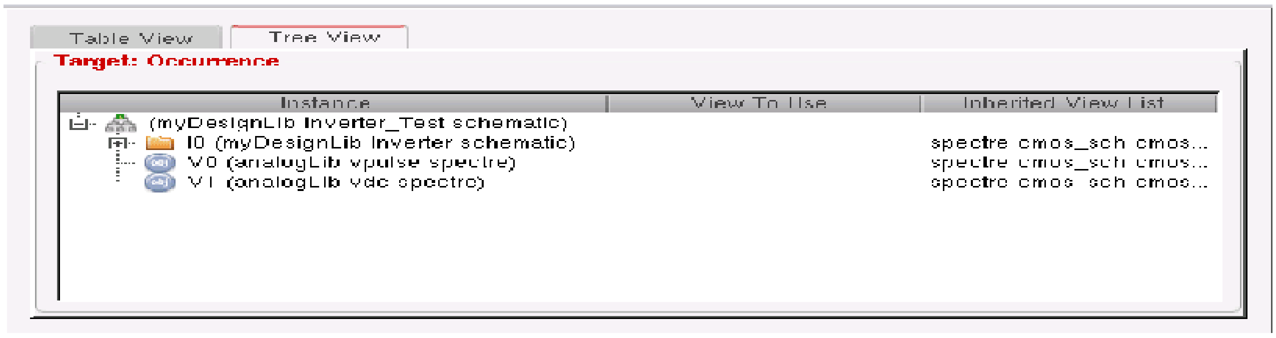
Cell Bindings

Library	Cell	View Found	View To Use	Inherited View List
analogLib	vdc	spectre		spectre cmos_sch cmos.s...
analogLib	vpulse	spectre		spectre cmos_sch cmos.s...
gpdk180	nmos	spectre		spectre cmos_sch cmos.s...
gpdk180	pmos	spectre		spectre cmos_sch cmos.s...
myDesignLib	Inverter	schematic		spectre cmos_sch cmos.s...
myDesignLib	Inverter_Test	schematic		spectre cmos_sch cmos.s...

Namespace: CDBA Filters: OFF

1(4) >

- Click on the Tree view option in the **hierarchy window editor** as shown in the image.



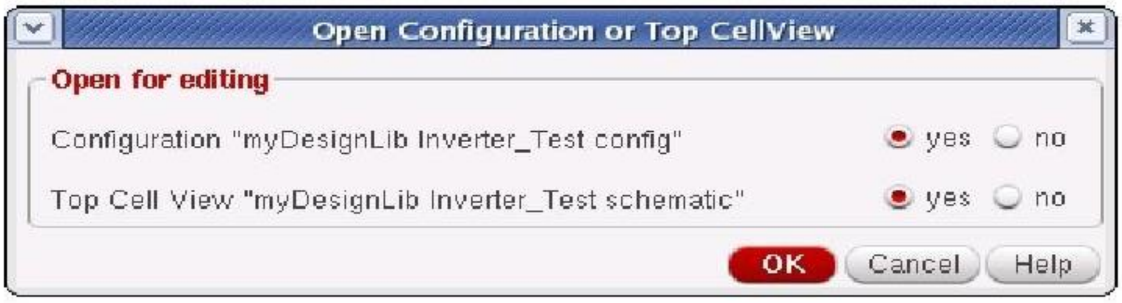
Instance	View To Use	Inherited View List
(myDesignLib Inverter_Test schematic)		
I0 (myDesignLib Inverter schematic)		spectre cmos_sch cmos...
V0 (analogLib vpulse spectre)		spectre cmos_sch cmos...
V1 (analogLib vdc spectre)		spectre cmos_sch cmos...

- Save the current configuration and close the Hierarchy Editor window by click on

File --> Close Window.

To run the Circuit without Parasitics

- From the Library Manager open the test schematic **Config** view. Open Configuration or Top cell view form appears.



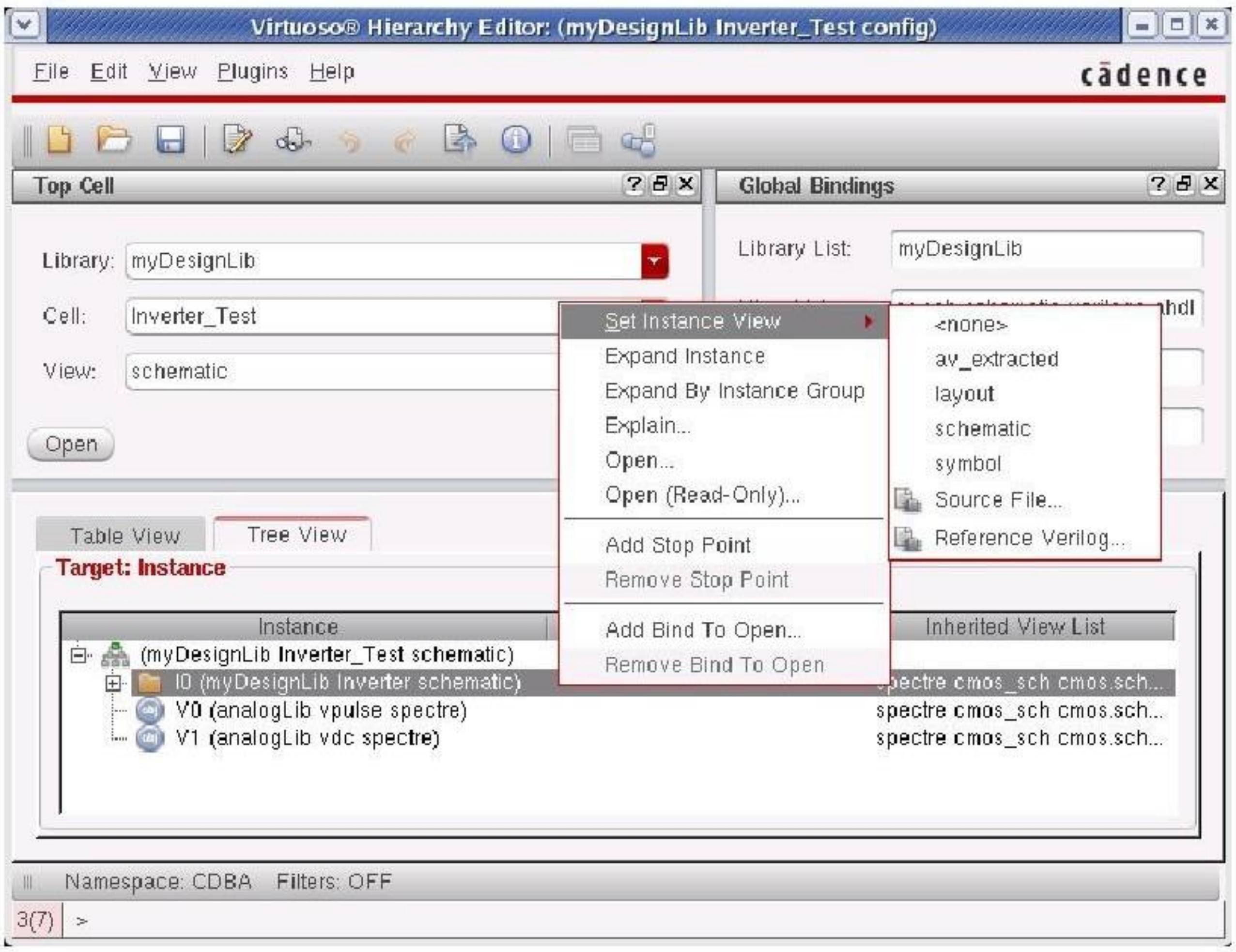
Configuration "myDesignLib Inverter_Test config" ● yes ○ no

Top Cell View "myDesignLib Inverter_Test schematic" ● yes ○ no

- Click on ok by selecting both options to yes as shown in the figure.
- Make sure that the window must open with test schematic design for simulation.
- Execute Launch – ADE L from the **test schematic** window.
- Now you need to follow the same procedure for running the simulation. Executing **Session– Load state**, the Analog Design Environment window loads the previous state.
- Click Netlist and Run icon to start the simulation. The simulation takes a few seconds and then waveform window appears.

To run the Circuit with Parasitics

- Open the same Hierarchy Editor form, which is already set for test schematic config.
- Select the Tree View icon: this will show the design hierarchy in the tree format.
- Click right mouse on the Inverter schematic. A pull down menu appears. Select av_extracted view from the Set Instance view menu, the View to use column now shows av_extracted view.



Library: myDesignLib

Cell: Inverter_Test

View: schematic

Library List: myDesignLib

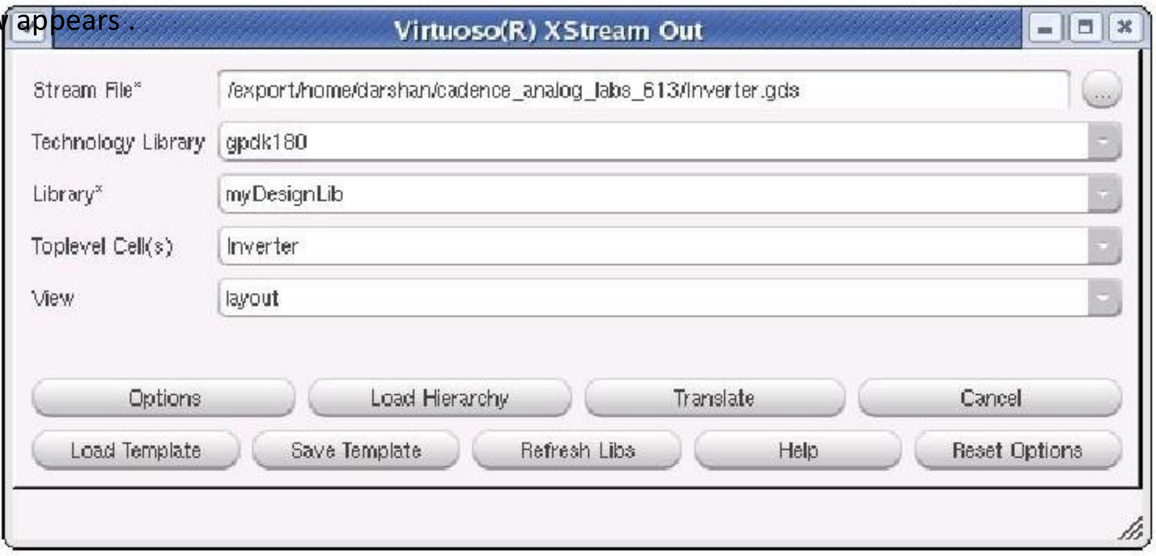
(myDesignLib Inverter_Test schematic)
I0 (myDesignLib Inverter schematic) spectre cmos_sch cmos.sch...
V0 (analogLib vpulse spectre) spectre cmos_sch cmos.sch...
V1 (analogLib vdc spectre) spectre cmos_sch cmos.sch...

- Click on the Recompute the hierarchy icon

- The configuration is now updated from schematic to av_extracted view.
- From the Analog Design Environment window click Netlist and Run to start the simulation again.

Streaming Out the Design

- Select **File – > Export –> Stream** from the CIW menu and Virtuoso **Xstream out** form window appears .



Virtuoso(R) XStream Out

Stream File*	/export/home/darshan/cadence_analog_labs_613/Inverter.gds
Technology Library	gpdk180
Library*	myDesignLib
Toplevel Cell(s)	Inverter
View	layout

- Click on the Options button.
- In the Stream Out-Options form select **Automatic mapping** and click OK.
- In the Virtuoso **Stream Out** form, click Translate button to start the stream translator.
- The stream file is stored in the specified location with **.gds** format.
- From the Library Manager open the Inverter cell view from the GDS_LIB library and notice the design.

LAB 2 :Common Source Amplifier

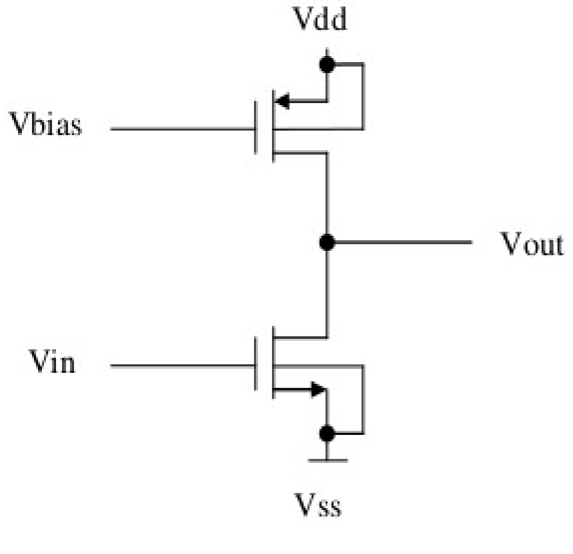
AIM : To simulate the schematic of the common source amplifier.

Theory :

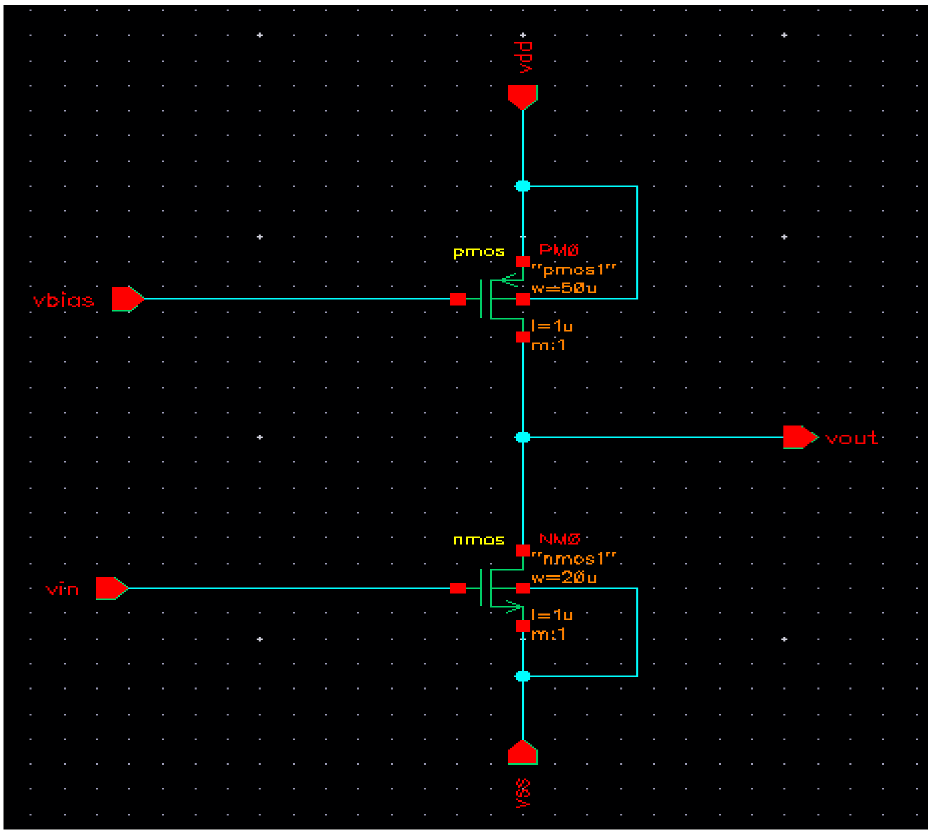
In electronics, a common-source amplifier is one of three basic single-stage field-effect transistor (FET) amplifier topologies, typically used as a voltage or transconductance amplifier. The easiest way to tell if a FET is common source, common drain, or common gate is to examine where the signal enters and leaves. The remaining terminal is what is known as "common". In this example, the signal enters the gate, and exits the drain. The only terminal remaining is the source. This is a common-source FET circuit. The analogous bipolar junction transistor circuit is the common-emitter amplifier.

The common-source (CS) amplifier may be viewed as a transconductance amplifier or as a voltage amplifier. (See classification of amplifiers). As a transconductance amplifier, the input voltage is seen as modulating the current going to the load. As a voltage amplifier, input voltage modulates the amount of current flowing through the FET, changing the voltage across the output resistance according to Ohm's law. However, the FET device's output resistance typically is not high enough for a reasonable transconductance amplifier (ideally infinite), nor low enough for a decent voltage amplifier (ideally zero). Another major drawback is the amplifier's limited high-frequency response.

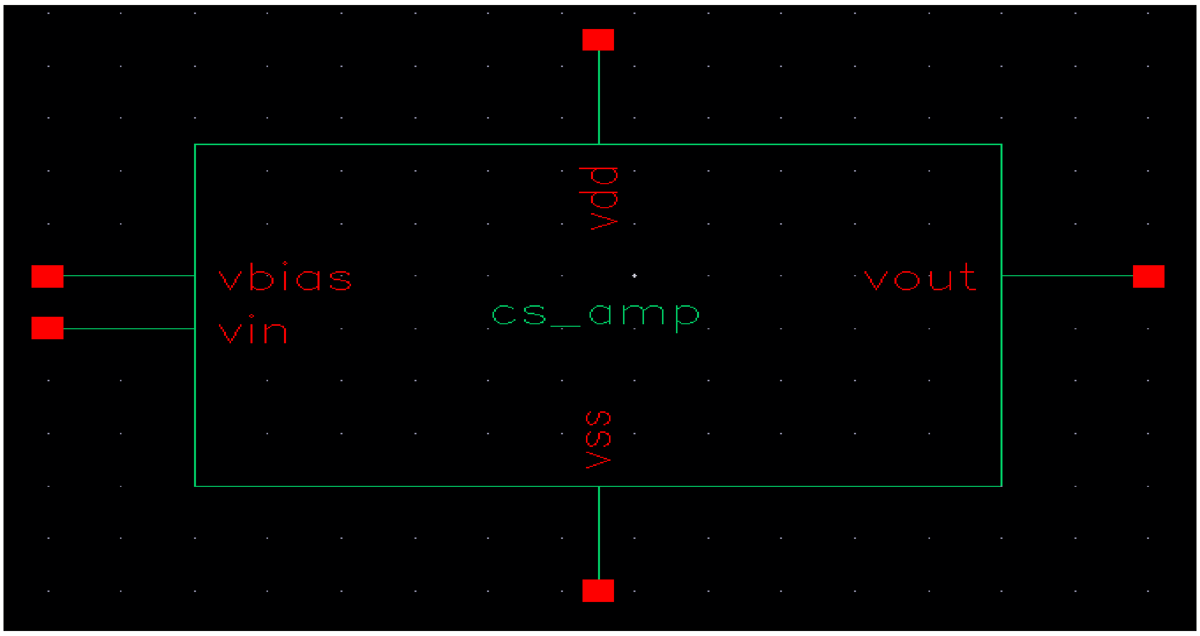
Circuit Diagram :



Schematic Entry :



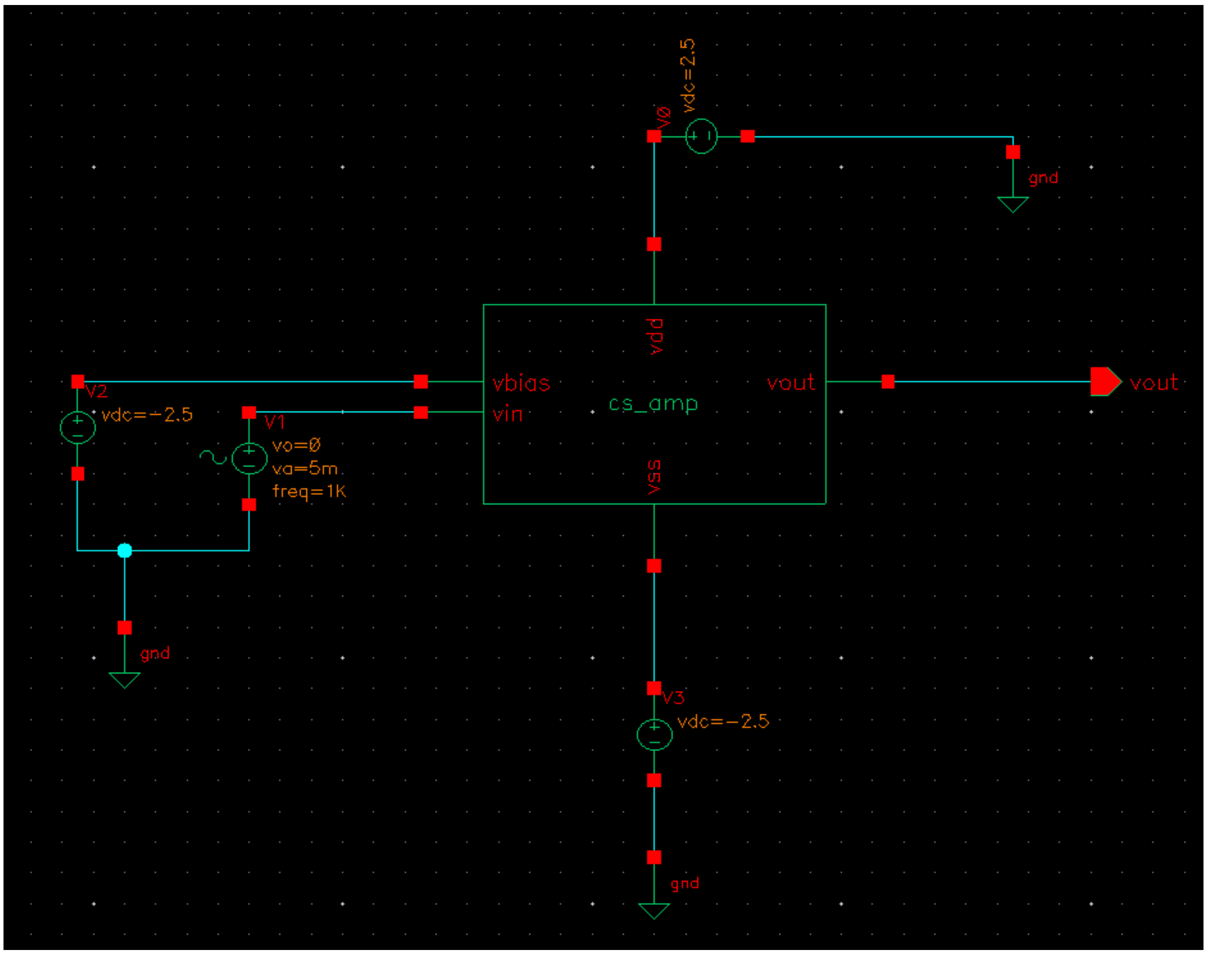
Symbol Creation :



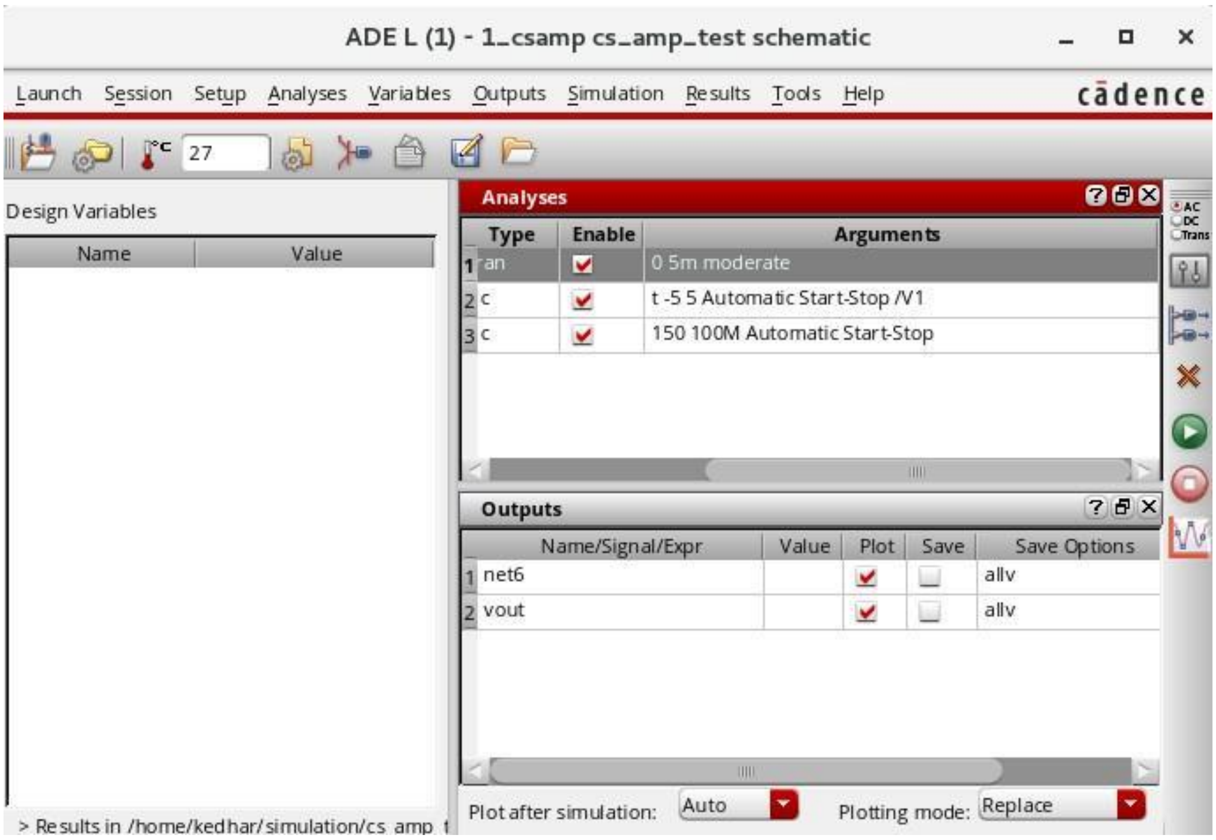
Design entry for Test Schematic :

Library name	Cellview name	Properties
myDesignLib	cs_amplifier	Symbol
analogLib	vsin	Define pulse specification as AC Magnitude=1; DC Voltage=0; Offset Voltage=0; Amplitude=5m; Frequency=1K
analogLib	vdd,vss,gnd	Vdd=2.5;vss=-2.5;vbias=-2.5

Test Schematic Circuit :



Settings in ADE L window :



ADE L (1) - 1_csamp cs_amp_test schematic

Design Variables

Name	Value

Analyses

	Type	Enable	Arguments
1	ran	✓	0 5m moderate
2	c	✓	t -5 5 Automatic Start-Stop /V1
3	c	✓	150 100M Automatic Start-Stop

Outputs

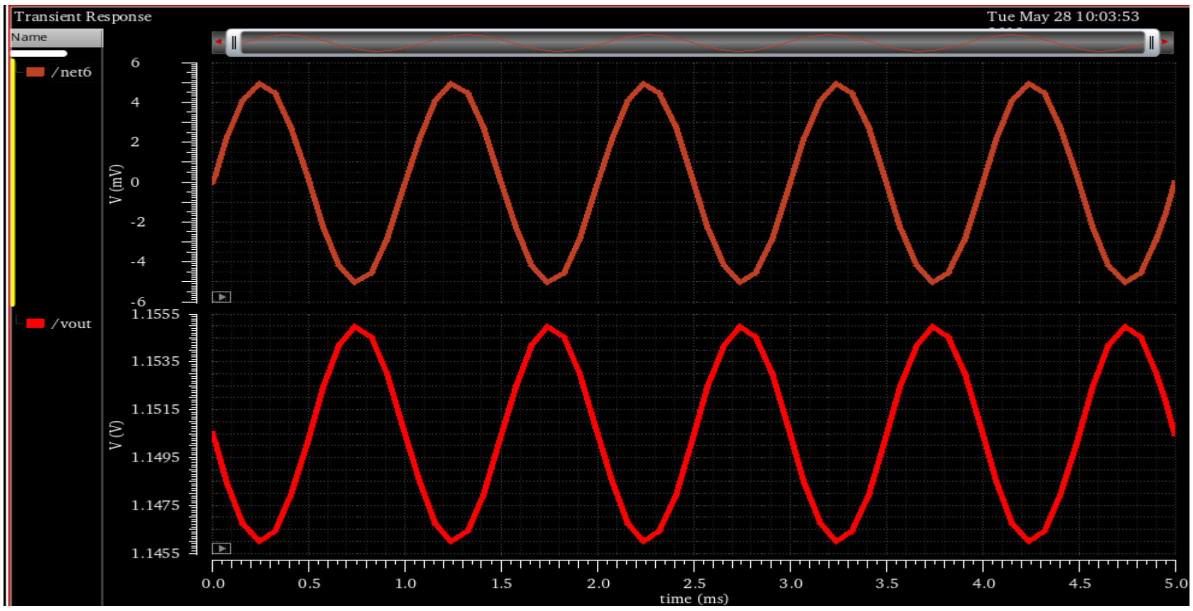
	Name/Signal/Expr	Value	Plot	Save	Save Options
1	net6		✓		allv
2	vout		✓		allv

Plot after simulation: Auto   Plotting mode: Replace

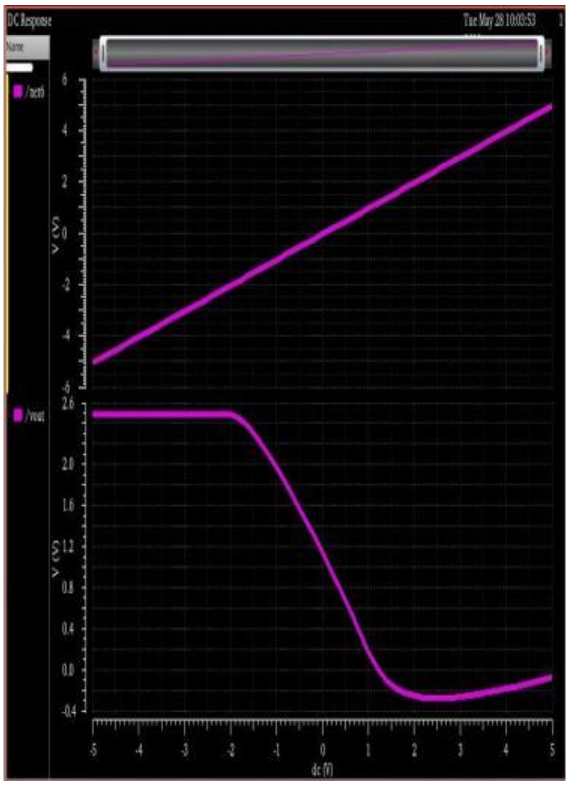
> Results in /home/kedhar/simulation/cs amp

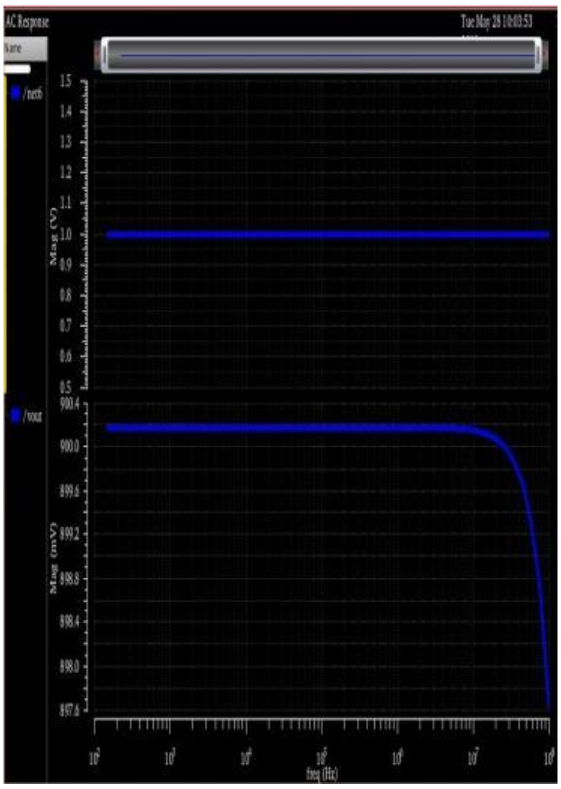
Simulation Results :

Transient Response :

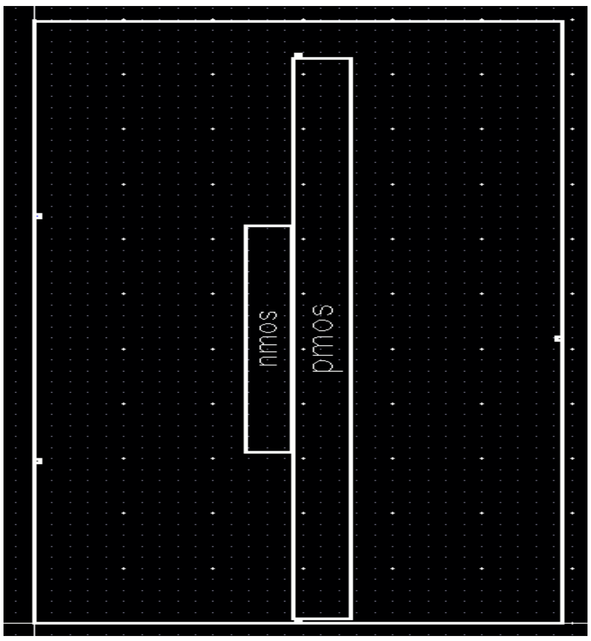


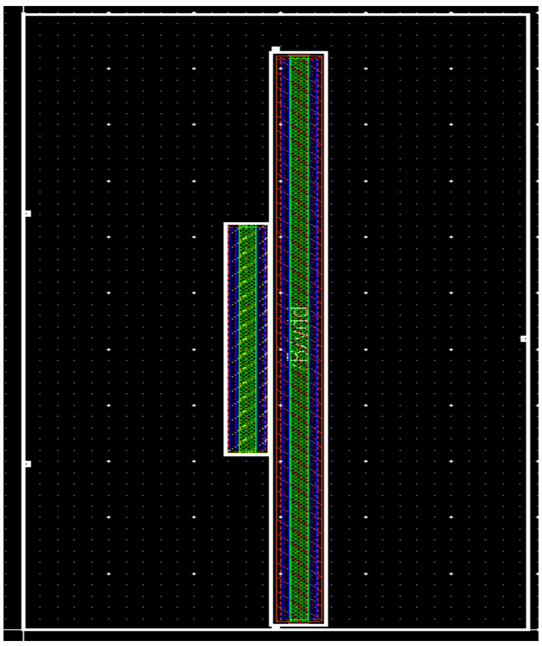
DC and AC Response :

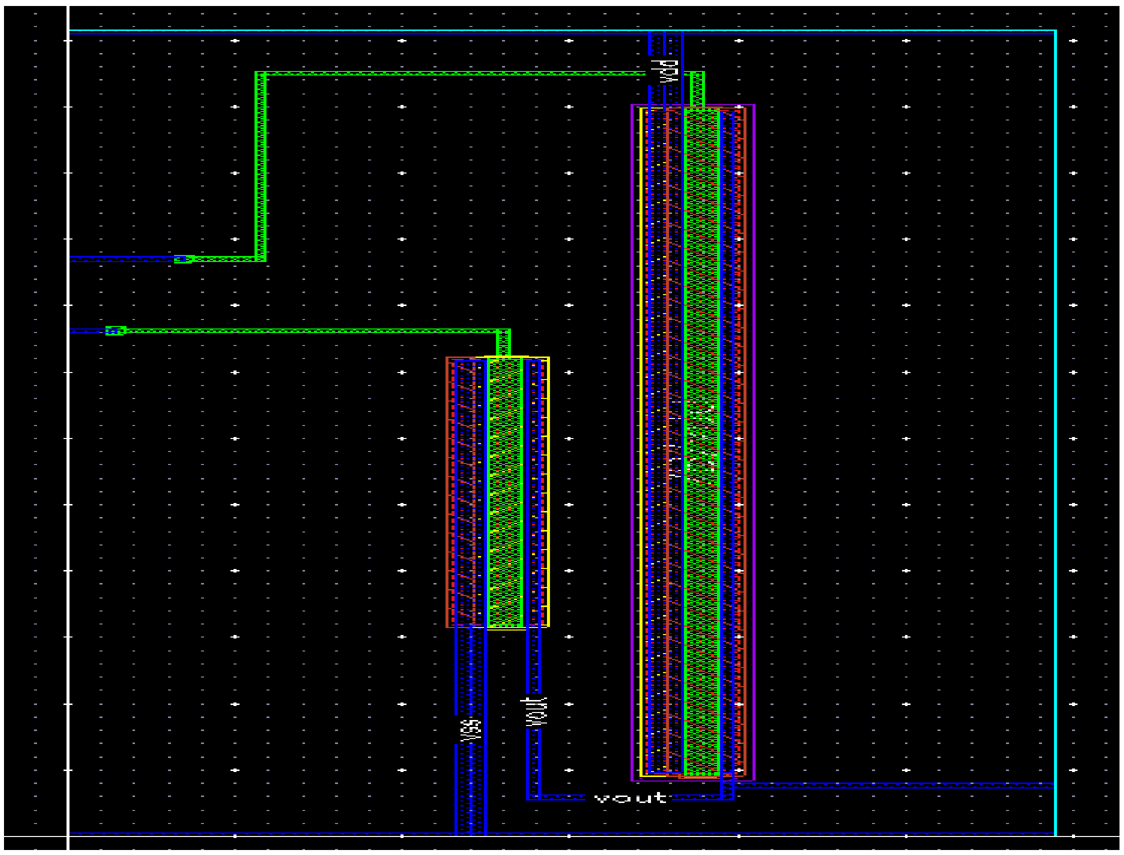




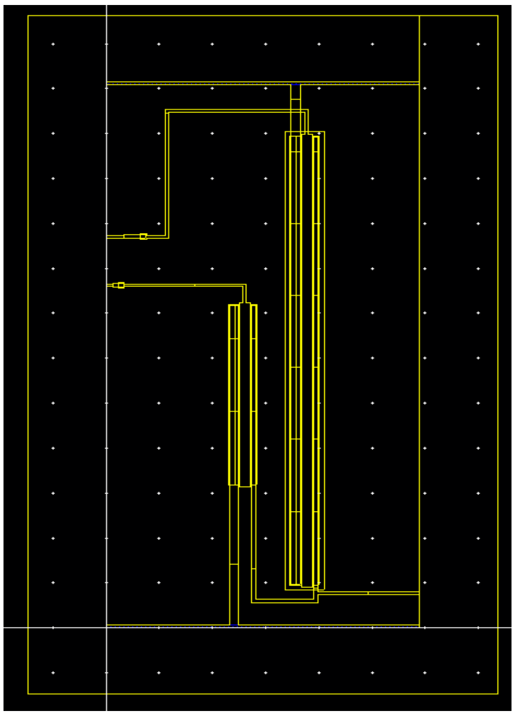
Layout :

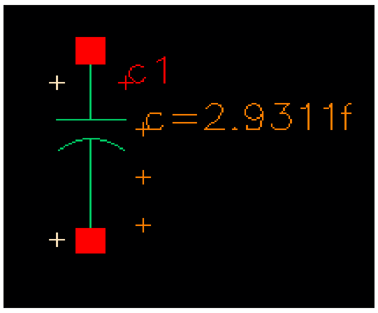


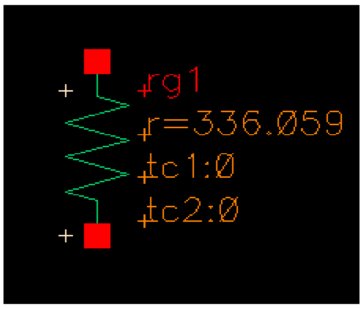




Extracted View :







LAB 3 : Common Drain Amplifier

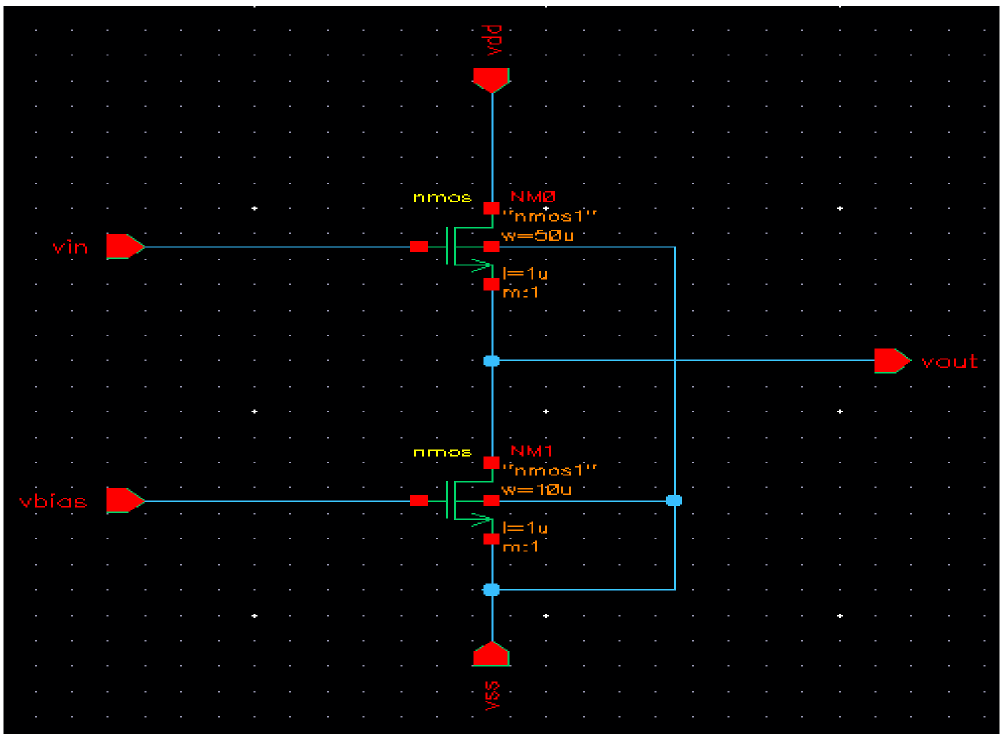
AIM : To simulate the schematic of the common drain amplifier and then to perform the physical verification for the layout of the design.

Theory :

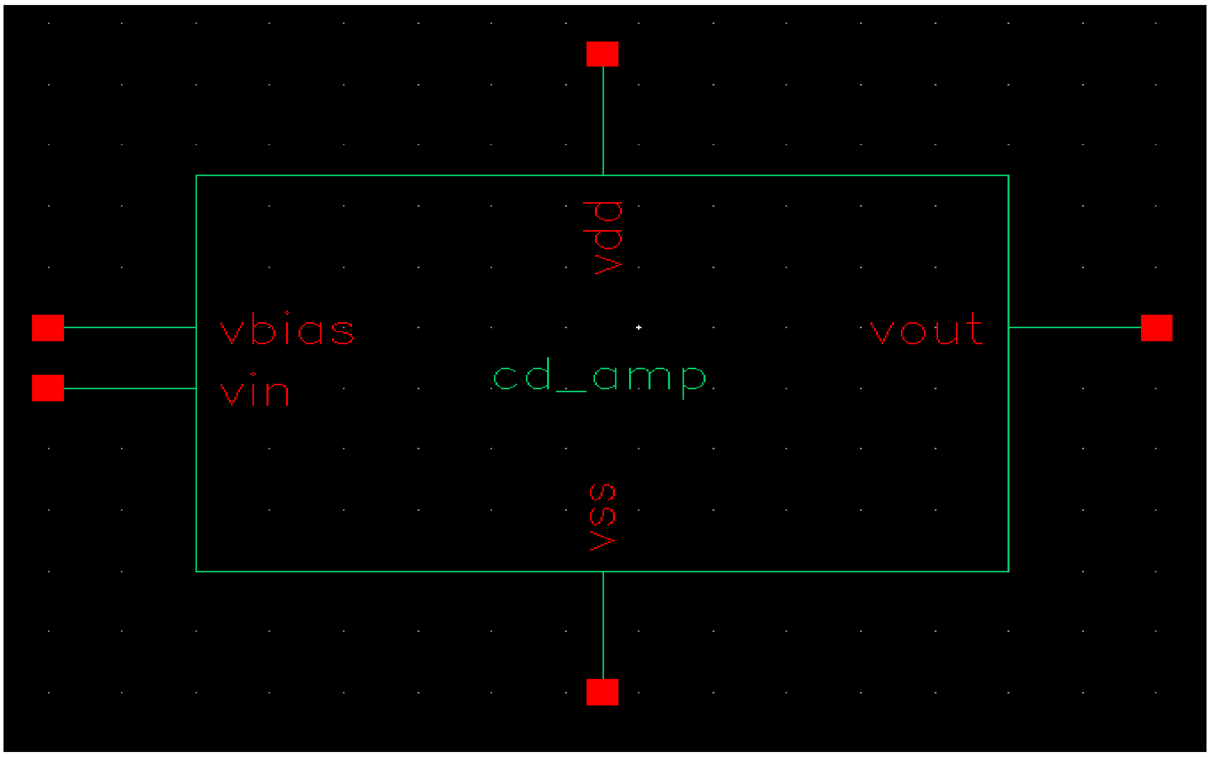
Common drain amplifier is a source follower or buffer amplifier circuit using a MOSFET. The output is simply equal to the input minus about 2.2V. The advantage of this circuit is that the MOSFET can provide current and power gain; the MOSFET draws no current from the input. It provides low output impedance to any circuit using the output of the follower, meaning that the output will not drop under load.

Its output impedance is not as low as that of an emitter follower using a bipolar transistor (as you can verify by connecting a resistor from the output to -15V), but it has the advantage that the input impedance is infinite. The MOSFET is in saturation, so the current across it is determined by the gate-source voltage. Since a current source keeps the current constant, the gate-source voltage is also constant.

Schematic Entry :



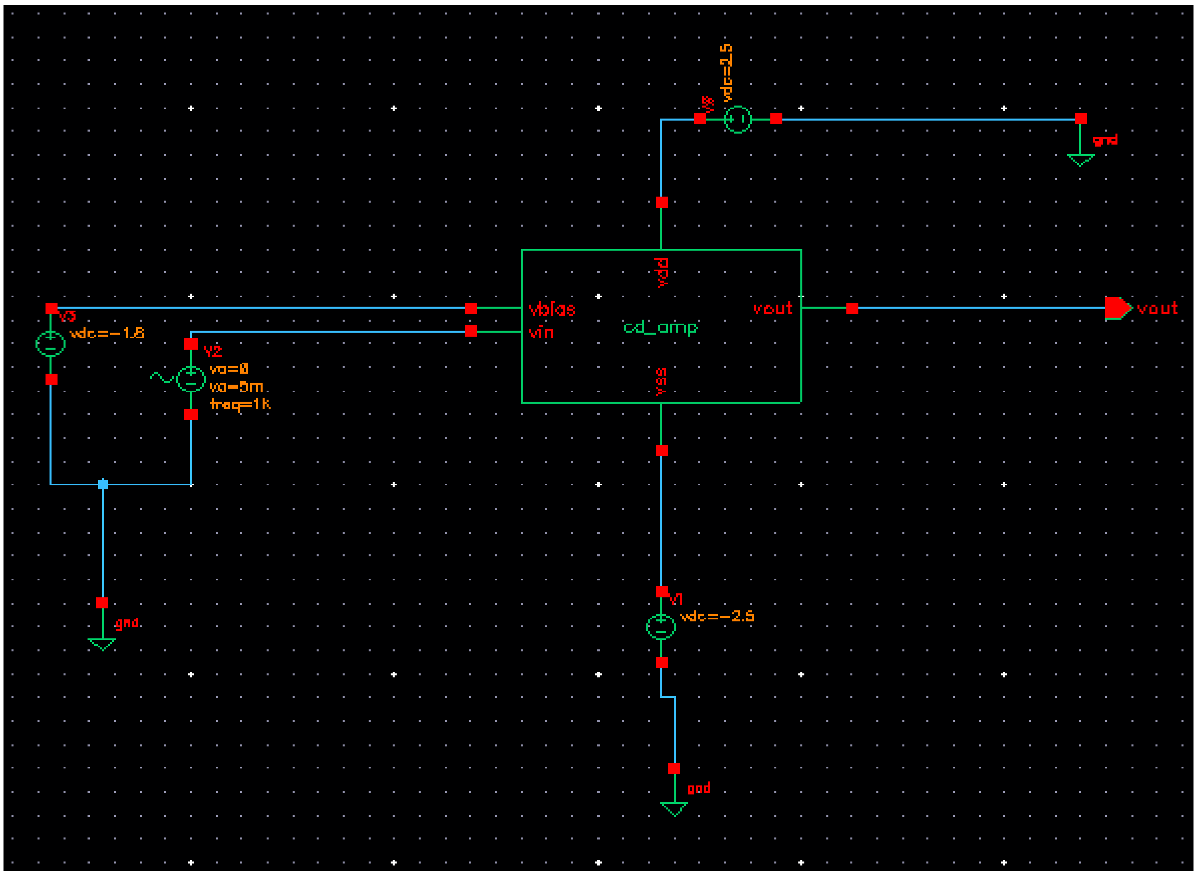
Symbol Creation :



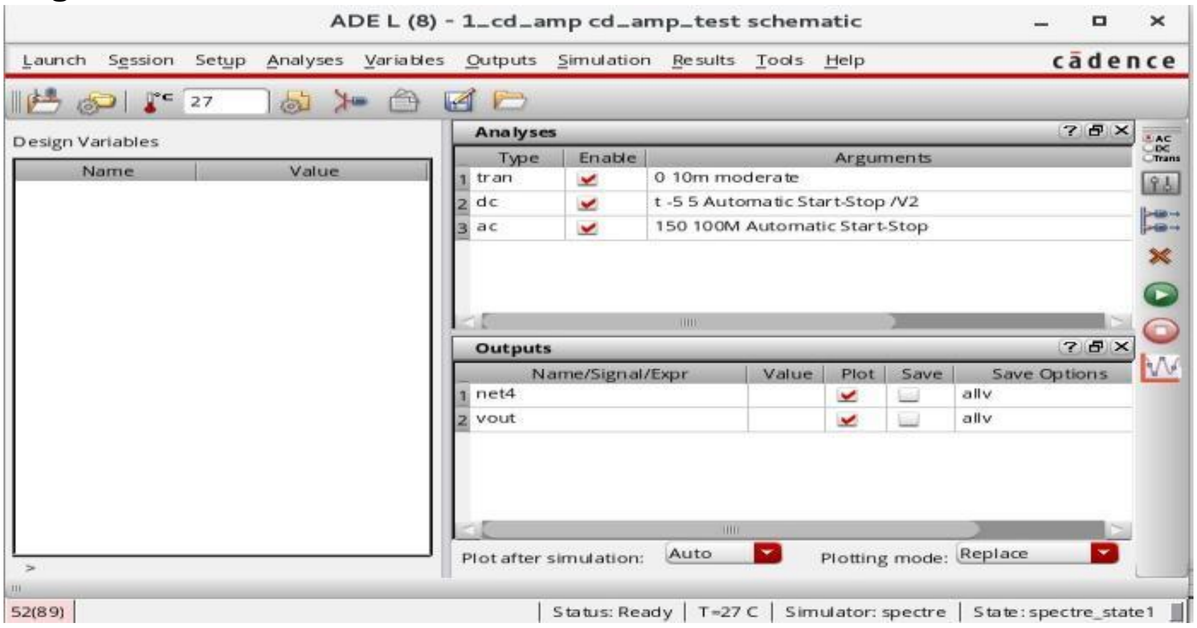
Design entry for test circuit :

Library name	**Cellview name**	**Properties**
myDesignLib	cd_amplifier	Symbol
analogLib	vsin	Define pulse specification as AC Magnitude=1; DC Voltage=0; Offset Voltage=0; Amplitude=5m; Frequency=1K
analogLib	vdd,vss,gnd	Vdd=2.5;vss=-2.5;

Test Schematic :



Settings in ADE L Window :



ADE L (8) - 1_cd_amp cd_amp_test schematic
Launch Session Setup Analyses Variables Outputs Simulation Results Tools Help
cadence
27

Design Variables

Name	Value

Analyses

	Type	Enable	Arguments
1	tran	✓	0 10m moderate
2	dc	✓	t -5 5 Automatic Start-Stop /V2
3	ac	✓	150 100M Automatic Start-Stop

Outputs

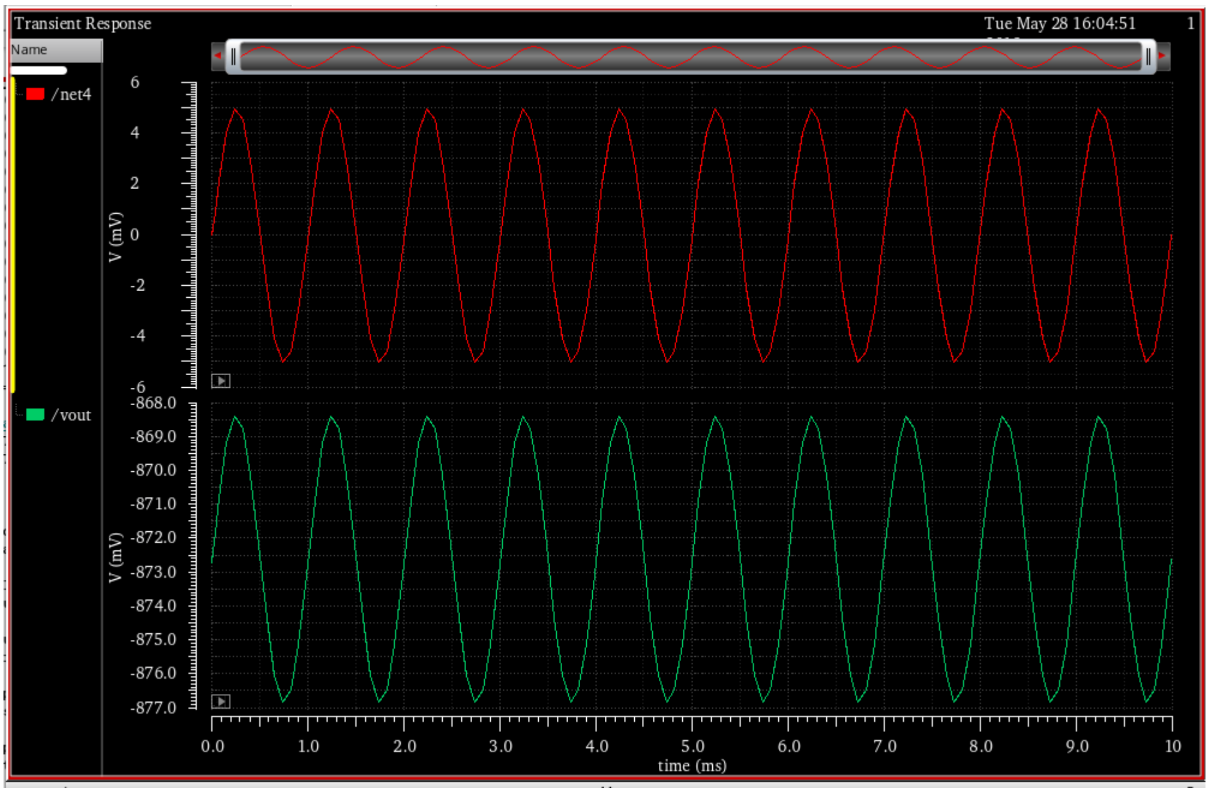
	Name/Signal/Expr	Value	Plot	Save	Save Options
1	net4		✓	☐	allv
2	vout		✓	☐	allv

Plot after simulation: Auto Plotting mode: Replace

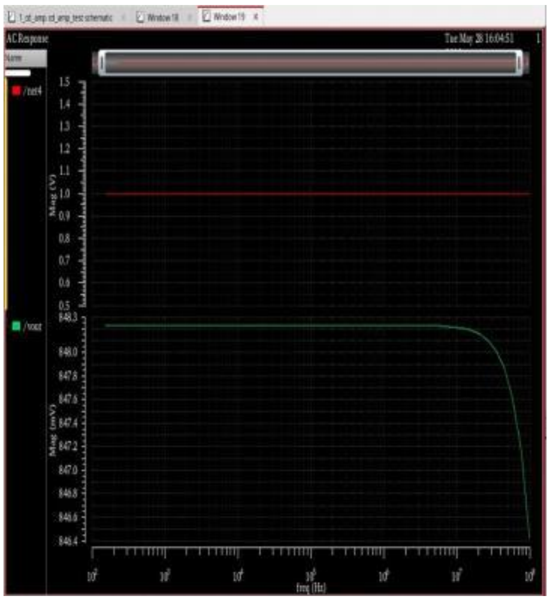
52(89) | Status: Ready | T=27 C | Simulator: spectre | State: spectre_state1

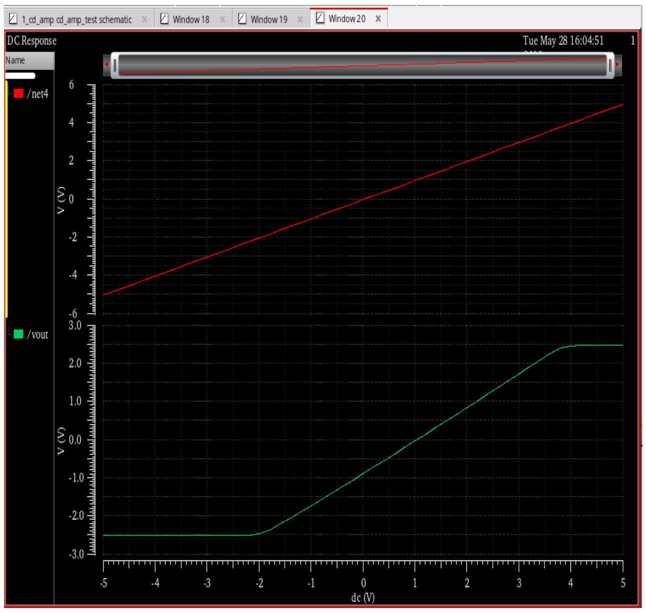
Simulation Results :

Transient Response :

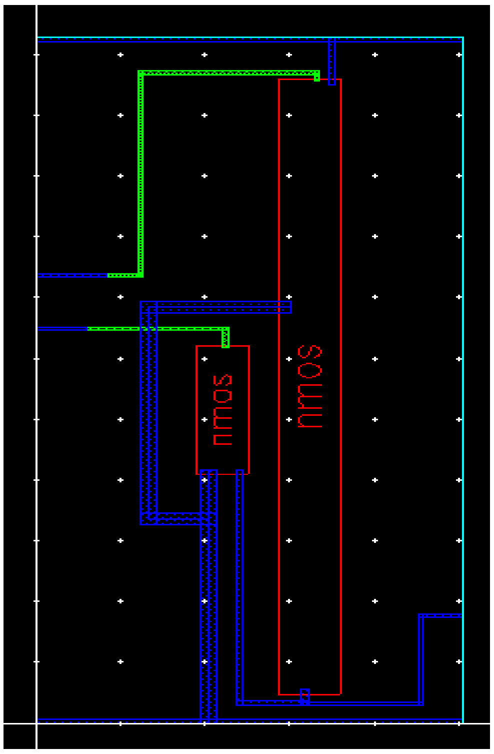


DC & AC Response :

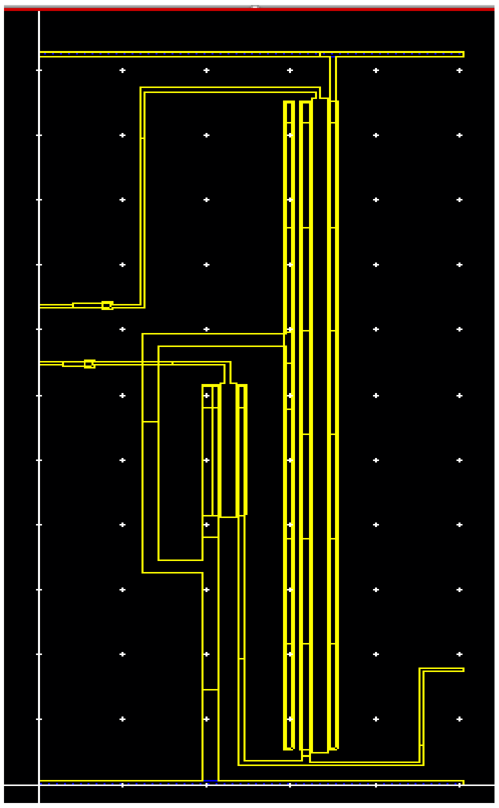




Layout :



Extracted view :



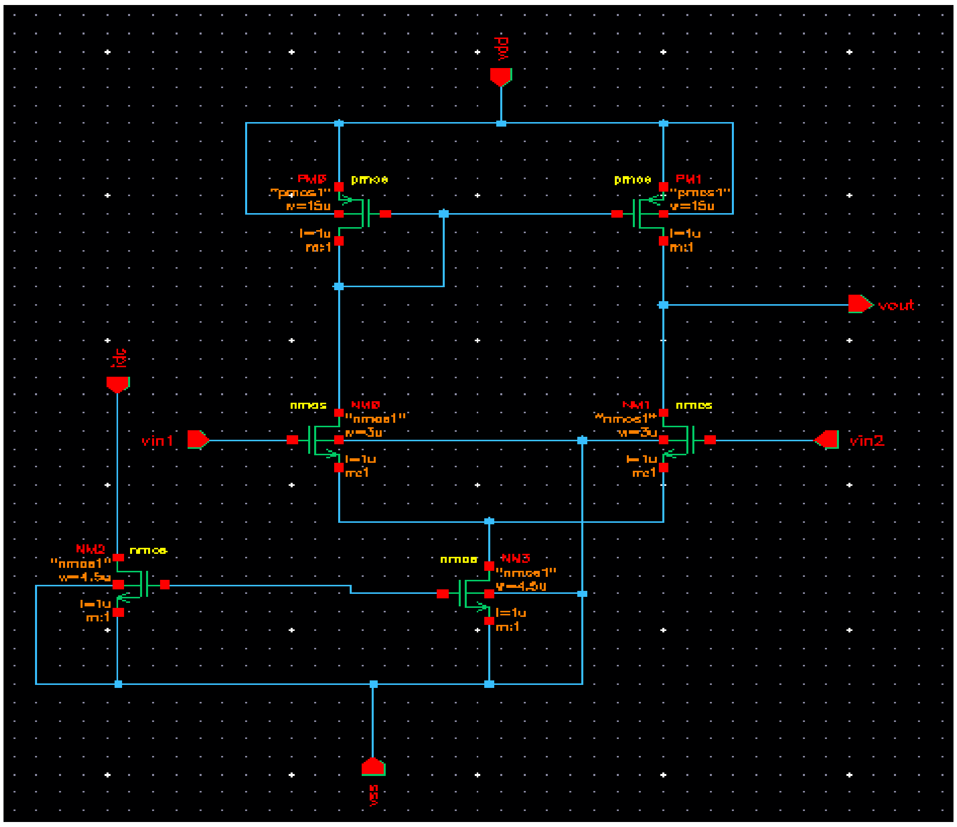
LAB 4 : Differential Amplifier

AIM : To simulate the schematic of the differential amplifier circuit.

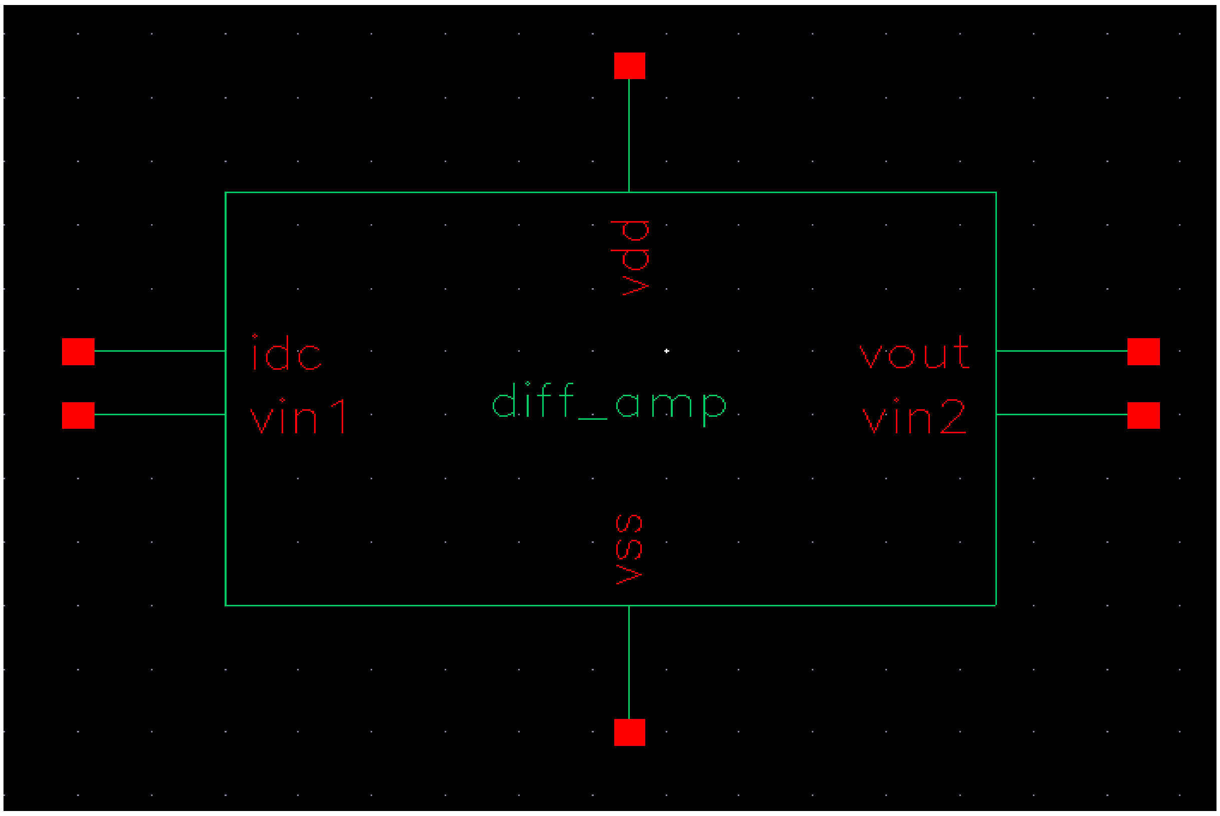
Theory :

The differential amplifier is probably the most widely used circuit building block in analog integrated circuits, principally op amps. We had a brief glimpse at one back in Chapter 3 section 3.4.3 when we were discussing input bias current. The differential amplifier can be implemented with BJTs or MOSFETs. A differential amplifier multiplies the voltage difference between two inputs (Vin+ - Vin-) by some constant factor Ad, the differential gain. It may have either one output or a pair of outputs where the signal of interest is the voltage difference between the two outputs. A differential amplifier also tends to reject the part of the input signals that are common to both inputs (Vin+ + Vin-)/2 . This is referred to as the common mode signal.

Schematic Entry :



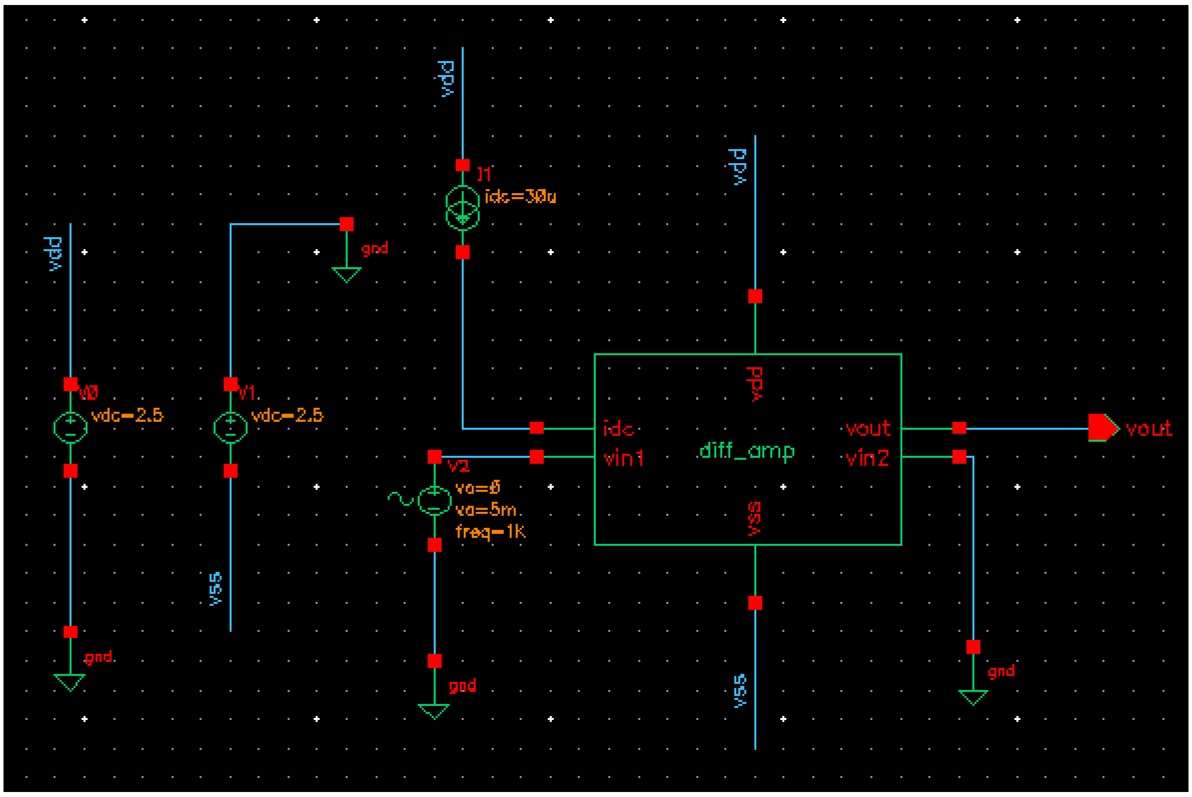
Symbol Creation :



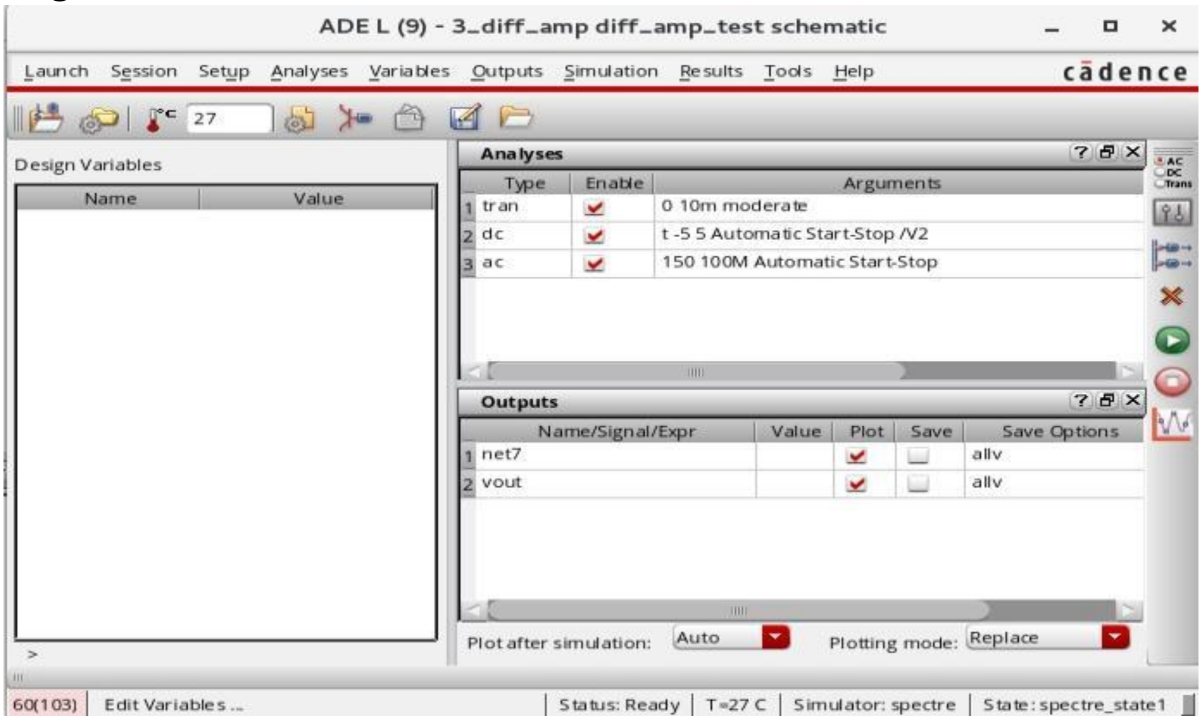
Design Entry for Test Schematic :

Library name	Cellview name	Properties
myDesignLib	cs_amplifier	Symbol
analogLib	vsin	Define pulse specification as AC Magnitude=1; DC Voltage=0; Offset Voltage=0; Amplitude=5m; Frequency=1K
analogLib	vdd,vss,gnd	Vdd=2.5;vss=-2.5;
AnalogLib	Idc	DC current = 30u

Test Schematic :



Settings in ADE L window :


ADE L (9) - 3_diff_amp diff_amp_test schematic

Launch Session Setup Analyses Variables Outputs Simulation Results Tools Help

27

Design Variables

Name	Value

Analyses

	Type	Enable	Arguments
1	tran	✓	0 10m moderate
2	dc	✓	t -5 5 Automatic Start-Stop /V2
3	ac	✓	150 100M Automatic Start-Stop

Outputs

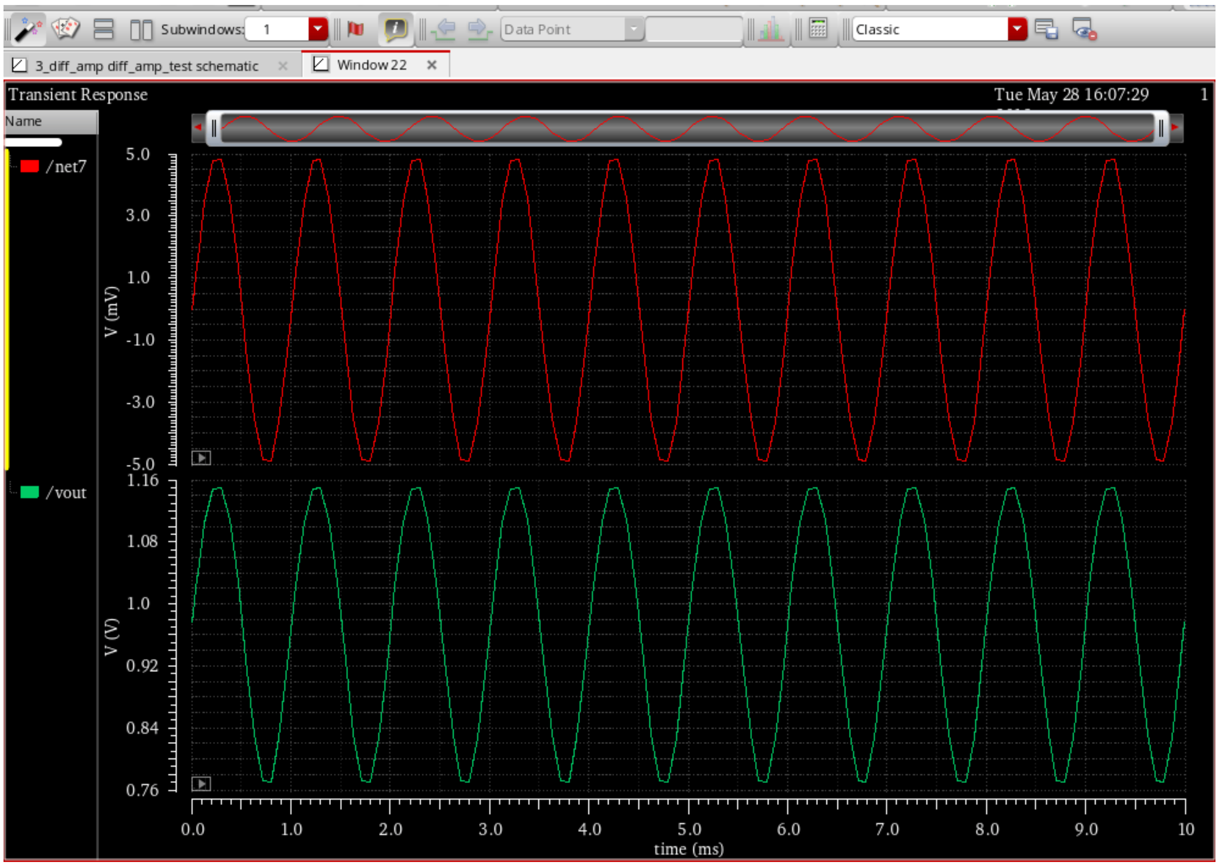
	Name/Signal/Expr	Value	Plot	Save	Save Options
1	net7		✓	☐	allv
2	vout		✓	☐	allv

Plot after simulation: Auto   Plotting mode: Replace

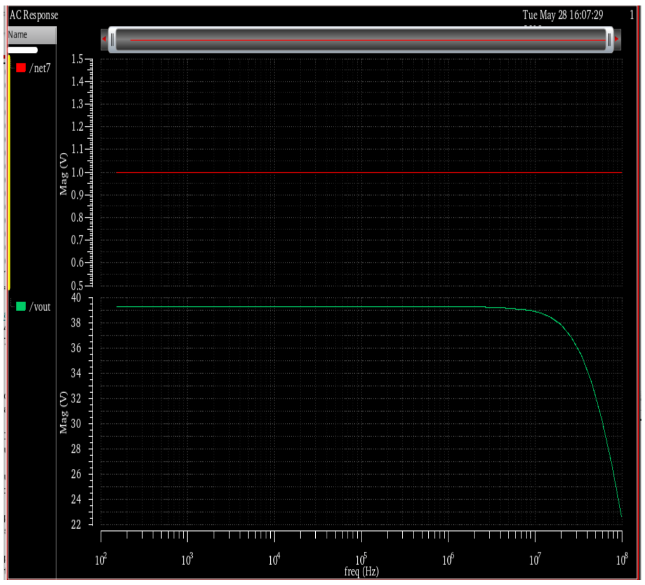
60(103) | Edit Variables ... | Status: Ready | T=27 C | Simulator: spectre | State: spectre_state1

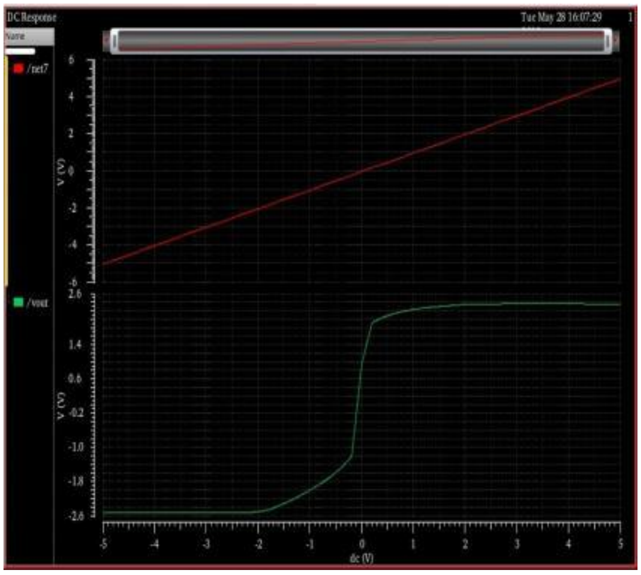
Simulation Results :

Transient Response :

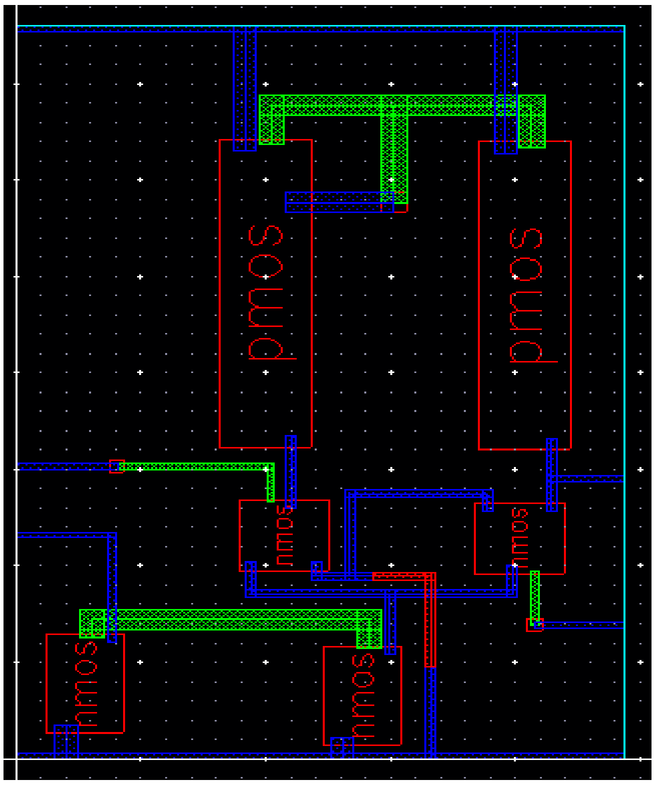


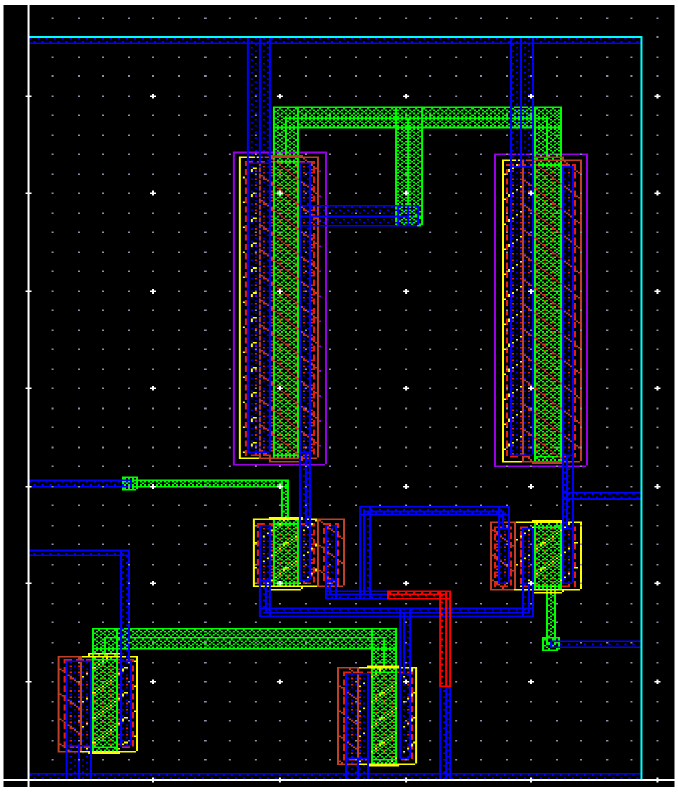
DC & AC Response :



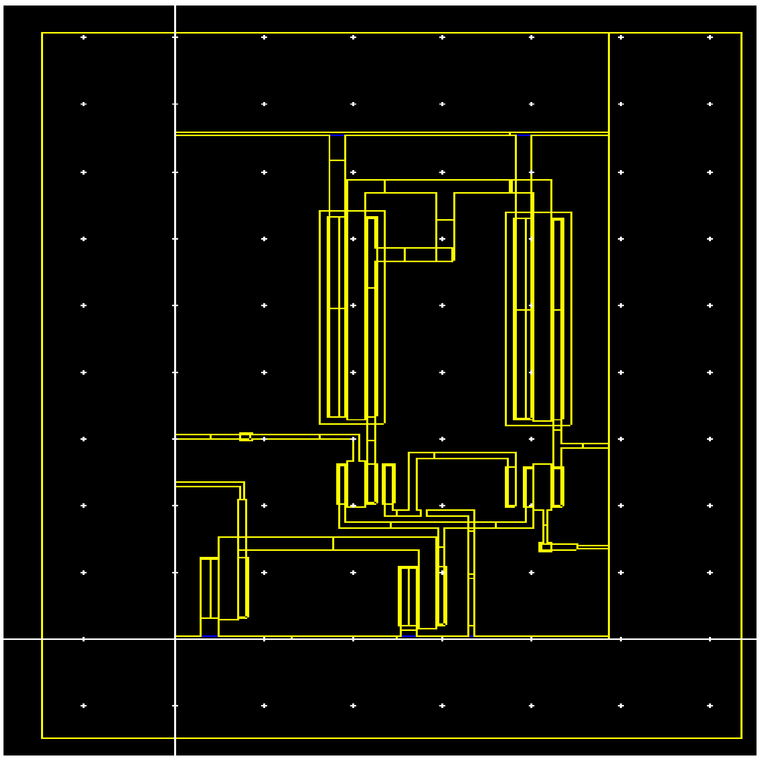


Layout :





Extracted View of Design :



LAB 5 : Operational Amplifier

AIM : To simulate the schematic circuit of the Operational Amplifier.

Theory :

An operational amplifier (often op-amp or op-amp) is a DC-coupled high- gain electronic voltage amplifier with a differential input and, usually, a single-ended output. In this configuration, an op-amp produces an output potential (relative to circuit ground) that is typically hundreds of thousands of times larger than the potential difference between its input terminals. Operational amplifiers had their origins in analog computers, where they were used to perform mathematical operations in many linear, non-linear and frequency-dependent circuits. The popularity of the op-amp as a building block in analog circuits is due to its versatility. Due to negative feedback, the characteristics of an op-amp circuit, its gain, input and output impedance, bandwidth etc. are determined by external components and have little dependence on temperature coefficients or manufacturing variations in the op-amp itself.

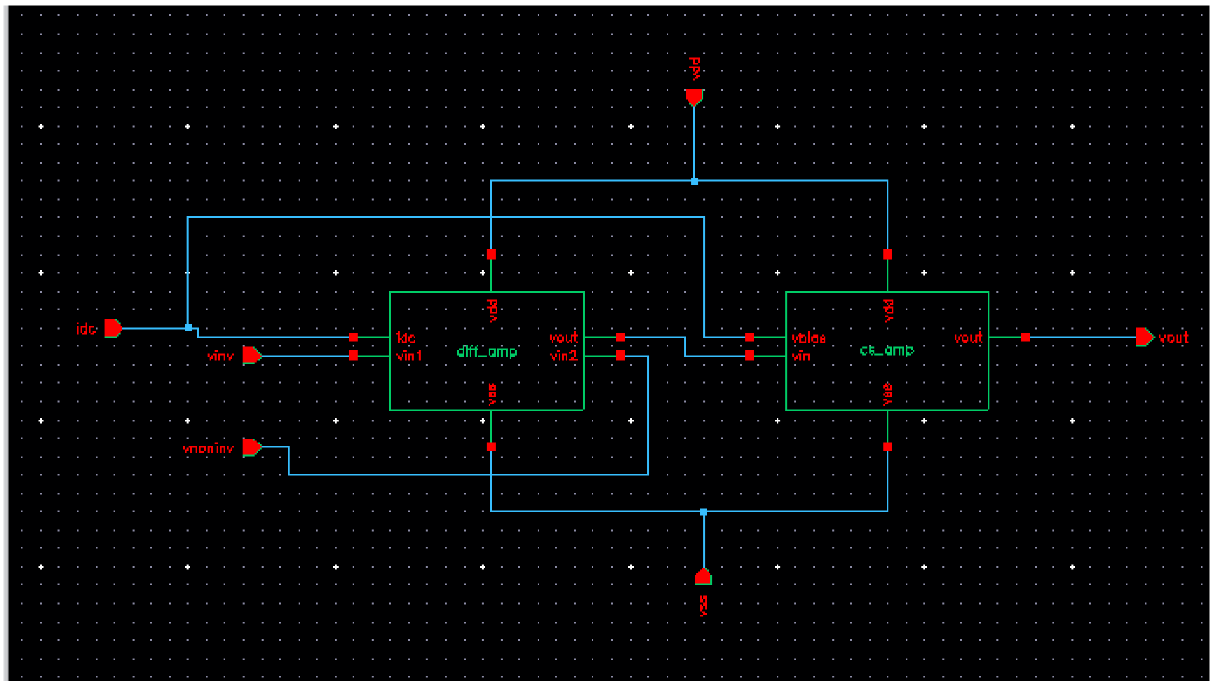
Op-amps are among the most widely used electronic devices today, being used in a vast array of consumer, industrial, and scientific devices. Many standard IC op-amps cost only a few cents in moderate production volume; however some integrated or hybrid operational amplifiers with special performance specifications may cost over $100 US in small quantities. Op- amps may be packaged as components, or used as elements of more complex integrated circuits. The op-amp is one type of differential amplifier.

The amplifier's differential inputs consist of a non-inverting input (+) with voltage V + and an inverting input (–) with voltage V − ; ideally the op- amp amplifies only the difference in voltage between the two, which is called the differential input voltage. The output voltage of the op-amp V out is given by the equation:

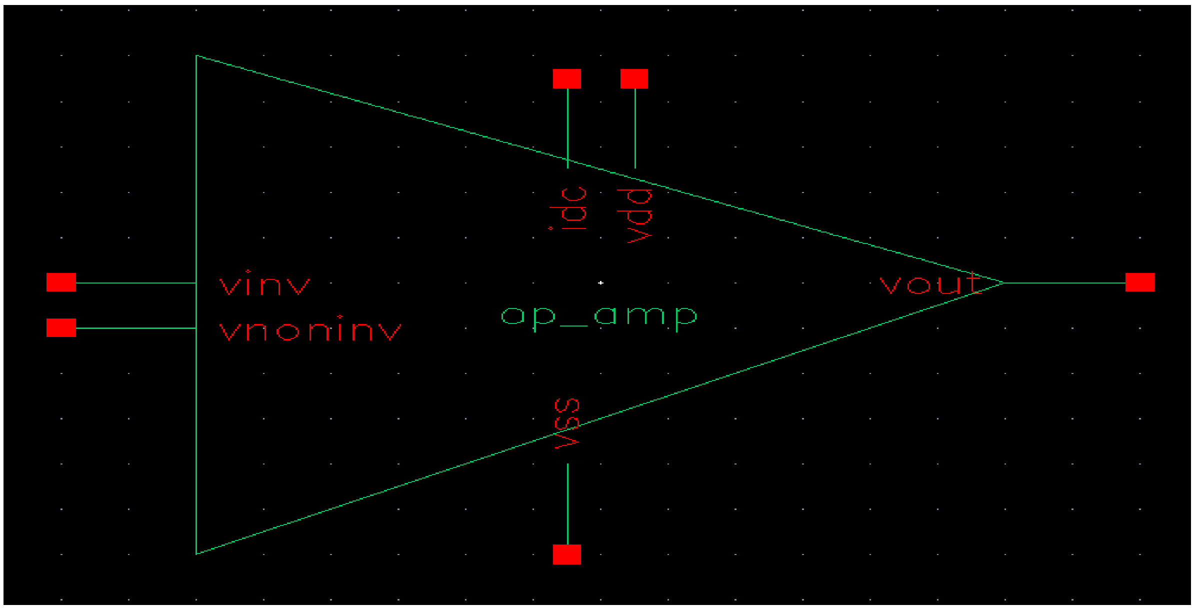
$$V_{out} = A_{OL} (V_{+} - V_{-})$$

where A_{OL} is the open-loop gain of the amplifier (the term "open-loop" refers to the absence of a feedback loop from the output to the input).

Schematic Entry :



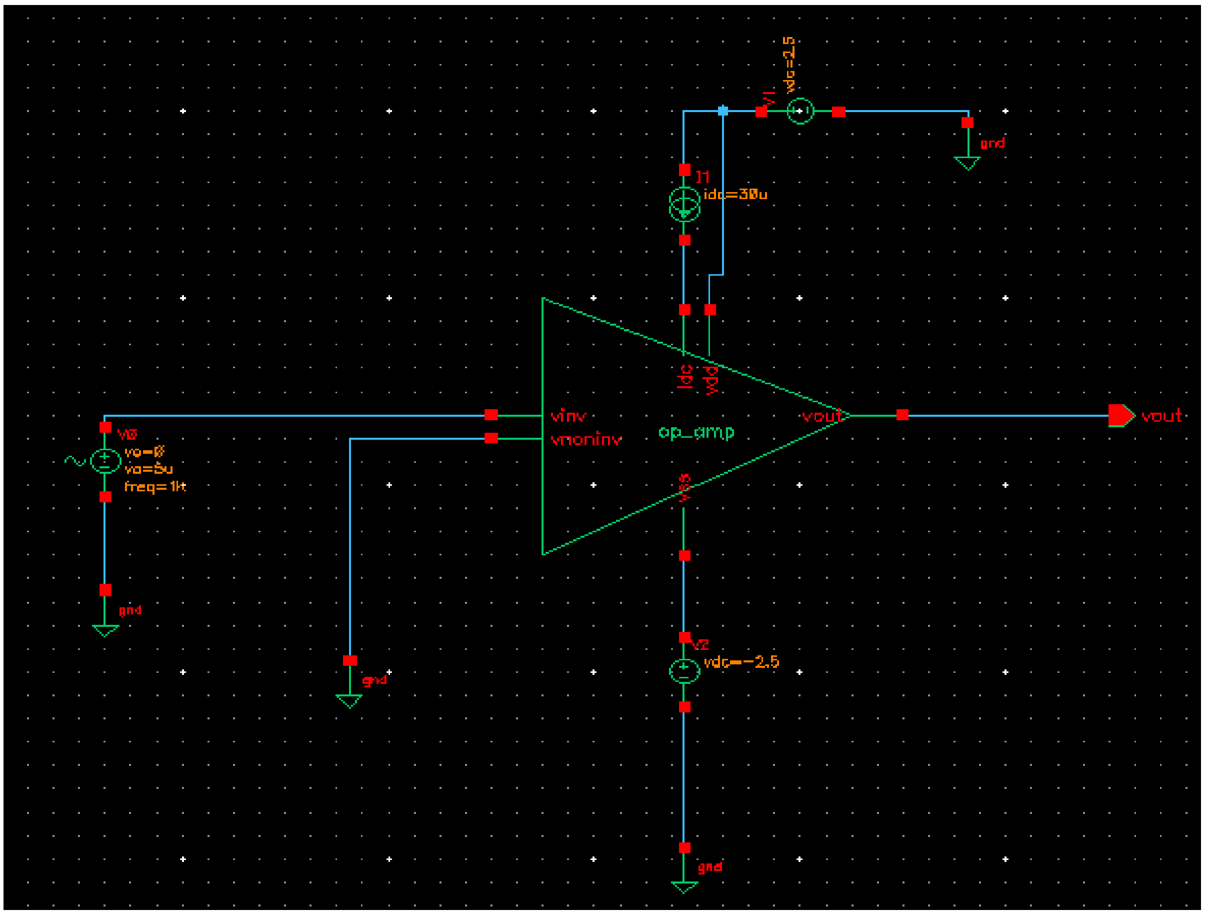
Symbol Creation :



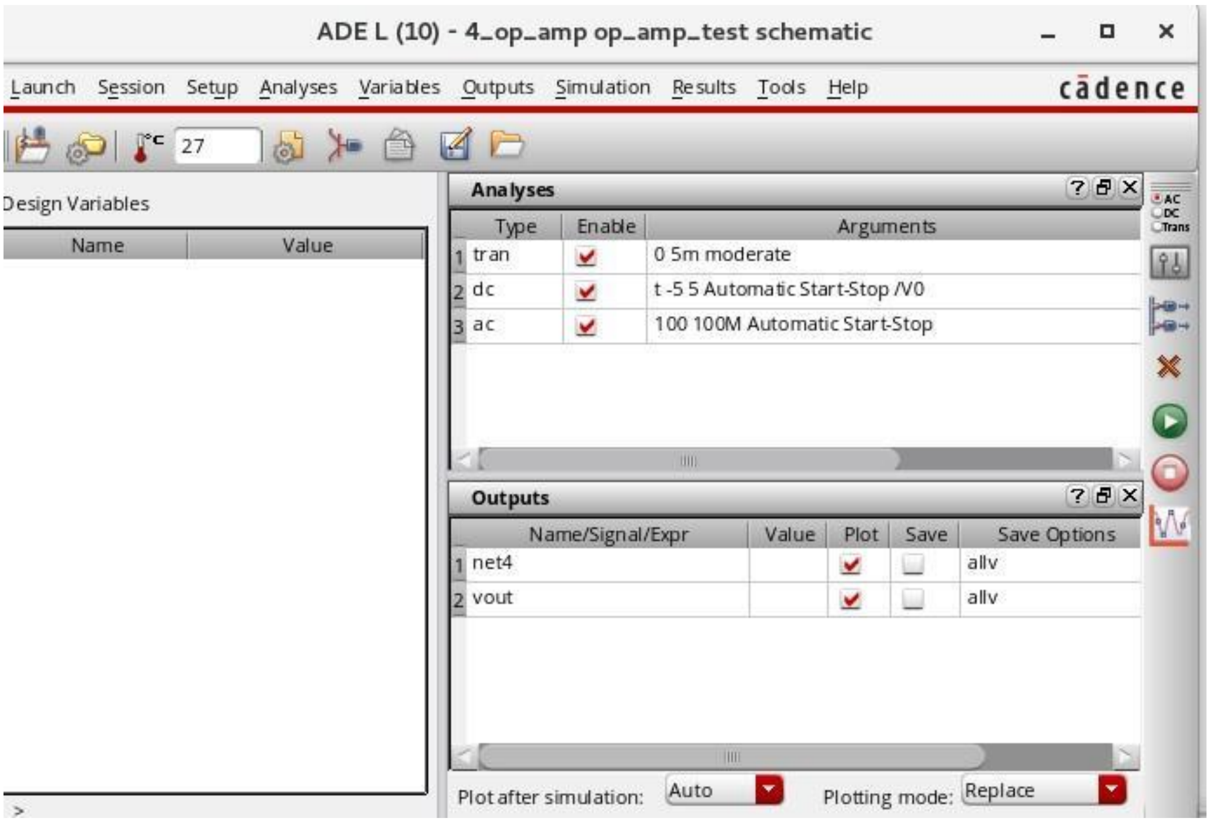
Design entry for test schematic :

Library name	Cellview name	Properties
myDesignLib	cs_amplifier	Symbol
analogLib	vsin	Define pulse specification as AC Magnitude=1; DC Voltage=0; Offset Voltage=0; Amplitude=5m; Frequency=1K
analogLib	vdd,vss,gnd	Vdd=2.5;vss=-2.5;
analogLib	Idc	DC current = 30u

Test Schematic of design :



Settings to be made in ADE L Window :



ADE L (10) - 4_op_amp op_amp_test schematic

Analyses

	Type	Enable	Arguments
1	tran	✓	0 5m moderate
2	dc	✓	t -5 5 Automatic Start-Stop /V0
3	ac	✓	100 100M Automatic Start-Stop

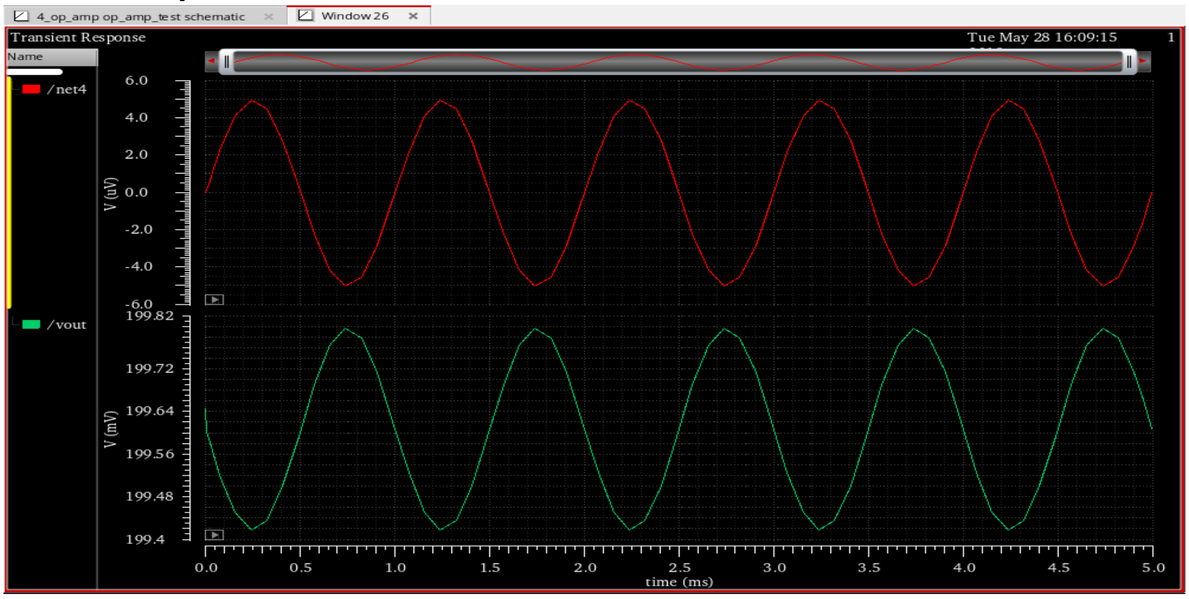
Outputs

	Name/Signal/Expr	Value	Plot	Save	Save Options
1	net4		✓		allv
2	vout		✓		allv

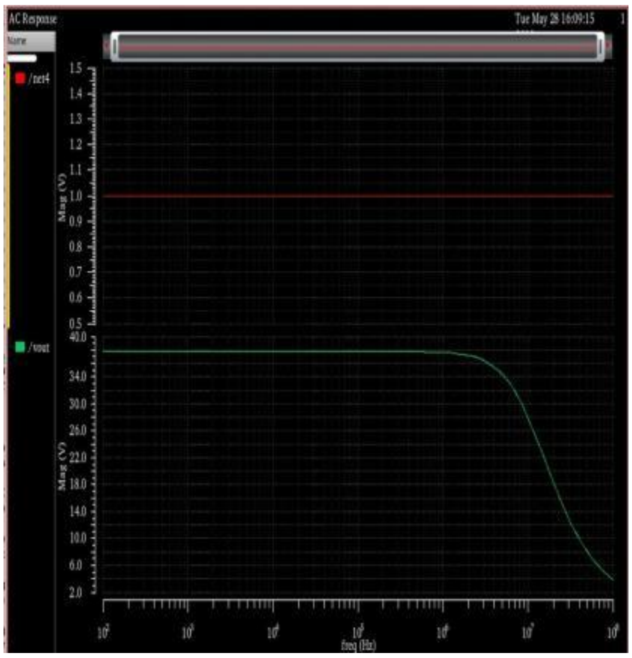
Plot after simulation: Auto Plotting mode: Replace

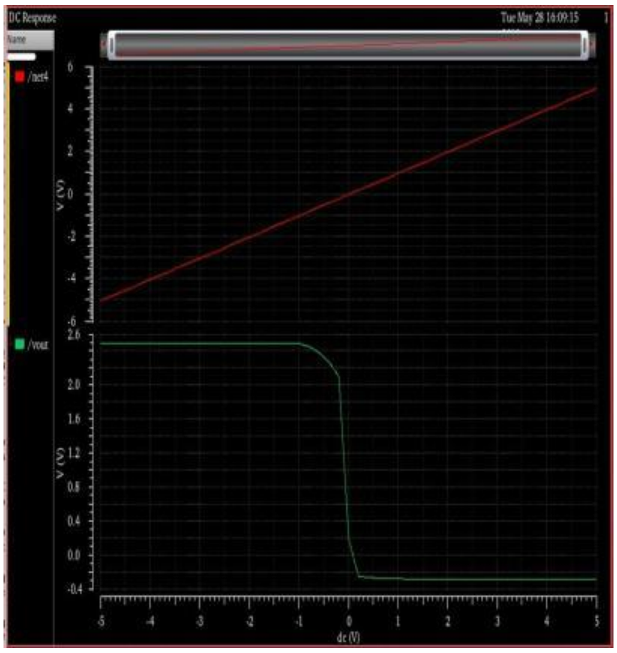
RESULTS :

Transient Response :

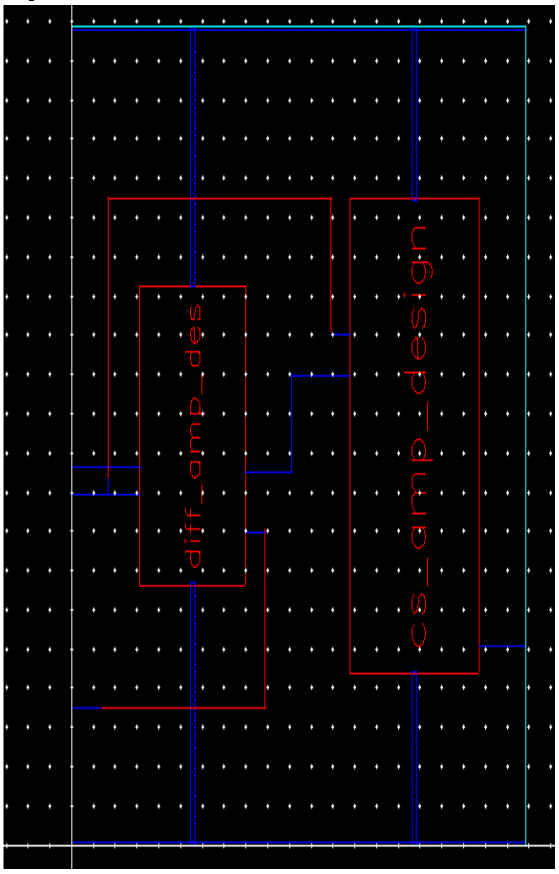


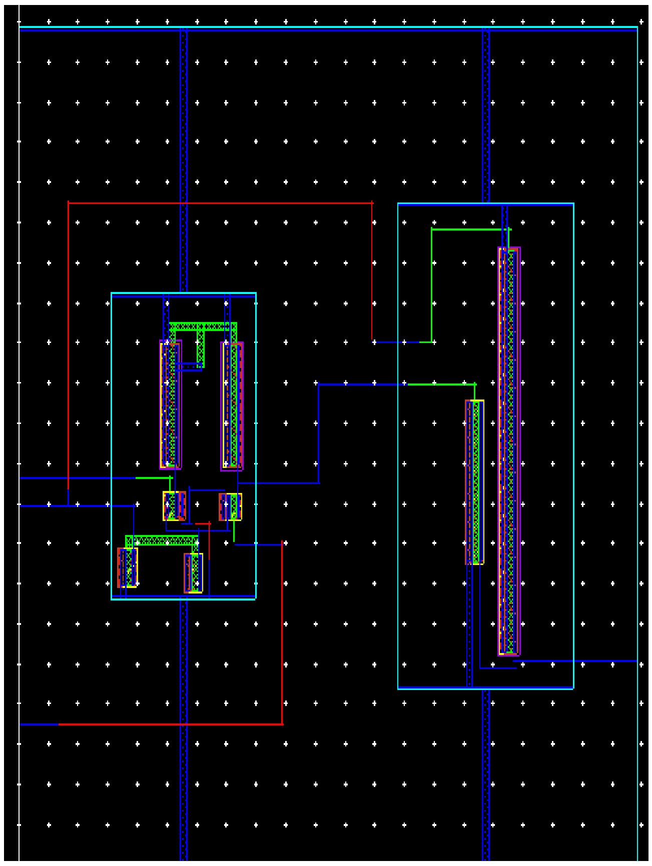
DC & AC Response :



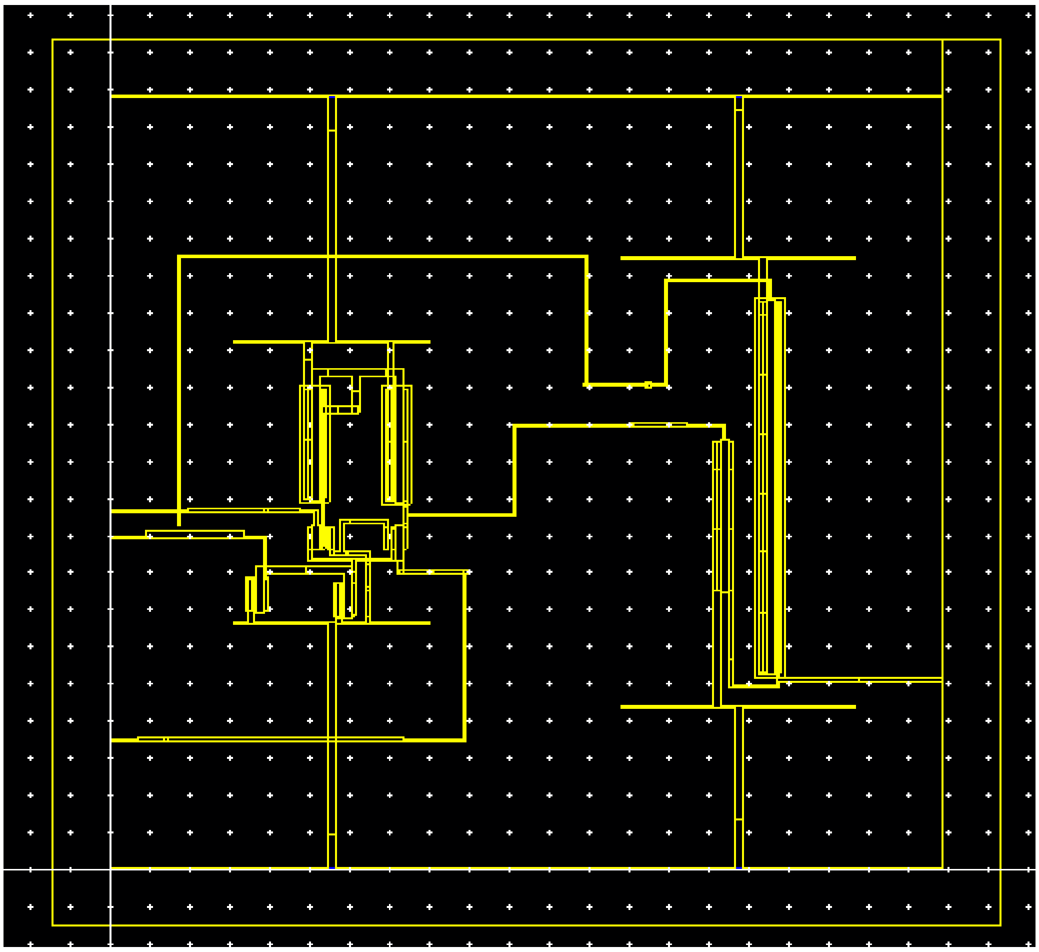


Layout View :





Extracted View of design :



LAB 6 : R-2R based DAC

AIM :

To design 4 bit R-2R DAC using Op-amp with given specifications and verifying the following

1. Schematic:

i) DC Analysis

ii) AC Analysis

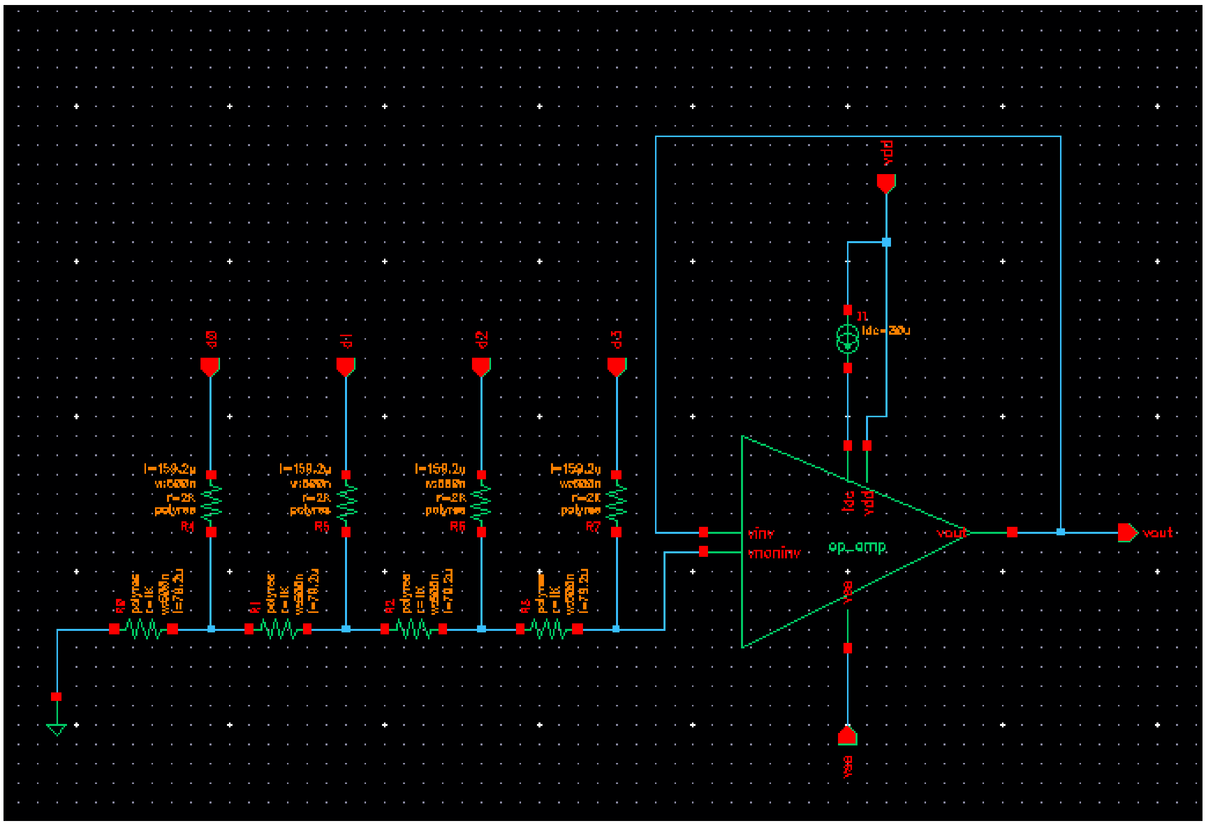
iii) Transient Analysis

Specifications :

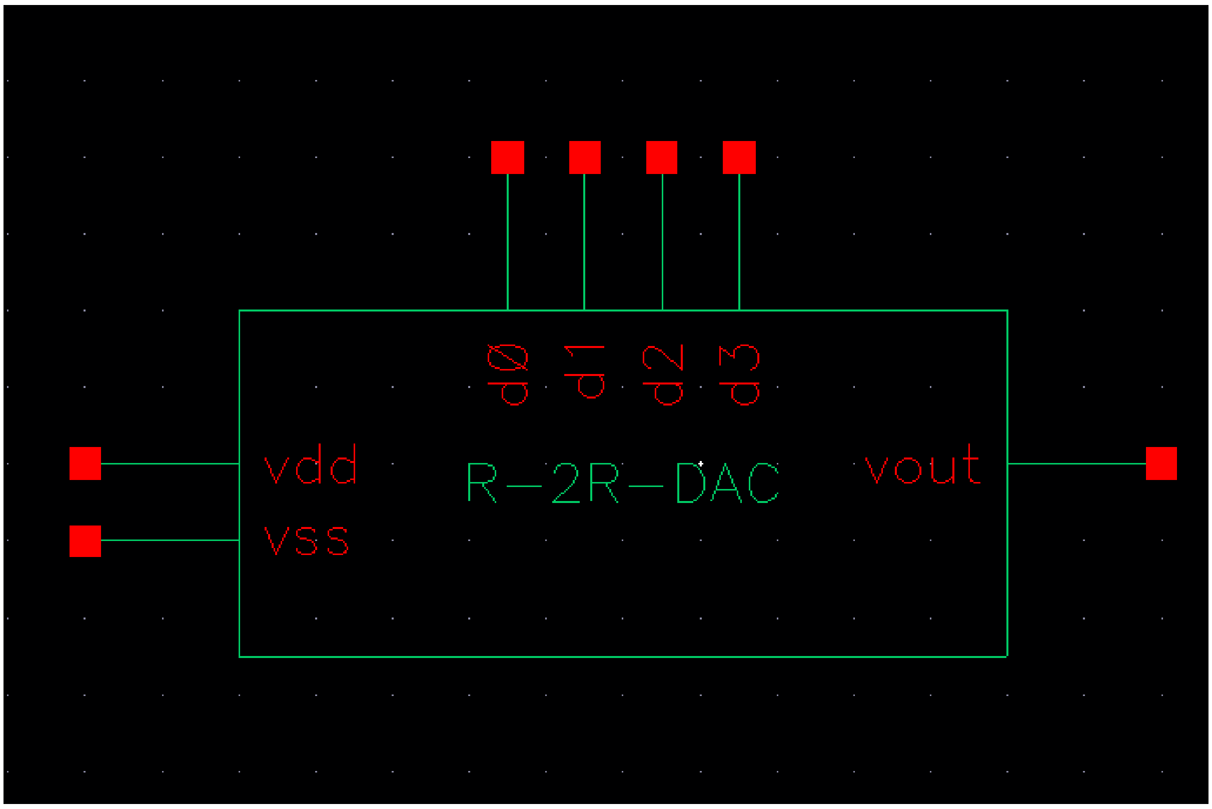
Library name	Cell name	Properties
gpdk180	Polyres	R=1K
gpdk180	Polyres	R=2K
analogLib	Idc	idc=30u
analogLib	gnd	-

Pin Names	Direction
D0, D1, D2, D3	Input
Vdd, vss	Input
Vout	Output

Schematic Entry :



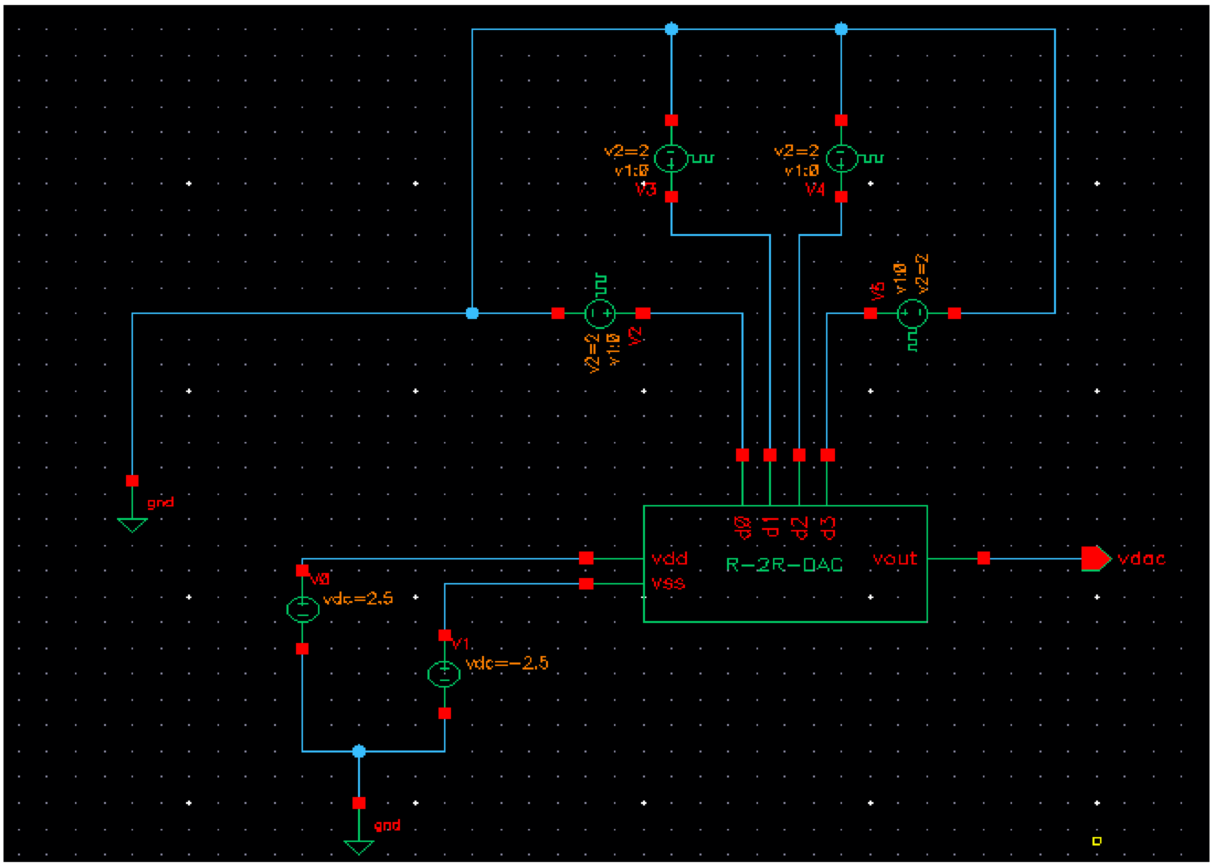
Symbol Creation :



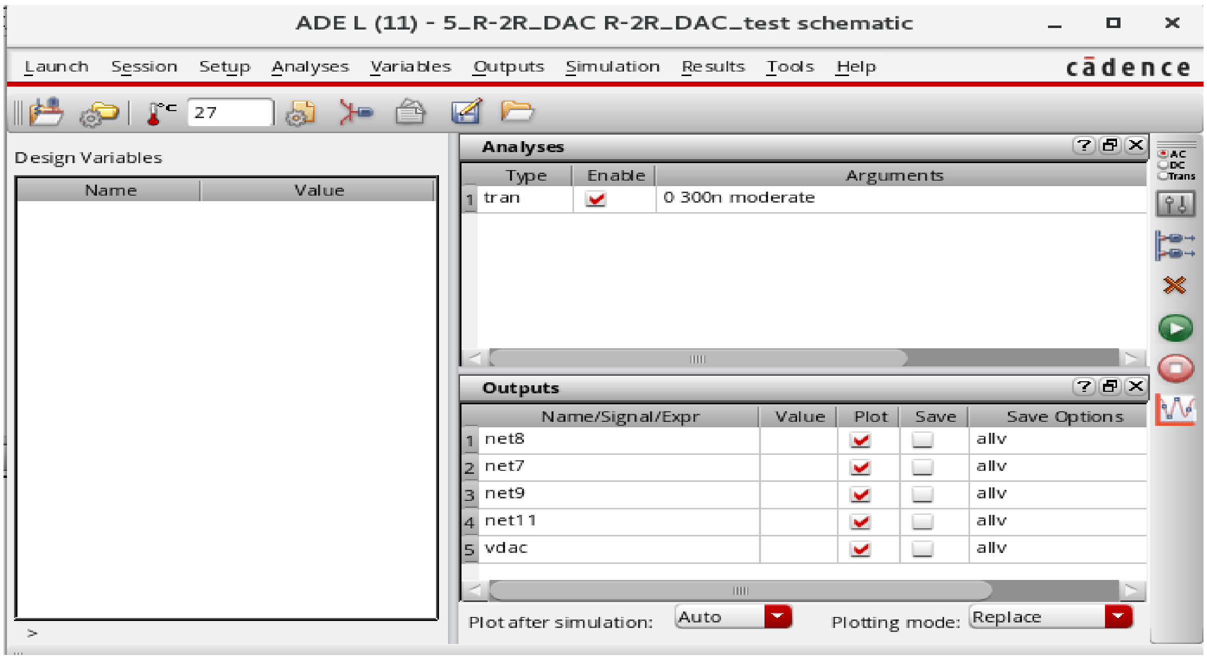
Design Entry for Test Schematic :

Library Name	Cell Name	Properties
analogLib	vpulse	For V0: v1=0, v2=2, Ton=5n, T=10n For V1: v1=0, v2=2, Ton=10n, T=20n For V2: v1=0, v2=2, Ton=20n, T=40n For V3: v1=0, v2=2, Ton=40n, T=80n
analogLib	Vdc,vdc	For vdd: DC voltage=2.5 For vss: DC voltage=-2.5
analogLib	gnd	-

Test Schematic :



Settings in ADE L window :


ADE L (11) - 5_R-2R_DAC R-2R_DAC_test schematic

Design Variables

Name	Value

Analyses

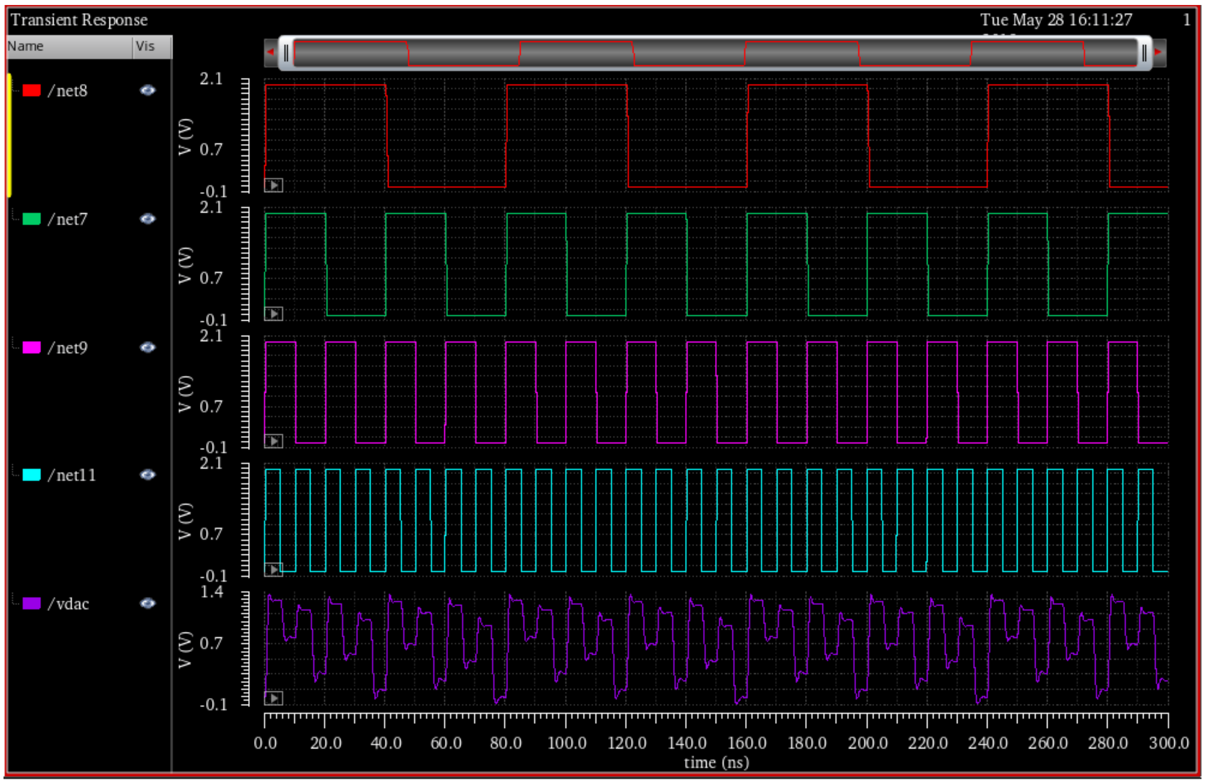
	Type	Enable	Arguments
1	tran	✓	0 300n moderate

Outputs

	Name/Signal/Expr	Value	Plot	Save	Save Options
1	net8		✓	☐	allv
2	net7		✓	☐	allv
3	net9		✓	☐	allv
4	net11		✓	☐	allv
5	vdac		✓	☐	allv

Plot after simulation: Auto   Plotting mode: Replace

RESULTS :


Transient Response
Tue May 28 16:11:27
1
Name
Vis
/net8
/net7
/net9
/net11
/vdac
V (V)
2.1
0.7
-0.1
1.4
0.0
20.0
40.0
60.0
80.0
100.0
120.0
140.0
160.0
180.0
200.0
220.0
240.0
260.0
280.0
300.0
time (ns)

VISION

To become a pioneer in developing competent professionals with societal and ethical values through transformational learning and interdisciplinary research in the field of Electronics and Communication Engineering.

MISSION

The department of Electronics and Communication is committed to:

M1: Offer quality technical education through experiential learning to produce competent engineering professionals.

M2: Encourage a culture of innovation and multidisciplinary research in collaboration with industries/universities.

M3: Develop interpersonal, intrapersonal, entrepreneurial and communication skills among students to enhance their employability.

M4: Create a congenial environment for the faculty and students to achieve their desired goals and to serve society by upholding ethical values.

PROGRAM EDUCATIONAL OBJECTIVES (PEOS)

Upon completion of the program, graduates will be able to:
PEO1: Tackle complex engineering problems with the sound knowledge of basic science and mathematics.

PEO2: Utilize their knowledge and skills to develop solutions in multi-disciplinary environments through collaborative research.

PEO 3: Inculcate effective communication skills, teamwork and leadership for a successful career in industry and academia.

PEO4: Exhibit professional ethics and social awareness in their professional career and engage in lifelong learning.

PROGRAM OUTCOMES (POs)

Engineering Graduates will be able to:
PO1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2. Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3. Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4. Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5. Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6. The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7. Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9. Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10. Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11. Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12. Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO1
Apply the knowledge of leading-edge hardware and software tools to solve problems in the area of Embedded Systems, VLSI and IoT.

PSO2
Apply the concepts of Signal and Image Processing to solve problems in communication systems.