B. E. ECE
Choice Based Credit System (CBCS) and Outcome Based Education (OBE) SEMESTER – VII

VLSI LAB

Course Code	**18ECL77**	**CIE Marks**	**40**
Number of Lecture Hours/Week	**02HrTutorial (Instructions) + 02 Hours Laboratory**	**SEE Marks**	**60**
RBT Levels	**L1, L2, L3**	**Exam Hours**	**03**

CREDITS – 02

Course Learning Objectives: This course will enable students to:

- Design, model, simulate and verify CMOS digital circuits
- Design layouts and perform physical verification of CMOS digital circuits
- Perform ASIC design flow and understand the process of synthesis, synthesis constraints and evaluating the synthesis reports to obtain optimum gate level netlist
- Perform RTL-GDSII flow and understand the stages in ASIC design

Experiments can be conducted using any of the following or equivalent design tools: Cadence/Synopsis/Mentor Graphics/Microwind

Laboratory
Experiments Part – A
Analog Design

Use any VLSI design tools to carry out the experiments, use library files and technology files below 180 nm.

1. a) Capture the schematic of CMOS inverter with load capacitance of 0.1pF and set the widths of inverter with Wn = Wp, Wn = 2Wp, Wn = Wp/2 and length at selected technology. Carry out the following:
   a. Set the input signal to a pulse with rise time, fall time of 1ns and pulse width of 10ns and time period of 20ns and plot the input voltage and output voltage of designed inverter?
   b. From the simulation results compute tpHL, tpLH and td for all three geometrical settings of width?
   c. Tabulate the results of delay and find the best geometry for minimum delay for CMOS inverter?

1. b)Draw layout of inverter with Wp/Wn = 40/20, use optimum layout methods. Verify for DRC and LVS, extract parasitic and perform post layout simulations, compare the results with pre-layout simulations. Record the observations.

2. a) Capture the schematic of 2-input CMOS NAND gate having similar delay as that of CMOS inverter computed in experiment 1. Verify the functionality of NAND gate and also find out the delay td for all four possible combinations of input vectors. Table the results. Increase the drive strength to 2X and 4X and tabulate the results.
2.b)Draw layout of NAND withWp/Wn = 40/20, use optimum layout methods. Verify for DRC and LVS, extract parasitic and perform post layout simulations, compare the results with pre-layout simulations. Record the observations.

3.a) Capture schematic of Common Source Amplifier with PMOS Current Mirror Load and find its transient response and AC response? Measures the Unity Gain Bandwidth (UGB), amplification factor by varying transistor geometries, study the impact of variation in width to UGB.

1. b) Draw layout of common source amplifier, use optimum layout methods. Verify for DRC and LVS, extract parasitic and perform post layout simulations, compare the results with pre-layout simulations. Record the observations.

4. a)Capture schematic of two-stage operational amplifier and measure the following:

a. UGB
b. dB bandwidth
c. Gain margin and phase margin with and without coupling capacitance
d. Use the op-amp in the inverting and non-inverting configuration and verify its functionality
e. Study the UGB, 3dB bandwidth, gain and power requirement in op-amp by varying the stage wise transistor geometries and record the observations.

4. b) Draw layout of two-stage operational amplifier with minimum transistor width set to 300 (in 180/90/45 nm technology), choose appropriate transistor geometries as per the results obtained in 4.a. Use optimum layout methods. Verify for DRC and LVS, extract parasitic and perform post layout simulations, compare the results with pre-layout simulations. Record the observations.

Part - B

Digital Design

Carry out the experiments using semicustom design flow or ASIC design flow, use technology library 180/90/45nm and below

Note: The experiments can also be carried out using FPGA design flow, it is required to set appropriate constraints in FPGA advanced synthesis options

1. Write Verilog code for 4-bit up/down asynchronous reset counter and carry out the following:

a. Verify the functionality using test bench
b. Synthesize the design by setting area and timing constraint. Obtain the gate level netlist, find the critical path and maximum frequency of operation. Record the area requirement in terms of number of cells required and properties of each cell in terms of driving strength, power and area requirement.
c. Perform the above for 32-bit up/down counter and identify the critical path, delay of critical path, and maximum frequency of operation, total number of cells required and total area.

2.Write Verilog code for 4-bit adder and verify its functionality using test bench. Synthesize the design by setting proper constraints and obtain the net list. From the report generated identify critical path, maximum delay, total number of cells, power requirement and total area required. Change the constraints and obtain optimum synthesis results.

3. Write Verilog code for UART and carry out the following:

a. Perform functional verification using test bench
b. Synthesize the design targeting suitable library and by setting area and timing constraints
c. For various constrains set, tabulate the area, power and delay for the synthesized netlist

d. Identify the critical path and set the constraints to obtain optimum gate level netlist with suitable constraints

4. Write Verilog code for 32-bit ALU supporting four logical and four arithmetic operations, use case statement and if statement for ALU behavioral modeling.

a. Perform functional verification using test bench
b. Synthesize the design targeting suitable library by setting area and timing constraints
c. For various constrains set, tabulate the area, power and delay for the synthesized netlist
d. Identify the critical path and set the constraints to obtain optimum gate level netlist with suitable constraints

Compare the synthesis results of ALU modeled using IF and CASE statements.

5. Write Verilog code for Latch and Flip-flop, Synthesize the design and compare the synthesis report (D, SR, JK).

6. For the synthesized netlist carry out the following for any two above experiments:

a. Floor planning (automatic), identify the placement of pads
b. Placement and Routing, record the parameters such as no. of layers used for routing, flip method for placement of standard cells, placement of standard cells, routes of power and ground, and routing of standard cells
c. Physical verification and record the LVS and DRC reports
d. Perform Back annotation and verify the functionality of the design
e. Generate GDSII and record the number of masks and its color composition

Course Outcomes: On the completion of this laboratory course, the students will be able to:

- Design and simulate combinational and sequential digital circuits using Verilog HDL
- Understand the Synthesis process of digital circuits using EDA tool.
- Perform ASIC design flow and understand the process of synthesis, synthesis constraints and evaluating the synthesis reports to obtain optimum gate level net list
- Design and simulate basic CMOS circuits like inverter, common source amplifier and differential amplifiers.
- Perform RTL-GDSII flow and understand the stages in ASIC design.

List of Experiments

Part-A Analog Design Flow	
Sl No.	Name of the Experiment
1	A CMOS inverter
2	2-input CMOS NAND gate
3	Common Source Amplifier
4	Two-stage operational amplifier
Part-B Digital Design Flow	
1	4-bit Up/Down Asynchronous Reset Counter
2	4-bit Adder
3	UART
4	32-bit ALU
5	Flip-flop and Latch(D-FF/D-Latch, JK-FF/JK-Latch, SR-FF/SR-Latch)
6	Physical Design (for two experiments)

Course Outcomes

On the completion of this laboratory course, the students will be able to:	
CO1	Apply the basics of MOSFET's and Verilog HDL programming in the design of digital and analog circuits.
CO2	Develop Verilog code for various combinational and sequential digital circuits and Design various analog circuits using concepts of MOSFETS
CO3	Analyze the design for the complete ASIC and FPGA flow with various test inputs, Perform the complete RTL to GDSII flow for the given circuits
CO4	Interpret the output obtained with relevant explanation.
CO5	Use EDA tool to simulate, debug and analyze the digital and analog circuits

CO-PO Mapping

CO/PO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO1	3	--	--	--	--	--	--	2	2	2	--	--
CO2	2	3	--	--	--	--	--	2	2	2	--	--
CO3	--	2	3	--	--	--	--	2	2	2	--	--
CO4	--	--	--	3	--	--	--	2	2	2	--	--
CO5	--	--	--	--	3	--	--	2	2	2	--	--

PART - A

ANALOG DESIGN

Introduction:

Before starting to work on a design, create a Workspace (Folder) for the project individually.

WORK SPACE CREATION:

Make a right click on the **Desktop** and select the option "**New Folder**" as shown in Figure - 1.

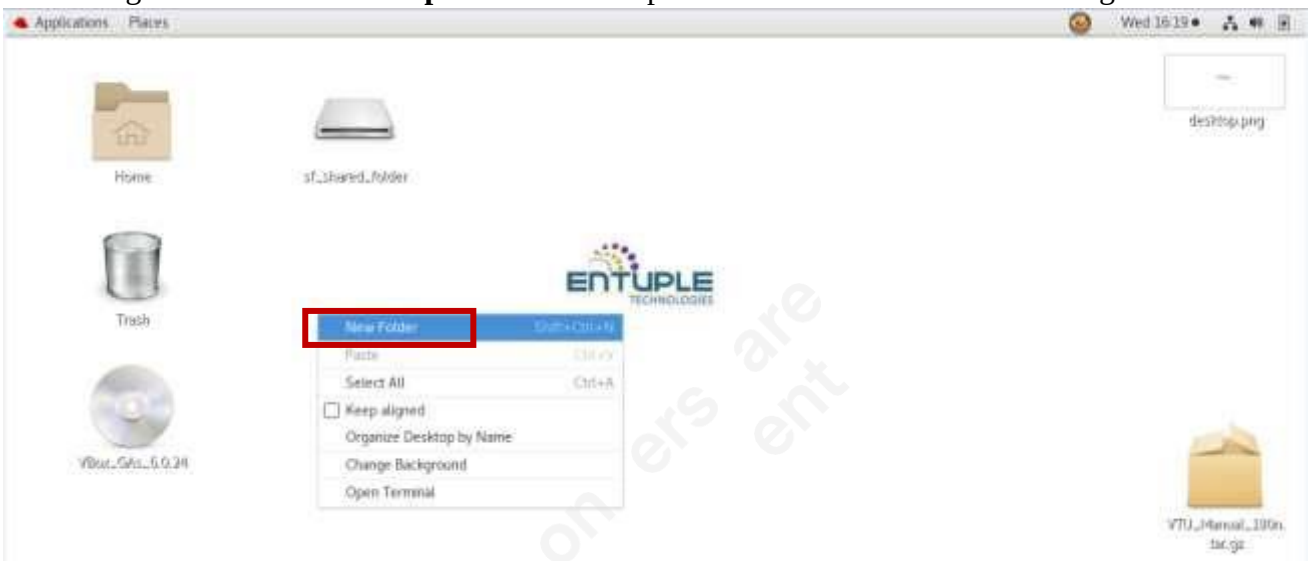


Figure – 1: Workspace Creation

Name the folder (for example: **VTU_LAB_EXP**) and click on "**Create**" as shown in Figure - 2.

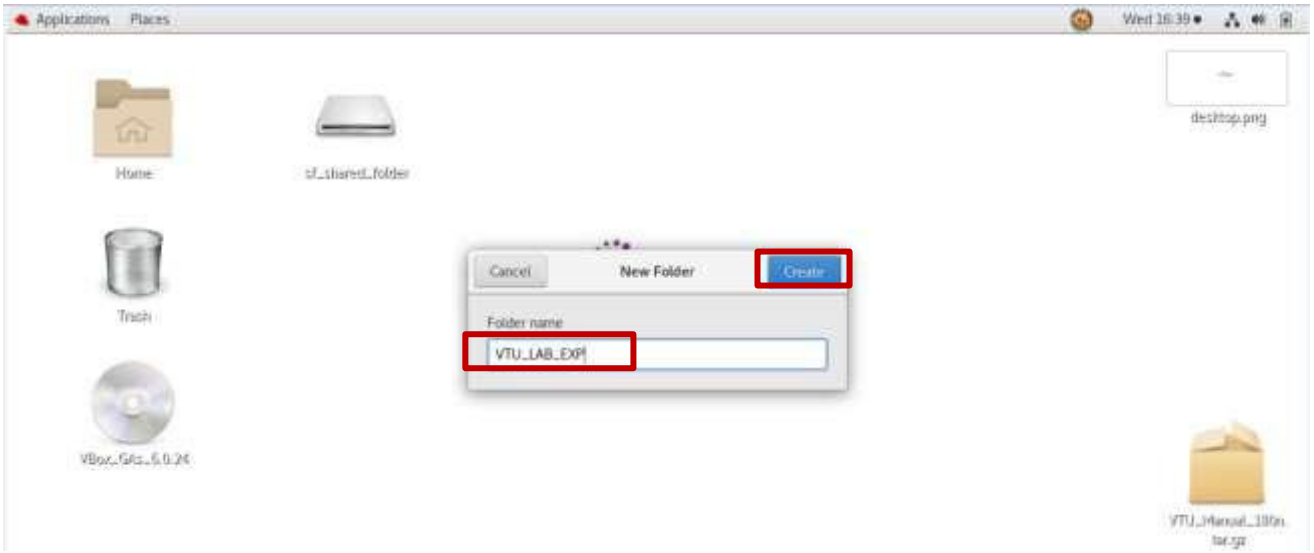
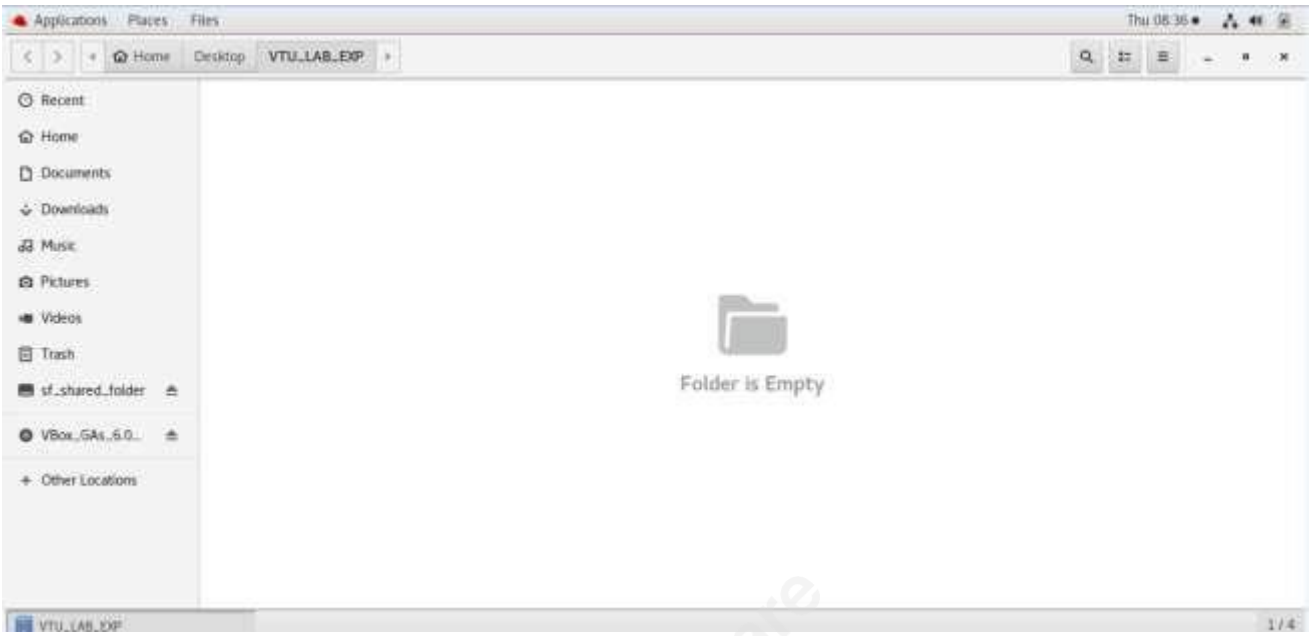


Figure – 2: Name the Folder

Open the folder by a double click and the window can be seen as shown in Figure - 3.



Make a right click and select "**Open in Terminal**" as shown in Figure - 4.

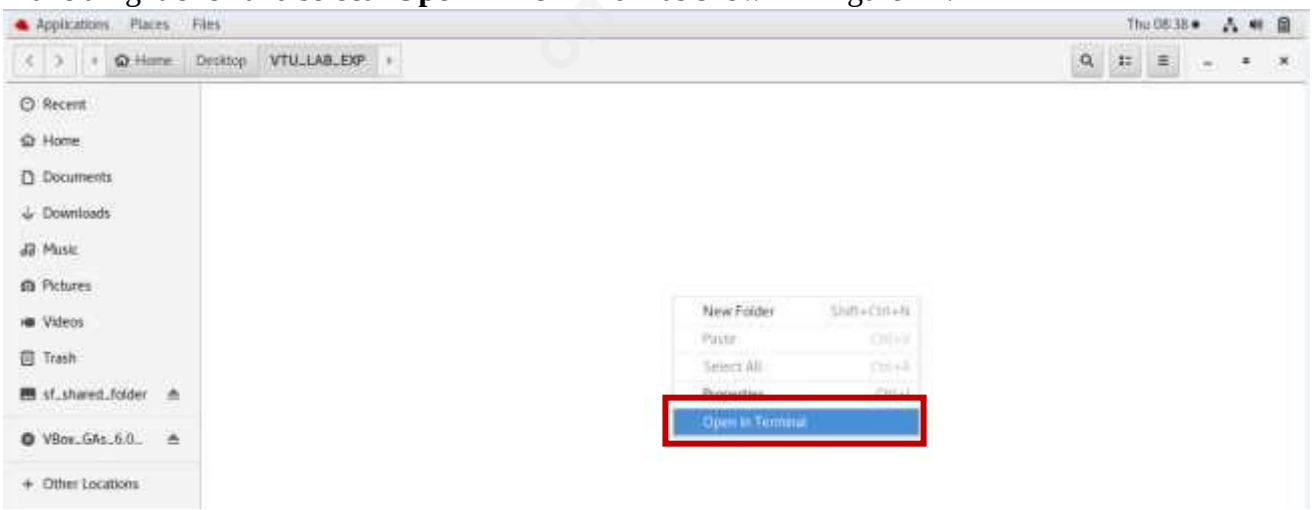


Figure – 4: Open in Terminal

Type the command "**csh**" to initialize **shell** and source the "**cshrc**" file with the command "**source /home/install/cshrc**". "cshrc" file will provide the details of the installation directory of the Cadence Tools.

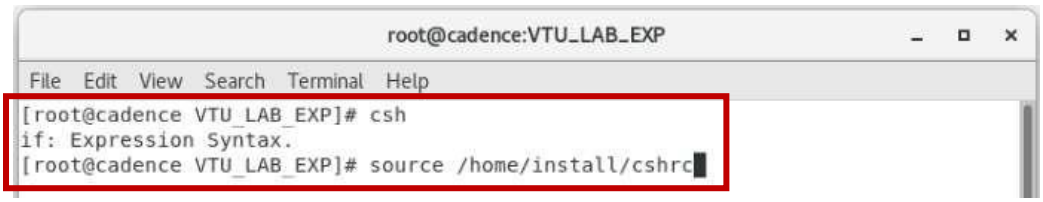


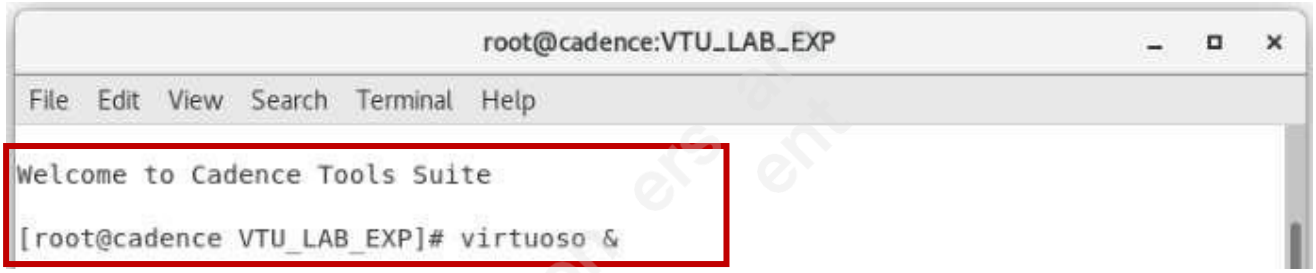
Figure – 5: "csh" and "source /home/install/cshrc" commands

INVOKING VIRTUOSO:

After sourcing the "**cshrc**" file, click on "**Enter**" on the keyboard. The welcome screen with the text "**Welcome to Cadence Tools Suite**" can be seen as shown in Figure - 5.

Figure - 6: Welcome screen

Invoke virtuoso using the command "**virtuoso &**" or "**virtuoso**" as shown in Figure – 7 and click on "**Enter**" in the keyboard.



he Virtuoso "**Command Interpreter Window (CIW)**" can be seen as shown in Figure - 8.

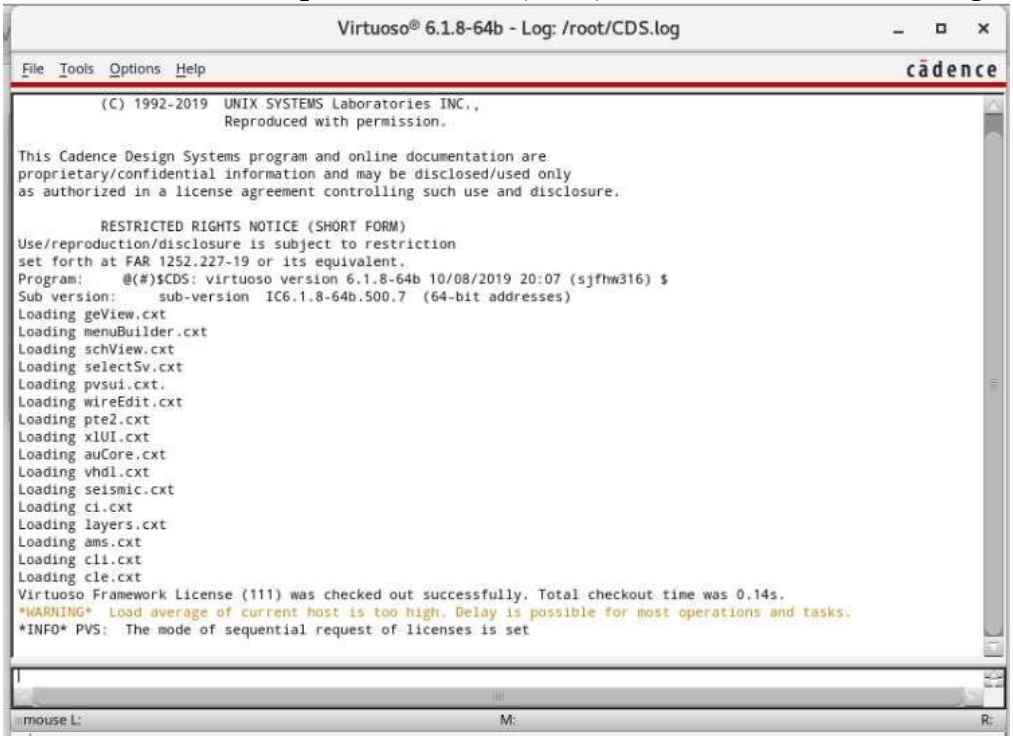


Figure – 8: Command Interpreter Window (CIW)

LAB – 01: CMOS INVERTER

Objective:

(a) Capture the Schematic of a CMOS Inverter with Load Capacitance of 0.1 pF and set the Widths of Inverter with

(i) $W_N = W_P$

(ii) $W_N = 2\ W_P$

(iii) $W_N = W_P / 2$

and Length at selected Technology. Carry out the following:

1. Set the Input Signal to a pulse with Rise Time, Fall Time of 1 ps and Pulse Width of 10 ns, Time Period of 20 ns and plot the input voltage and output voltage of the designed Inverter
2. From the Simulation Results, compute tp_{HL}, tp_{LH} and t_{PD} for all the three geometrical settings of Width
3. Tabulate the results of delay and find the best geometry for minimum delay for CMOS Inverter

Solution:

(a) Schematic Capture of CMOS Inverter

CREATE A LIBRARY:

To create a **New Library**, select "**Tools □ Library Manager**" from the top menu as shown in Figure – 1.1.

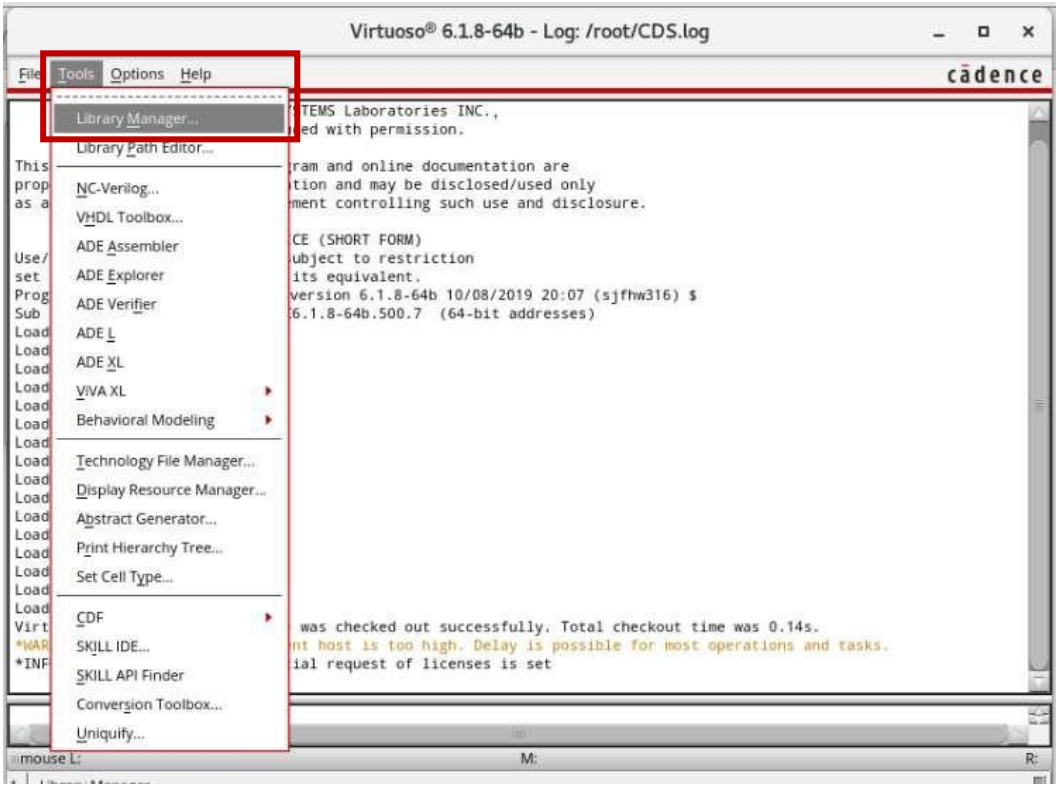


Figure – 1.1: Tools □ Library Manager

The Cadence Library Manager shows up as in Figure – 1.2.

Select "**File □ New □ Library**" from the top menu as shown in Figure – 1.3.

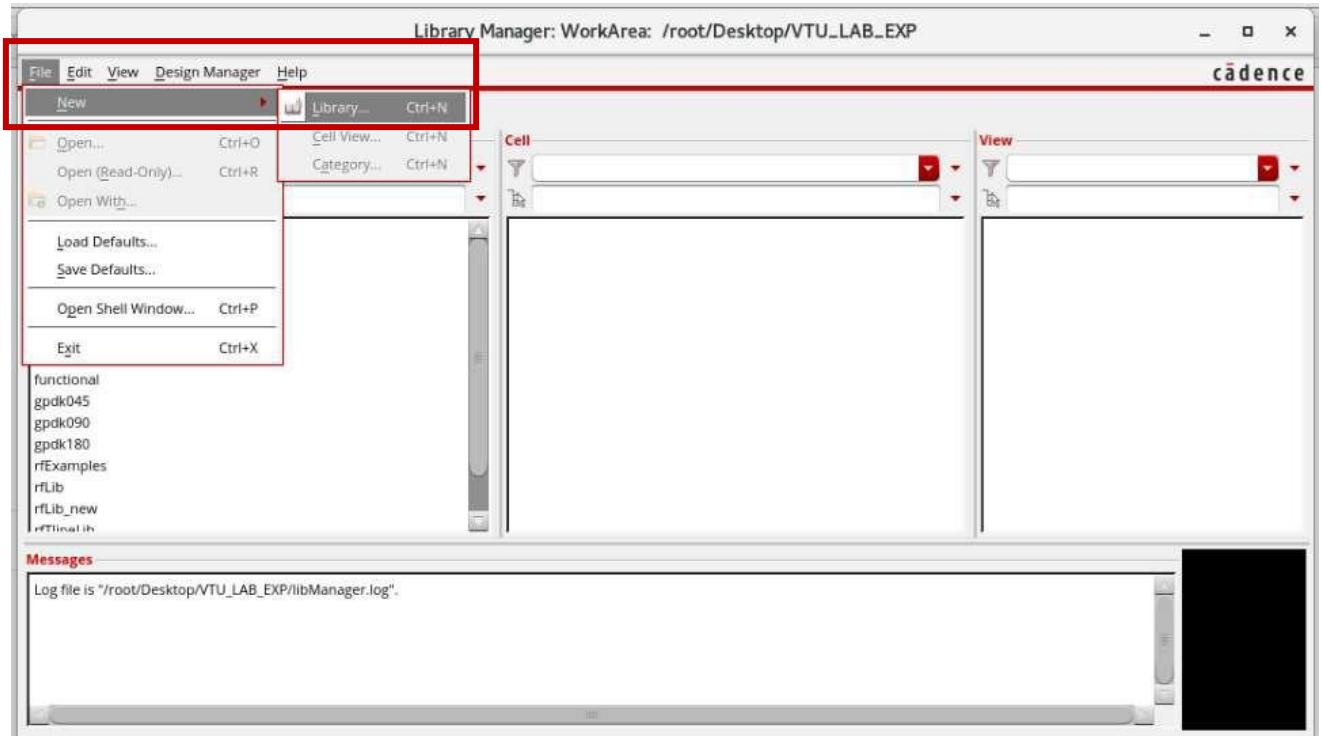


Figure – 1.3: File □ New □ Library

A "**New Library**" window will show up as in Figure – 1.4. Name the Library (for eg: VTU_LAB_MANUAL_180nm) and click on "**OK**".

Figure – 1.4: Name the Library

Select "**Technology File..**" tab that keeps blinking at the bottom of the screen as shown in Figure – 1.5 to map the New Library to a technology node based on the specification.

Figure – 1.5: "Technology File.." Tab

Click on the tab and "**Technology File for New Library**" window can be seen as in Figure –

Select "**Attach to an existing technology library**" and click on "**OK**".

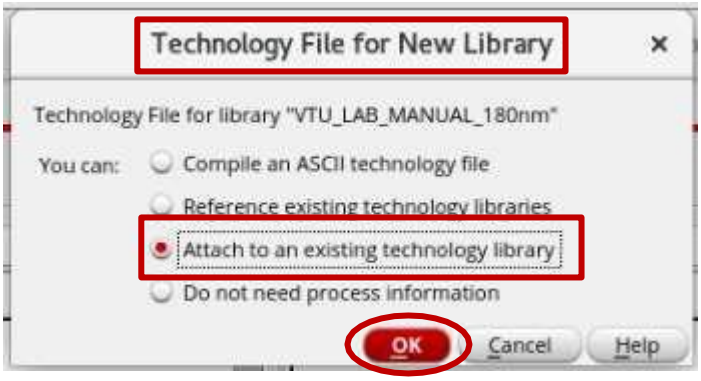


Figure – 1.6: Technology File for New Library form

From the list of available Technology Libraries, select the respective Technology Node as shown in Figure – 1.7 (for example: **gpdk180**) and click on "**OK**".

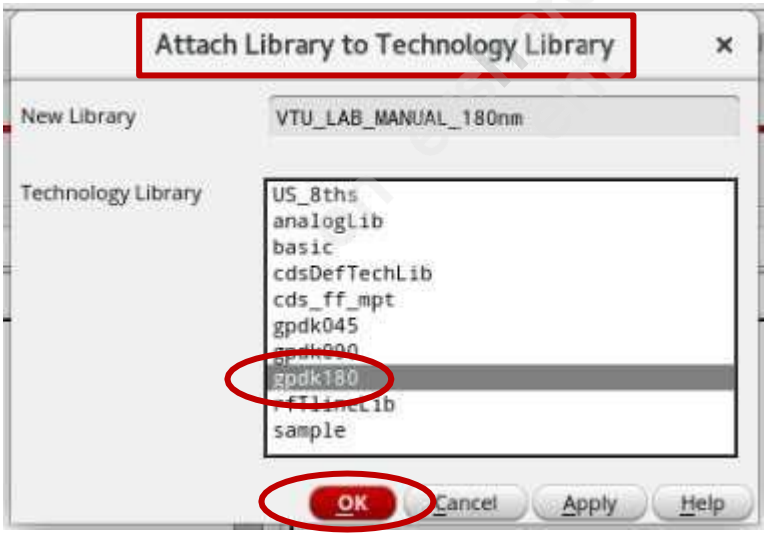


Figure – 1.7: Technology Node Selection

The New Library can be verified from the Library Manager under "**Library**" column as shown in Figure – 1.8.

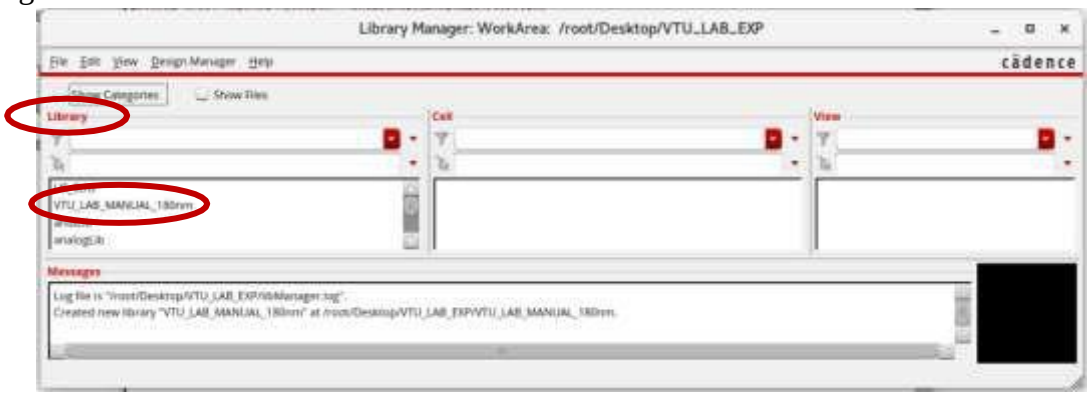


Figure – 1.8: New Library included to Library Manager

CREATE A CELLVIEW:

To create a Cellview within a Library, select the respective library as shown in Figure – 1.9.

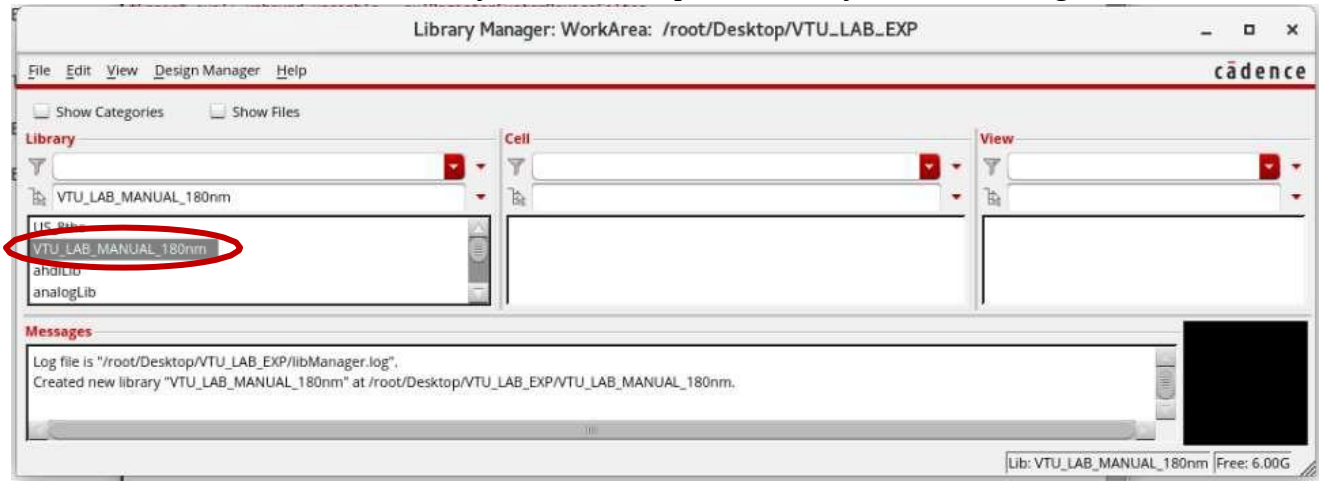


Figure – 1.9: Select the Library

Select **File → New → Cell View** as shown in Figure – 1.10.

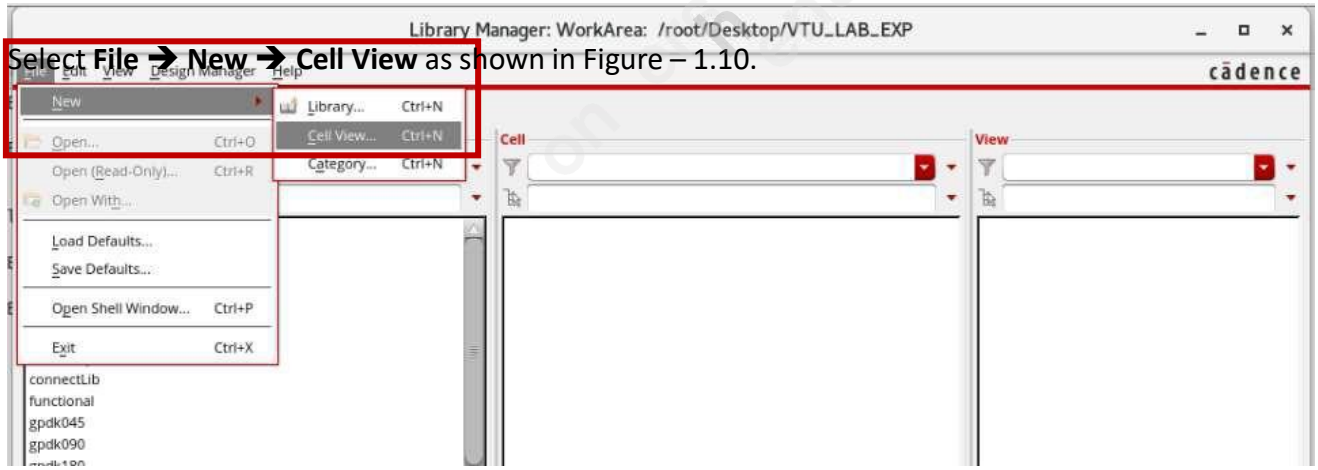


Figure – 1.10: File → New → Cell View

A "**New File**" window can be seen as shown in Figure – 1.11.

Figure – 1.11: "New File" Window

Name the Cell and click on "**OK**". A blank "**Virtuoso Schematic Editor L Editing**" window can be seen as shown in Figure – 1.12.

(i) SCHEMATIC CAPTURE FOR THE CMOS INVERTER

To complete the Schematic for a CMOS Inverter with $W_N = W_P$, components have to be included to the blank Virtuoso Schematic Editor window. These components are called Instances. The procedure to include the components to the Schematic are given below.

ADD AN INSTANCE:

Select "**Create Instance**" as in Figure – 1.13 (or) use the bind key '**I**' (or) the icon as in Figure – 1.13.

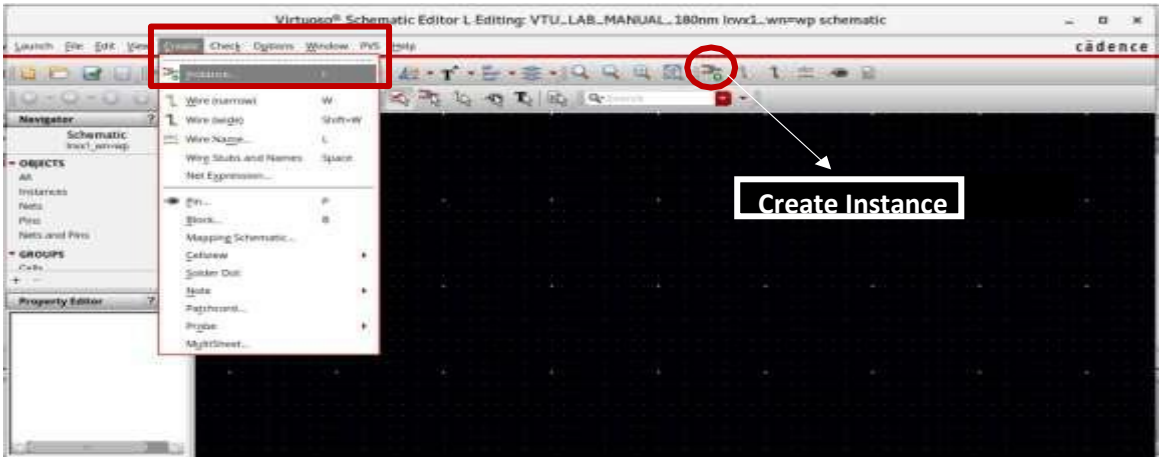


Figure – 1.13: Create Instance

The "**Add Instance**" form can be seen as shown in Figure – 1.14.

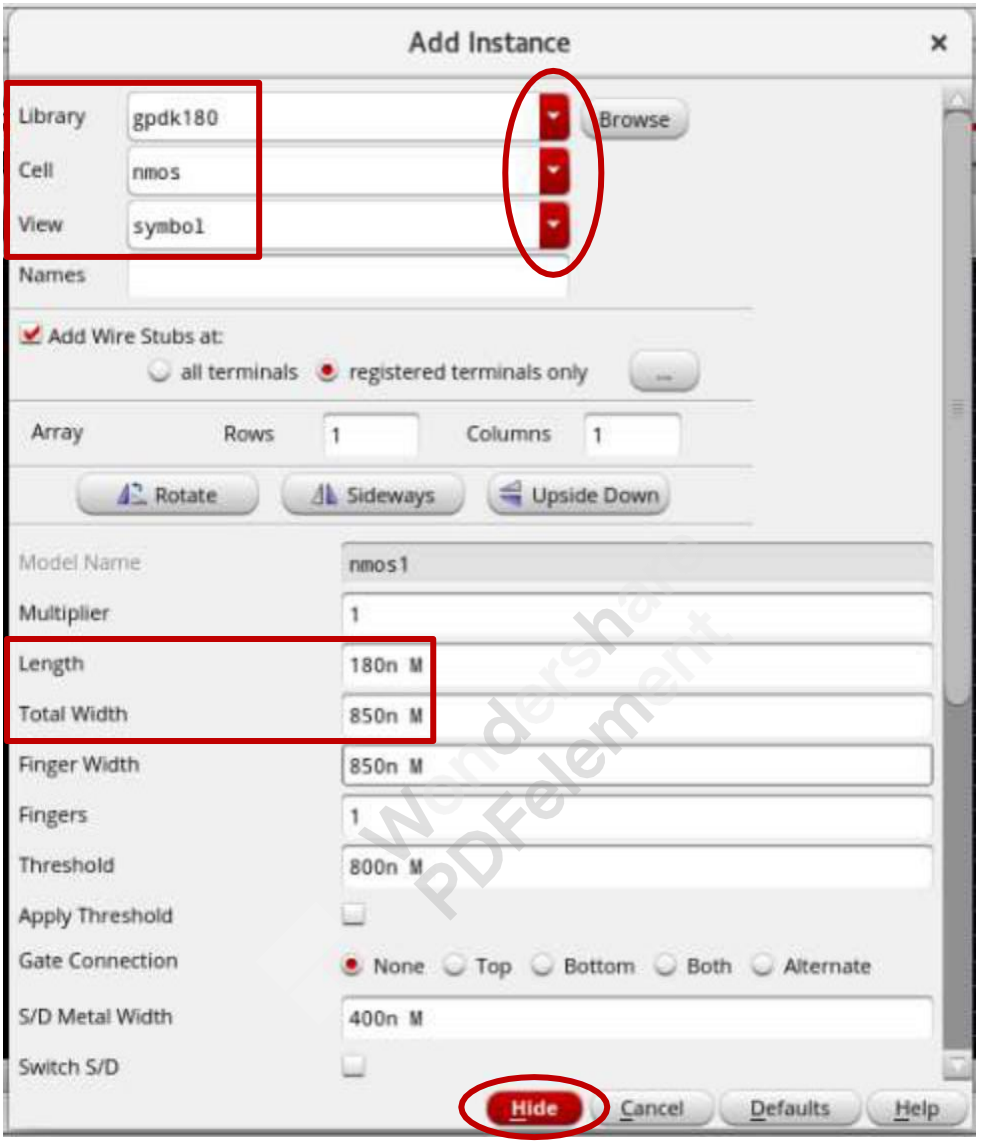


Figure – 1.14: "Add Instance" Window

Click on the drop down close to the **Browse** option as shown in Figure – 1.14. Select the Technology Node from the list of libraries. Similarly, click on the drop down next to **Cell** and select the required device from the list. For the CMOS Inverter circuit, PMOS and NMOS transistors are required. The parameters for the devices as given in the requirement are considered as in Table – 1, Table – 2 and Table – 3.

Table – 1: Length and Width of NMOS and PMOS Transistors for the condition $W_N = W_P$

Library Name	Cell Name	Comments / Properties
gpdk180	Nmos	Width, W_N = 850 n Length, L = 180 n
gpdk180	Pmos	Width, W_P = 850 n Length, L = 180 n

Table – 2: Length and Width of NMOS and PMOS Transistors for the condition $W_N = 2 * W_P$

Library Name	Cell Name	Comments / Properties
gpdk180	Nmos	Width, W_N = 850 n Length, L = 180 n
gpdk180	Pmos	Width, W_P = 1.7 u Length, L = 180 n

Table – 3: Length and Width of NMOS and PMOS Transistors for the condition $W_N = W_P / 2$

Library Name	Cell Name	Comments / Properties
gpdk180	Nmos	Width, W_N = 850 n Length, L = 180 n
gpdk180	Pmos	Width, W_P = 425 n Length, L = 180 n

Type the parameters and click on "**Hide**". The device can be seen as shown in Figure – 1.15.

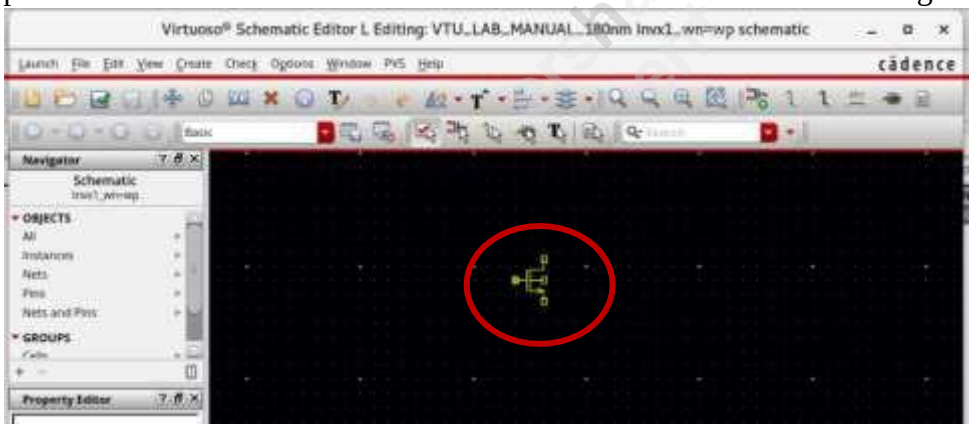


Figure – 1.15: Instance after Selection

Make a left mouse click to place it on the Schematic Editor. The device after placement on the Schematic Editor can be seen as shown in Figure – 1.16. Similarly, other components can be instantiated.

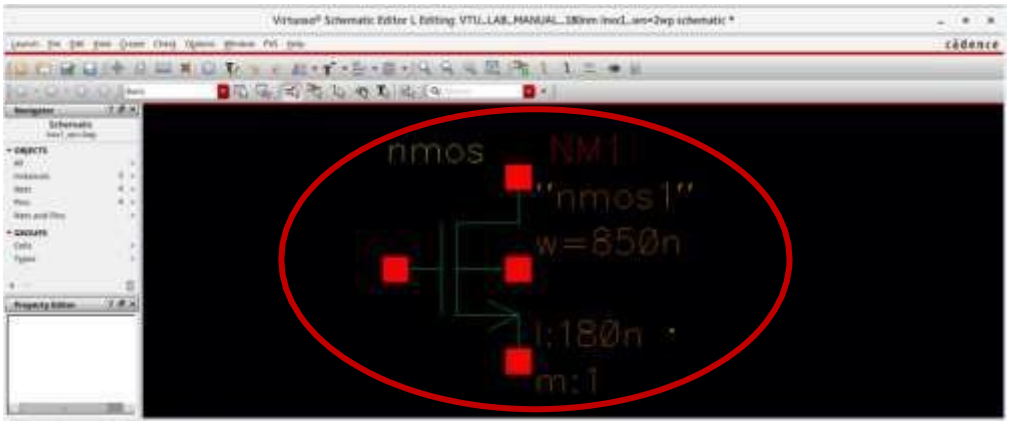


Figure – 1.16: Instance after left mouse click

ADD PIN:

To include pins to the schematic, select "**Create** □ **Pin**" from the top menu (or) use the bindkey '**P**' (or) use the icon from the top menu as shown in Figure – 1.17.

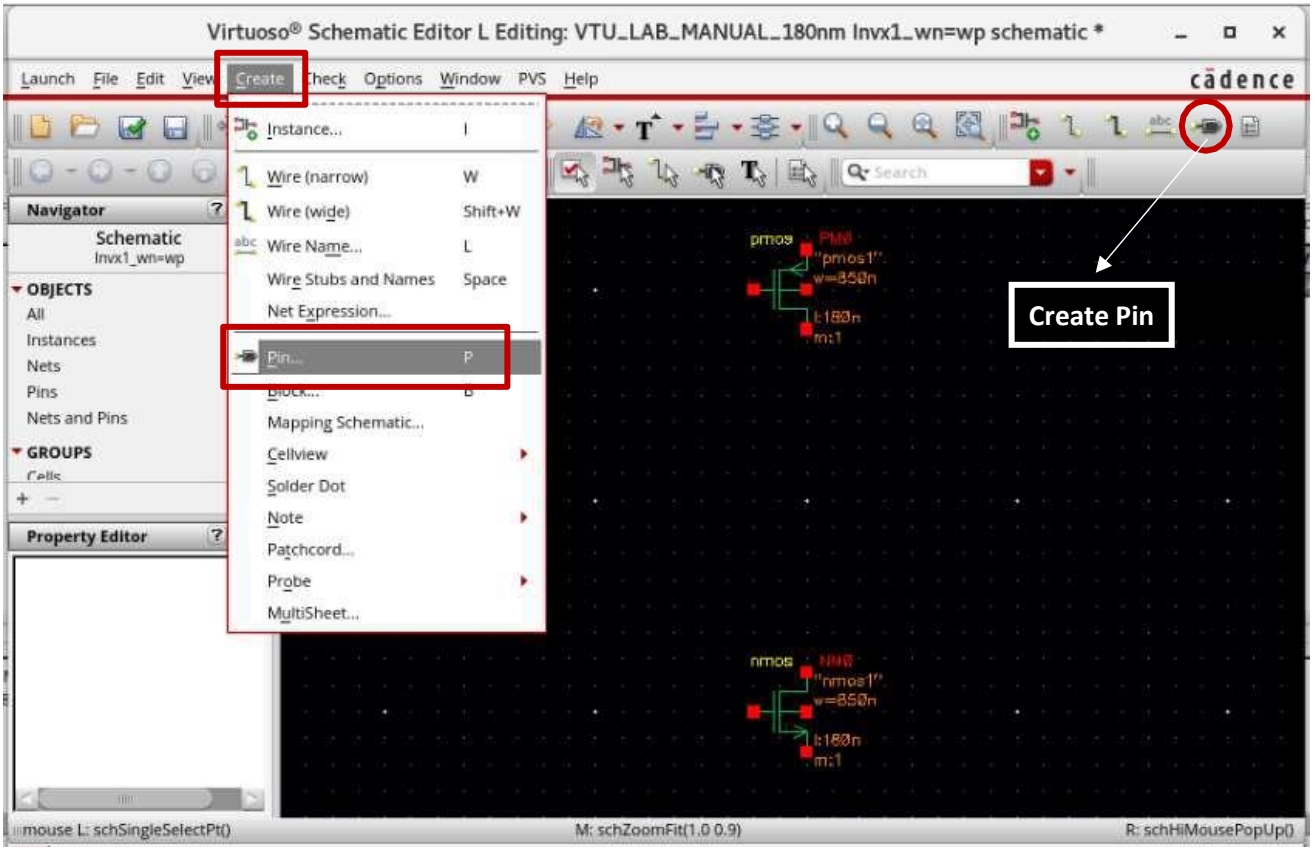


Figure – 1.17: Create □ Pin

The "**Create Pin**" window pops up as shown in Figure – 1.18.

Figure – 1.18: Create Pin window

Name the pins by separating them with "**space"**, choose its direction and click on "**Hide**" as shown in Figure – 1.19(a) and Figure – 1.19(b).

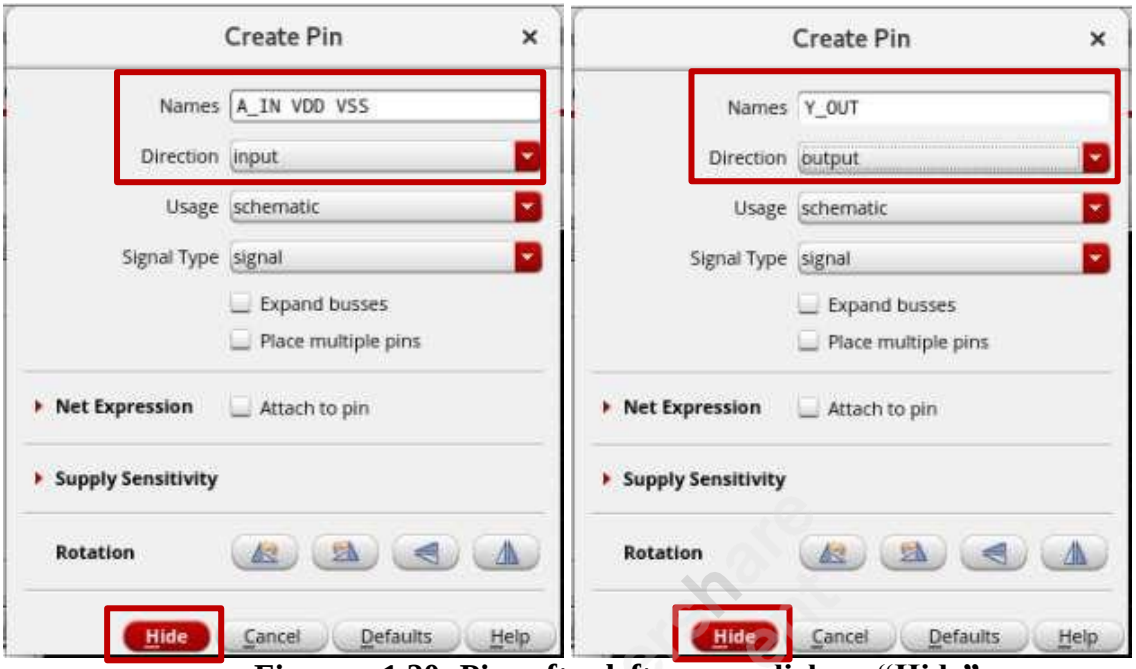
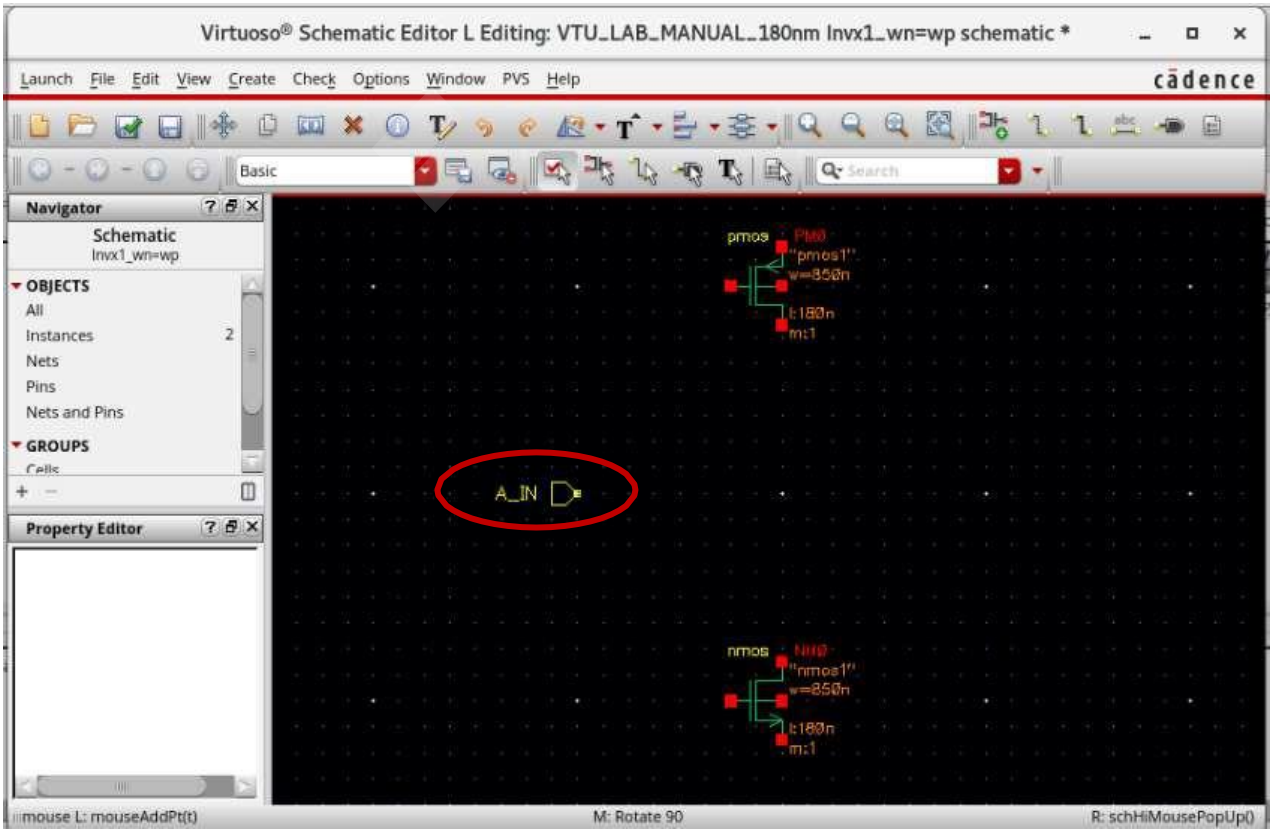


Figure – 1.20: Pins after left mouse click on "Hide"



Place the pins on the Schematic Editor using a left mouse click and the pins after placement can be visualized as shown in Figure – 1.21.

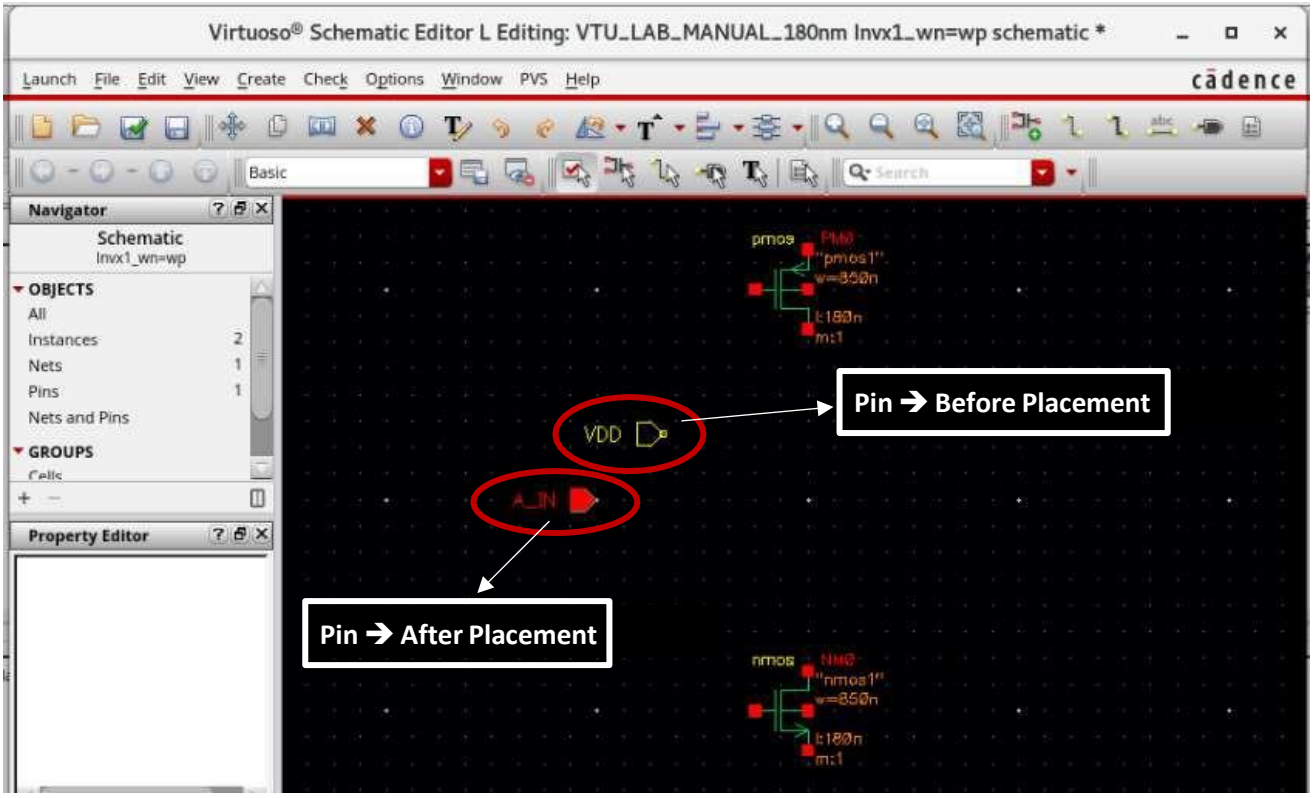


Figure – 1.21: Pins Before and After Placement

Use the bind key "**R**" to rotate the pins and it can be done either before or after Pin Placement. The direction of the pins before and after rotation are shown in Figure – 1.22.

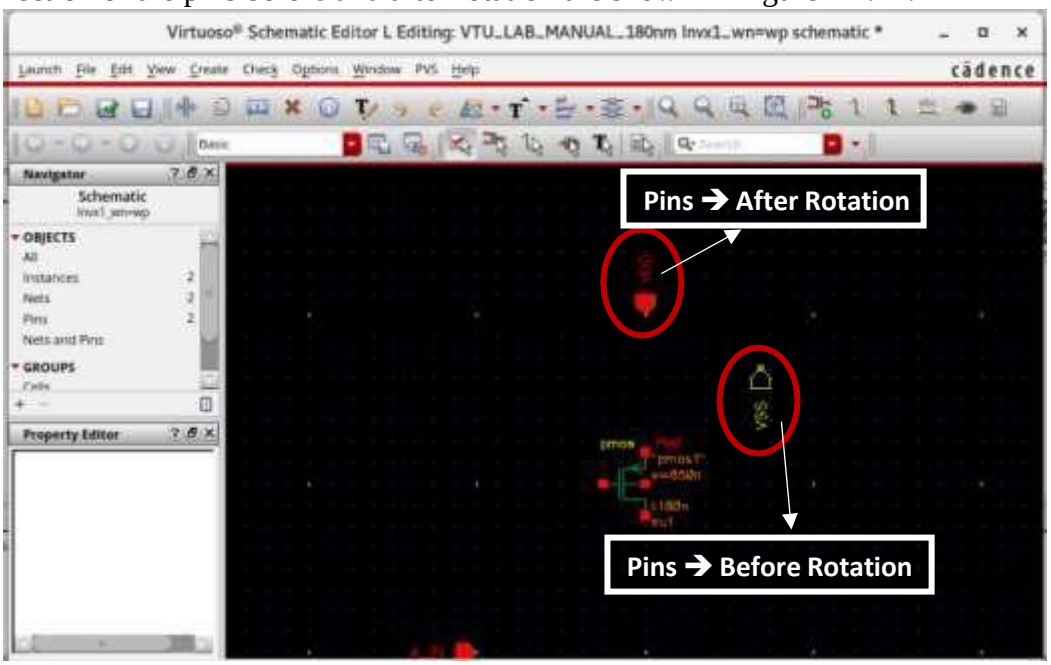


Figure – 1.22: Pins Before and After Rotation

The Schematic Editor window after pin placement is shown in Figure – 1.23.

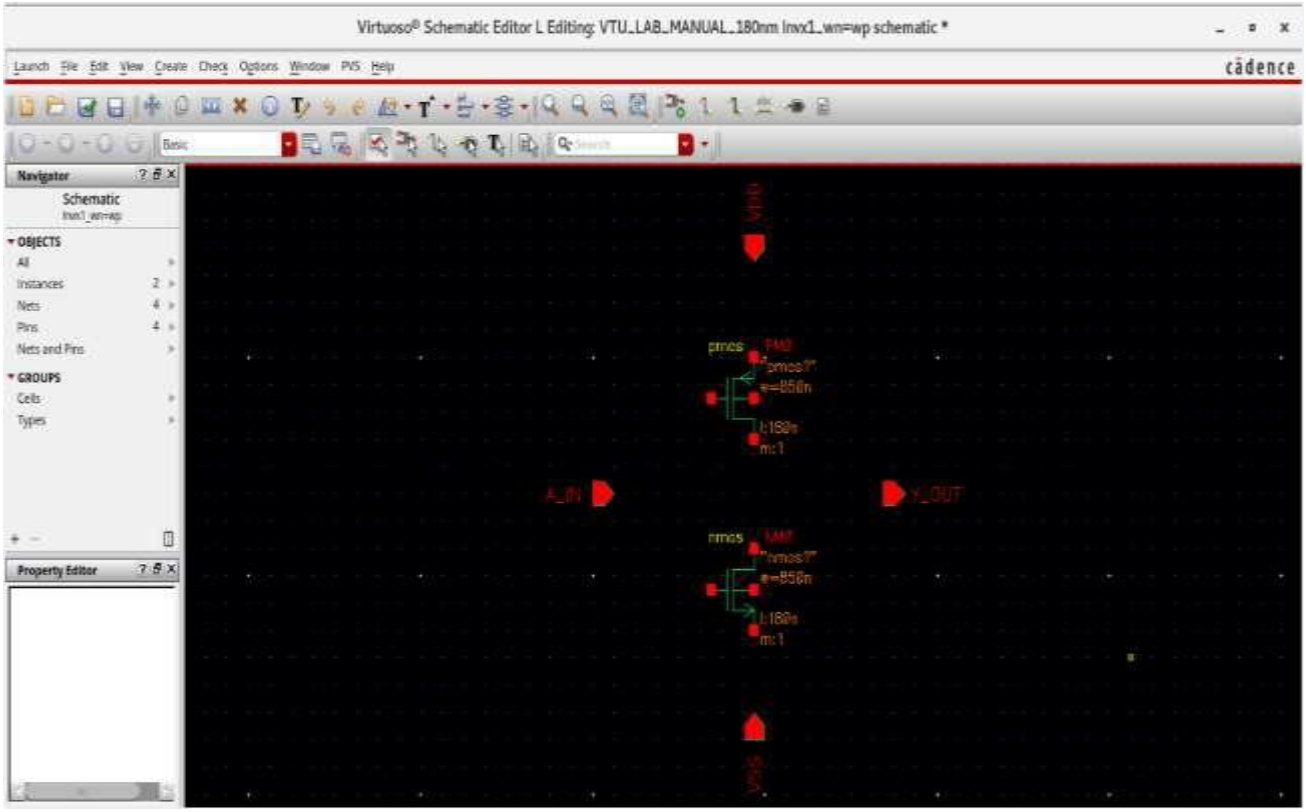


Figure – 1.23: Schematic Editor after pin placement

ADD WIRE:

For connecting the pins and the terminals, click on "**Create** □ **Wire**" from the top menu (or) use the bind key '**W**' (or) the icon from the top menu as shown in Figure – 1.24.

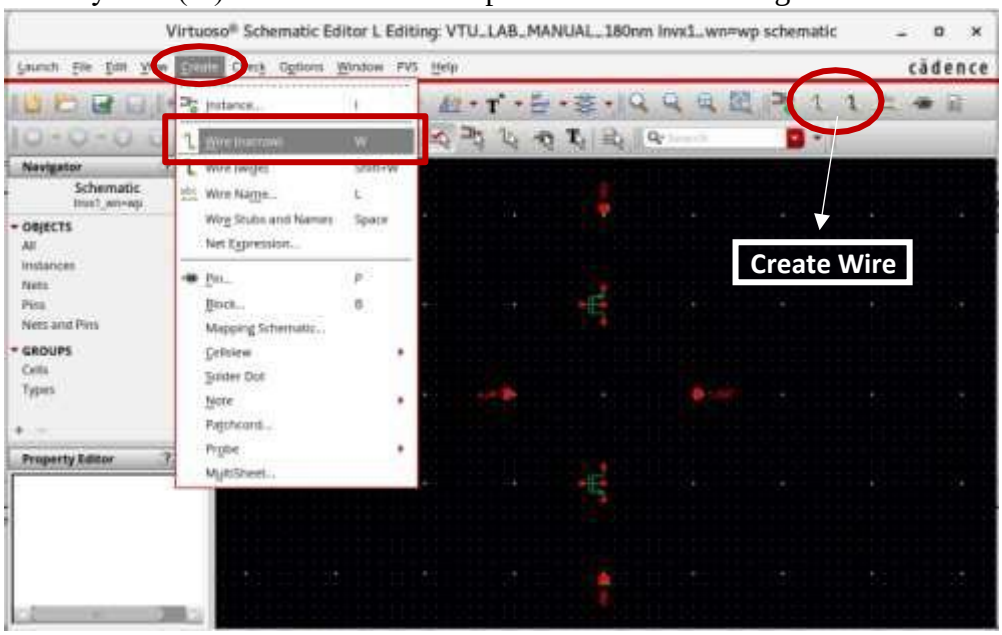


Figure – 1.24: Create □ Wire

Use the left mouse click to start / complete the wire from one terminal / pin to another. The complete Schematic after connecting the pins and terminals for all the three conditions $W_N = W_P$, $W_N = 2 * W_P$, $W_N = W_P / 2$ is shown in Figure – 1.25(a), 1.25(b) and 1.25(c) respectively.

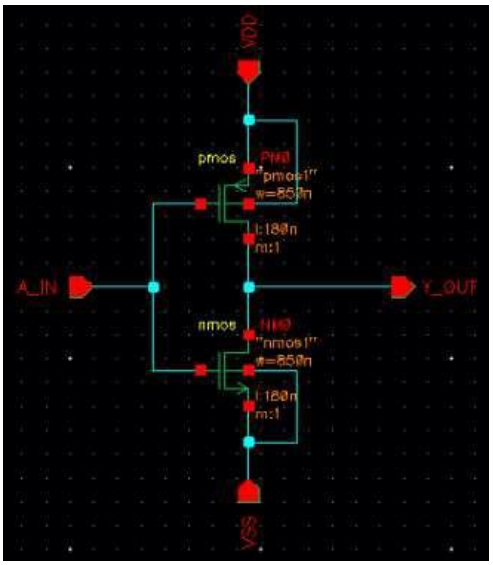


Figure – 1.25(a): $W_N = W_P$

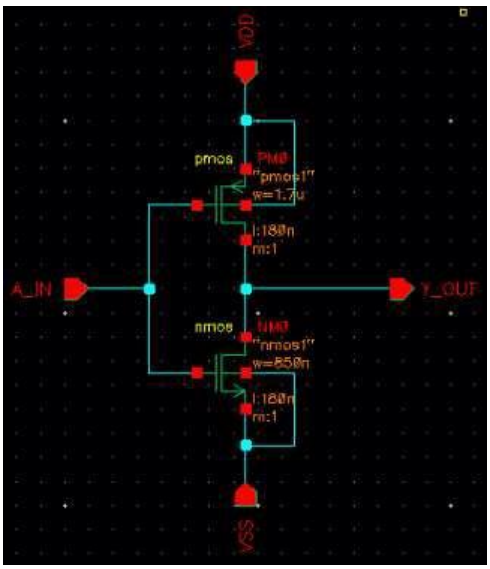
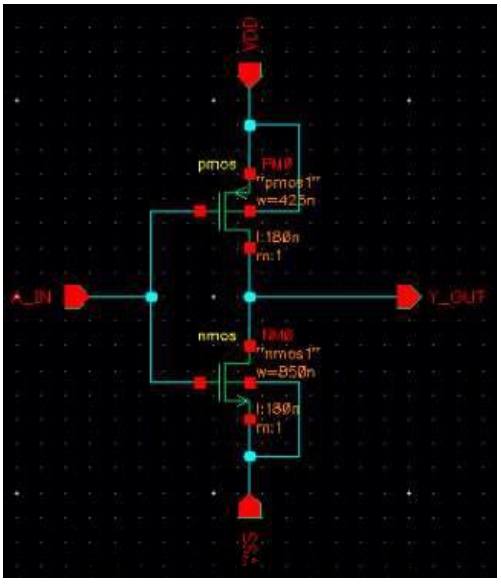


Figure – 1.25(b): $W_N = 2 * W_P$



CHECK AND SAVE THE DESIGN:

It is mandatory to save the design before we move ahead to the Simulation and there are two options, "**Save**" and "**Check and Save**" as in Figure – 1.26.

Figure – 1.26: "Check and Save" and "Save" option

"Save" option saves the design as it is and **"Check and Save"** option checks for discontinuities like floating net or terminal and provides the "error" or "warning" messages accordingly and then saves the design. Sample message can be seen in the "**Command Interpreter Window**" as shown in Figure – 1.27.

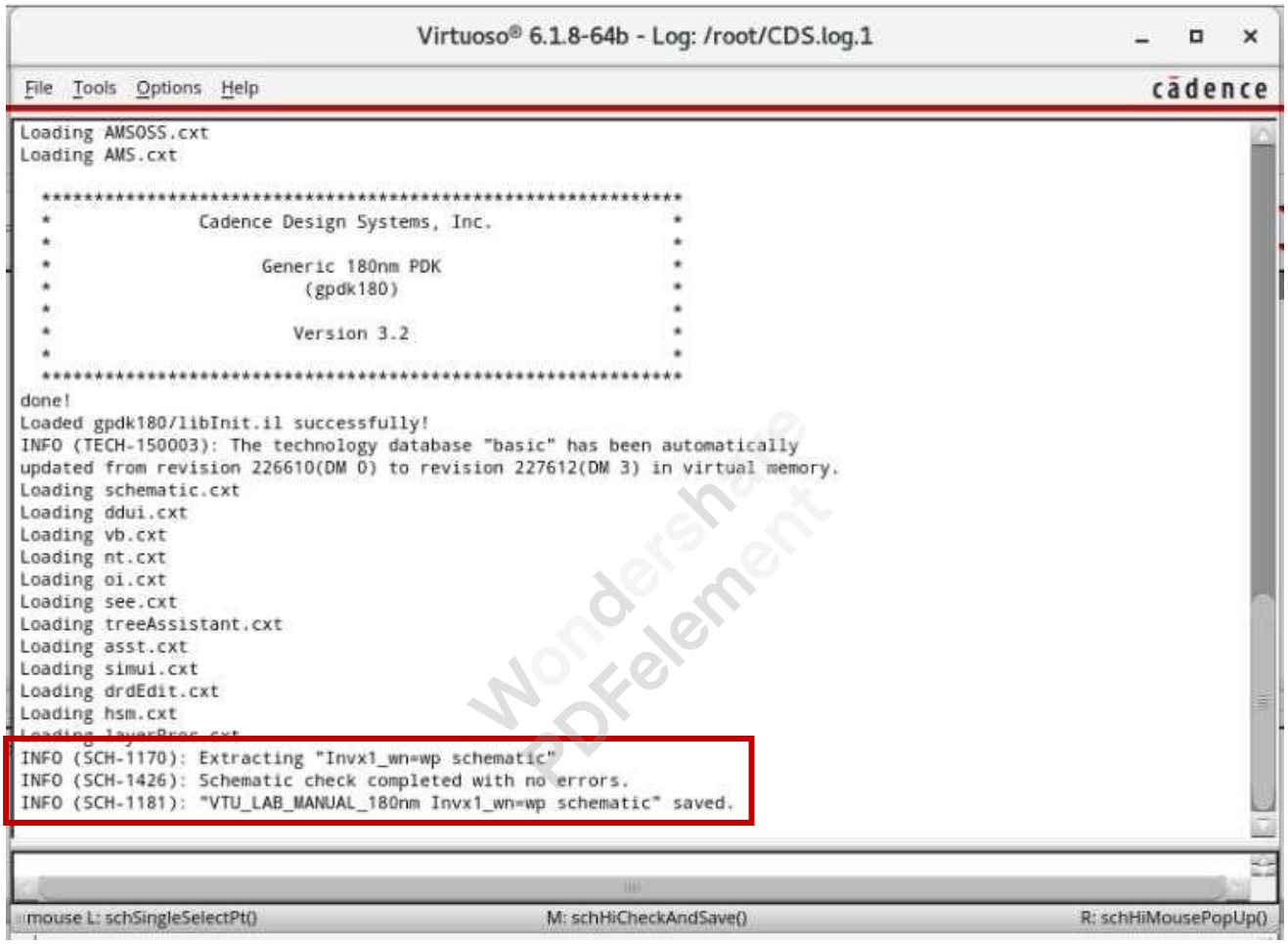
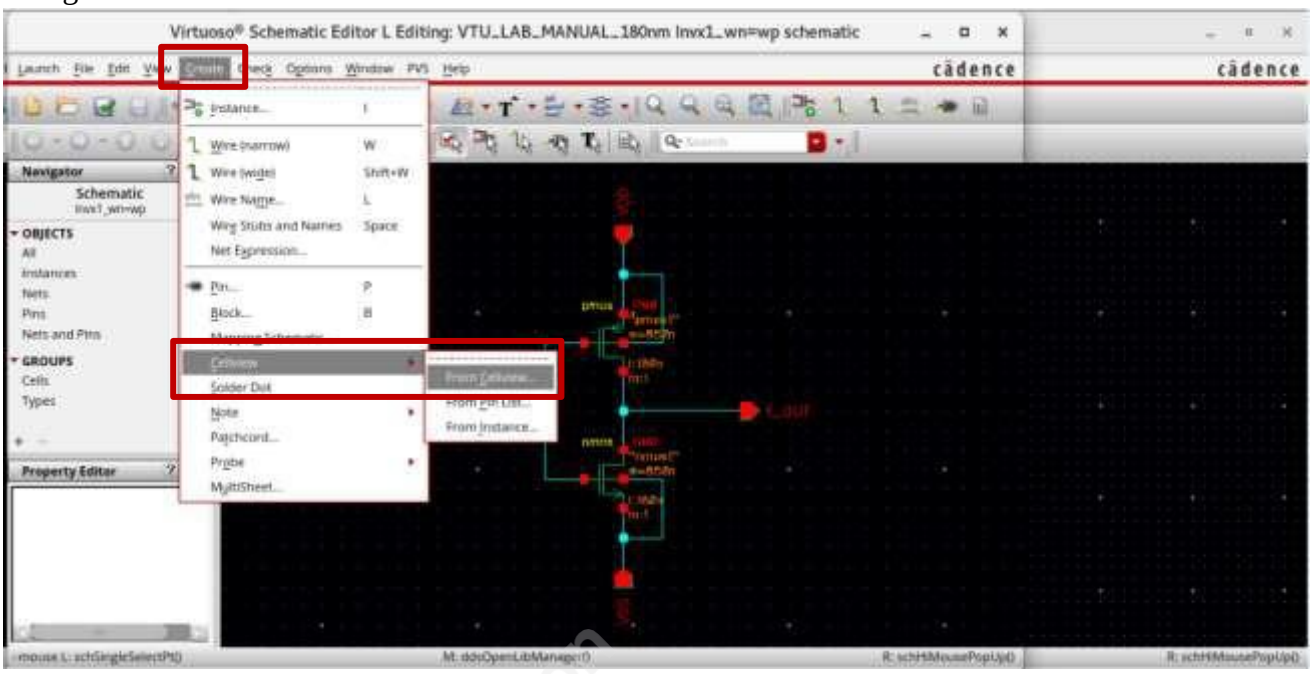


Figure – 1.27: Message after selecting "Check and Save"

SYMBOL CREATION:

A Symbol view is very important in a design process to make use of a Schematic in a hierarchy. To create a symbol, select "**Create □ Cellview □ From Cellview**" from the top menu as shown in Figure – 1.28.



Verify the Library Name, Cell Name, From View Name, To View Name, etc., as shown in Figure – 1.29 and Click on "**OK**".

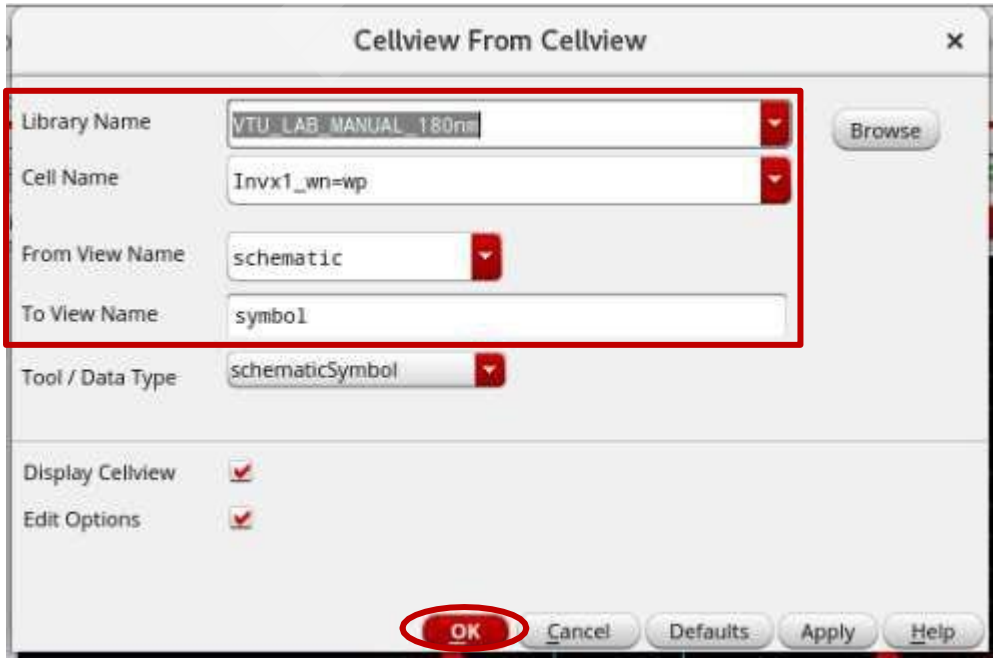
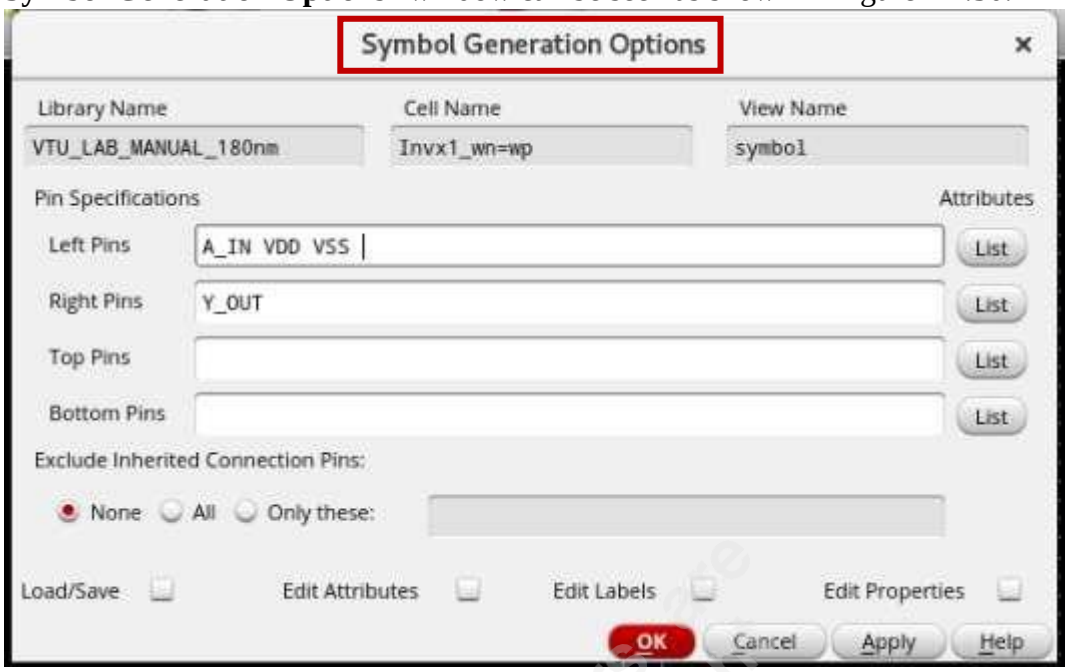


Figure – 1.29: "Cellview From Cellview" window

The "**Symbol Generation Options**" window can be seen as shown in Figure – 1.30.



The pin location on the symbol can be fixed using the options **Left Pins, Right Pins, Top Pins and Bottom Pins**. Assign the pins and click on '**OK**' as shown in Figure – 1.31.

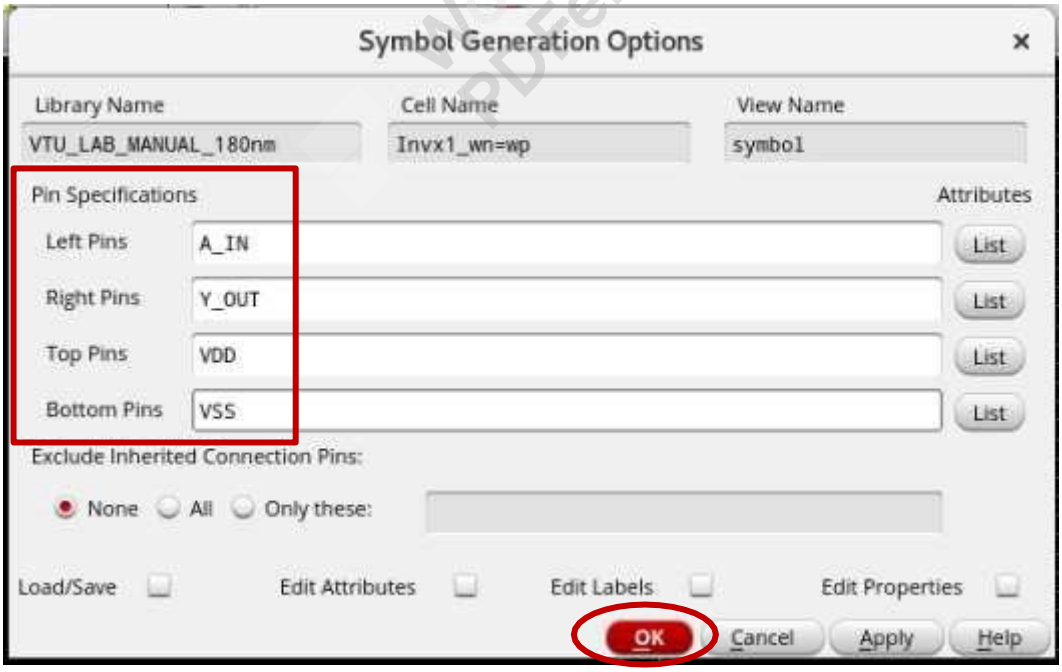


Figure – 1.31: "Symbol Generation Options" window

The "**Virtuoso Symbol Editor**" window pops up with a default symbol based on the Pin Assignment as shown in Figure – 1.32.

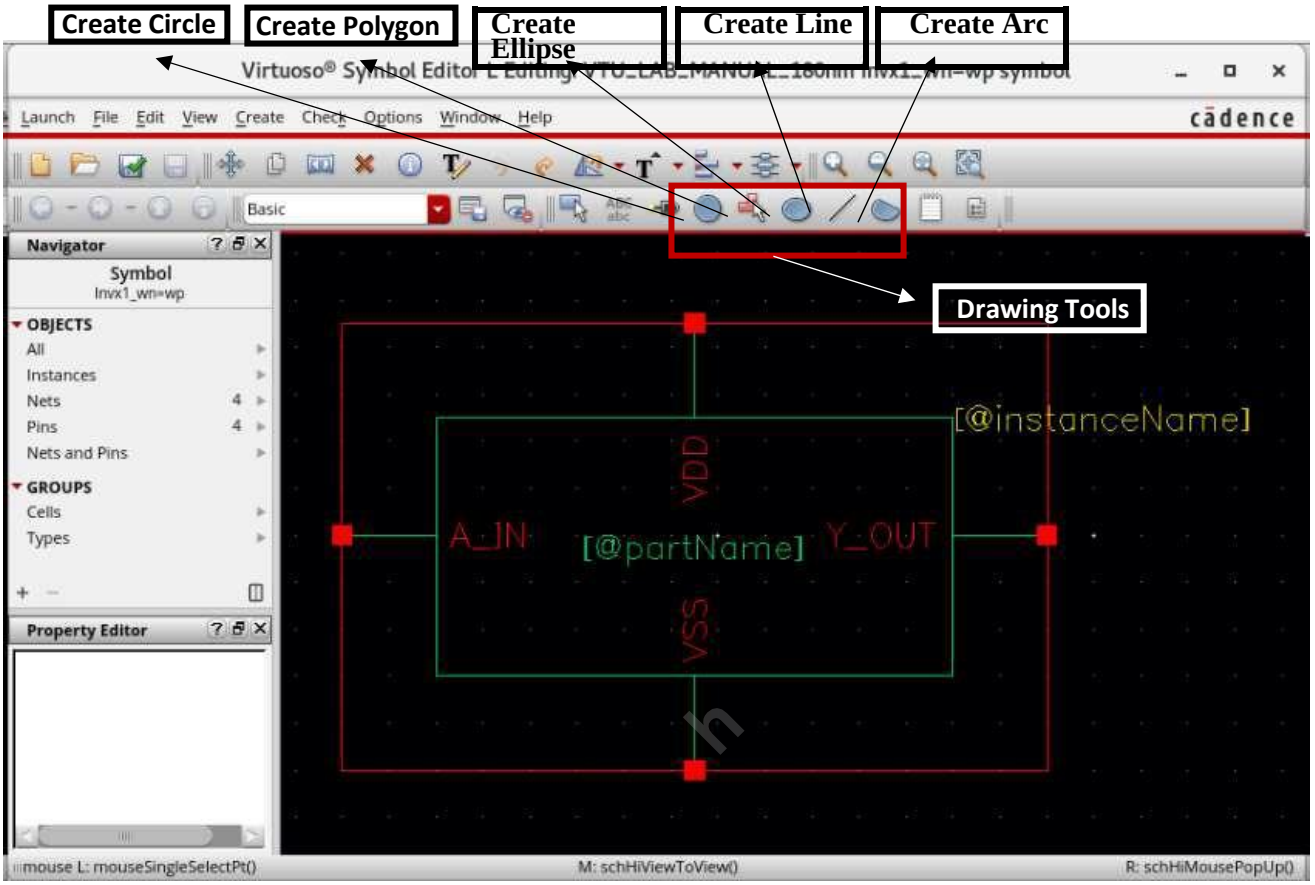


Figure – 1.32: "Virtuoso Symbol Editor" window with default symbol

SYMBOL MODIFICATION:

The symbol can be modified using the drawing tools from the top menu as shown in Figure – 1.32.

To modify the symbol, remove the inner rectangle (green), highlighted in Figure – 1.33(a). To remove the inner rectangle (green), place the mouse pointer within and make a left mouse click to select the entire rectangle as shown in Figure – 1.33(b). Click on 'Delete' in the keyboard to remove the rectangle as shown in Figure – 1.33(c).

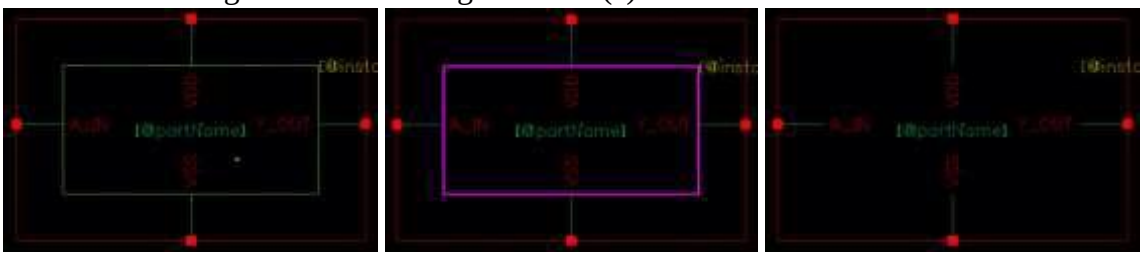


Figure – 1.33(a) **Figure – 1.33(b)** **Figure – 1.33(c)**

Figure – 1.33(a): Inner Rectangle Highlighted, Figure – 1.33(b): Inner Rectangle Selected, Figure – 1.33(c): Inner Rectangle Deleted

Since the focus is to design an Inverter, to create a triangle, use the "**Create Line**" option as shown in Figure – 1.32. Use the same procedure as "wiring the schematic" to create the triangle.

The symbol, after creating the triangle can be seen as shown in the Figure – 1.34.

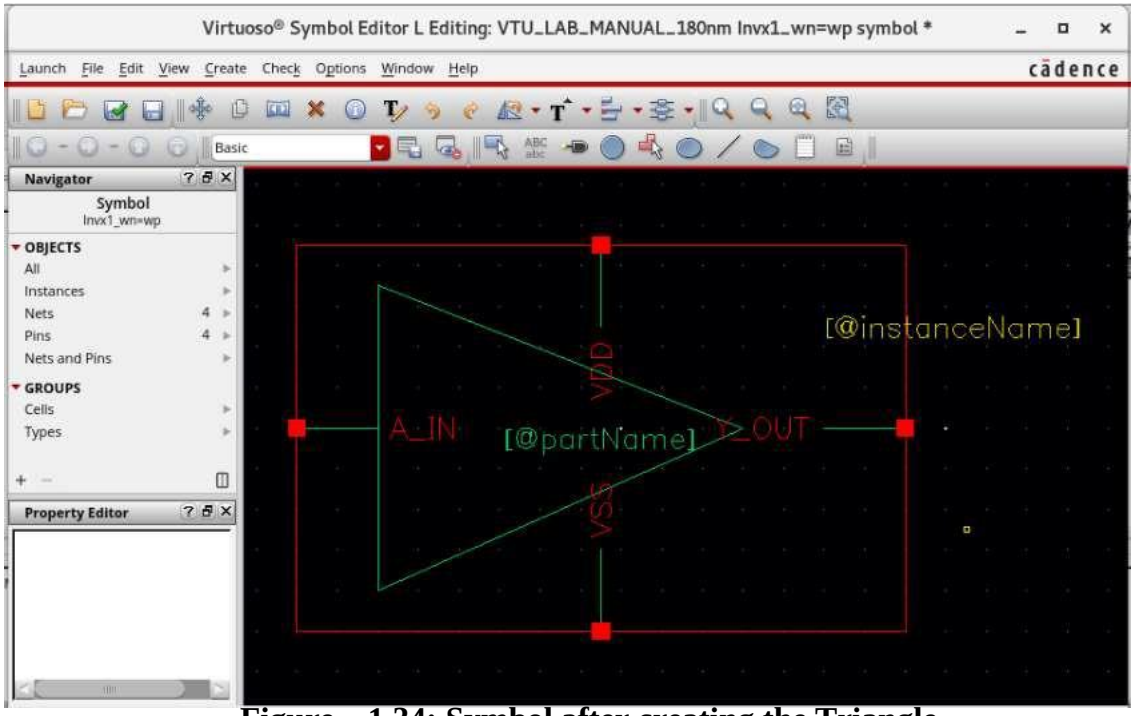


Figure – 1.34: Symbol after creating the Triangle

To create a bubble, use "**Create Circle**" option as shown in Figure – 1.32, place the mouse pointer
at the center between the 'Triangle' and the 'Output Pin', make a left mouse click and expand the circle and make a left mouse click to fix its size as shown in Figure – 1.35. Click on "**Check and Save**" option to '**Save**' the symbol.

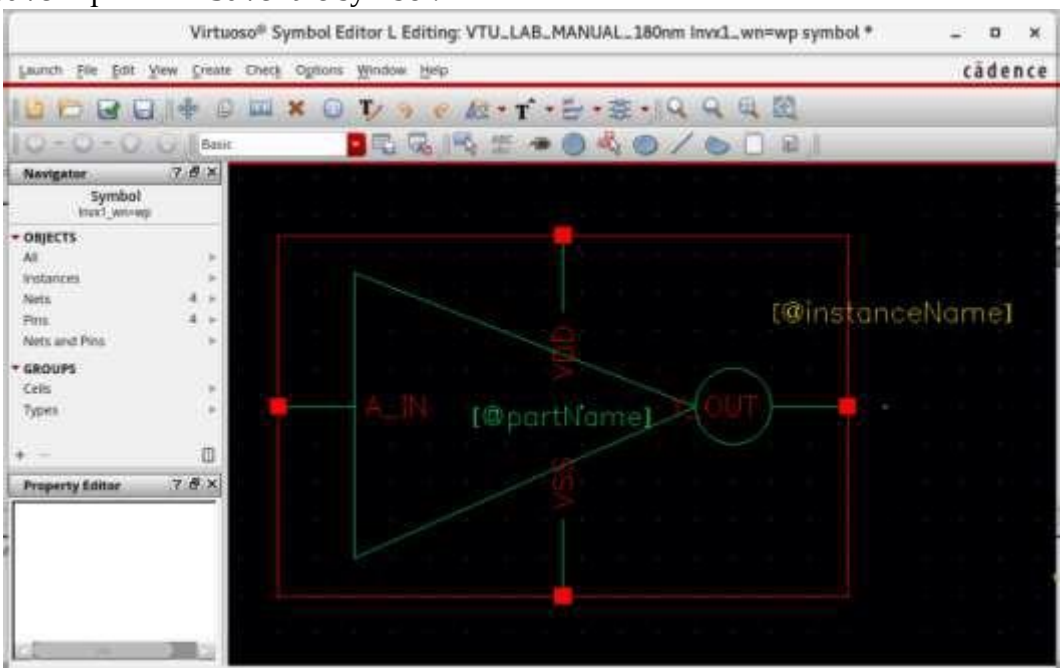
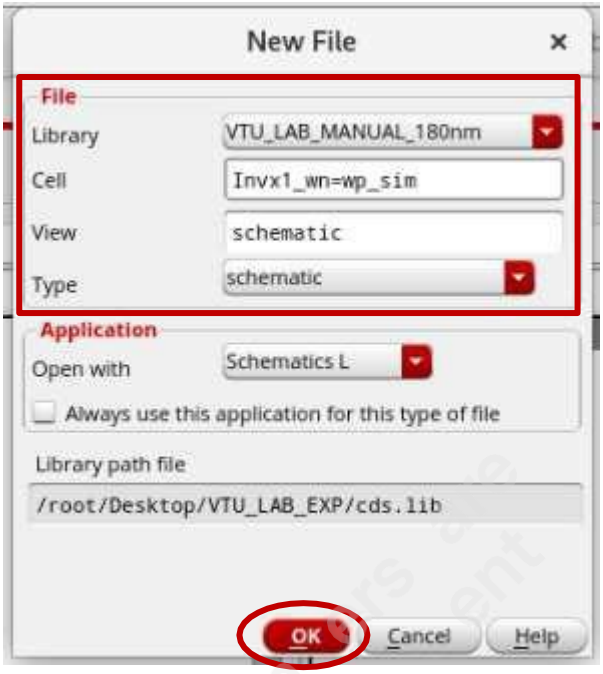


Figure – 1.35: Customized Symbol of an Inverter

(1) TEST CIRCUIT FOR SIMULATION:

The Test Circuit can be created using the symbol created in the previous section. To create a test circuit, create a **"New Cellview"** with a different **"Cell Name"** as shown in Figure – 1.36.



Use the "**Add Instance**" option, select the respective Library, Cell and View as in Figure – 1.37 to instantiate the symbol.

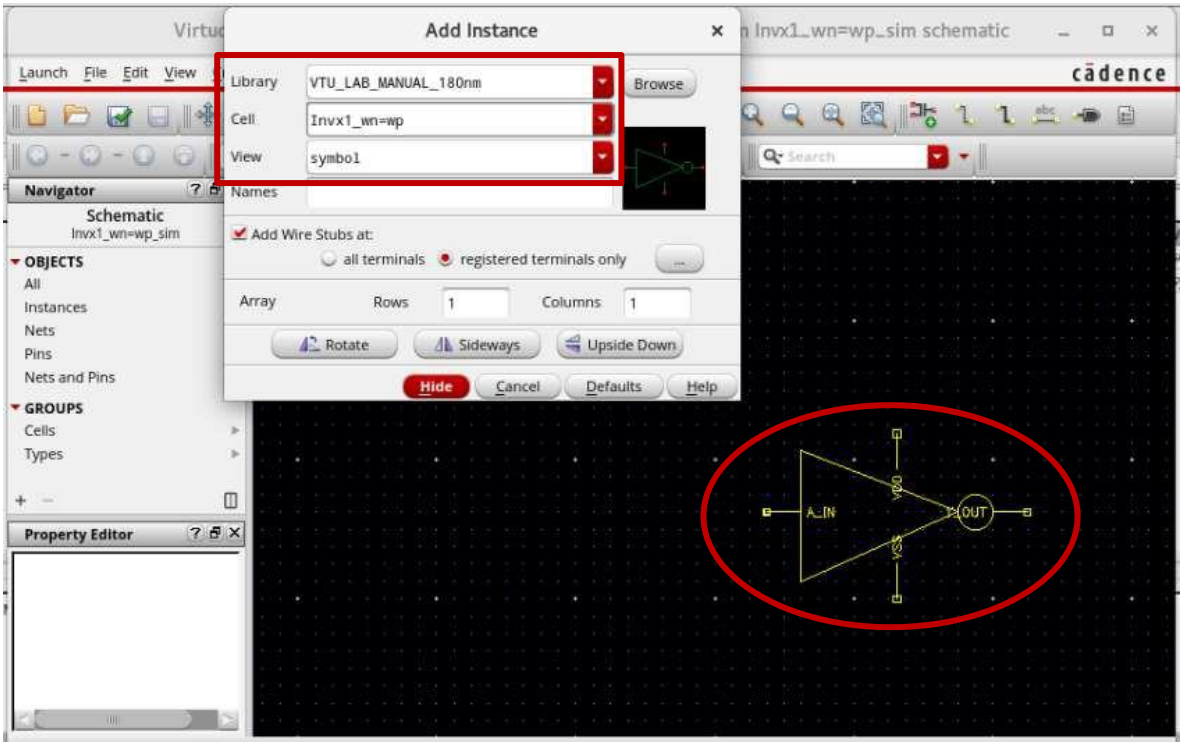


Figure – 1.37: Symbol Instantiation for the Test Circuit

The remaining devices to be included on the Schematic and its properties are given below in Table - 4.

Table – 4: Properties of vdc, vpulse, cap and gnd

Library Name	Cell Name	Comments / Properties
analogLib	Vdc	DC voltage = 1.8 V
analogLib	Vpulse	Voltage 1 = 0 V, Voltage 2 = 1.8 V, Period = 20n s, Delay time = 10n s, Rise time = 1p s, Fall time = 1p s, Pulse width = 10n s
analogLib	Cap	Capacitance = 100f F
analogLib	Gnd	

The screenshot of the device properties for the instances vdc, vpulse, cap and gnd are shown in Figure – 1.38, Figure – 1.39, Figure – 1.40 and Figure – 1.41. The complete Test Schematic after wiring is shown in Figure – 1.42.

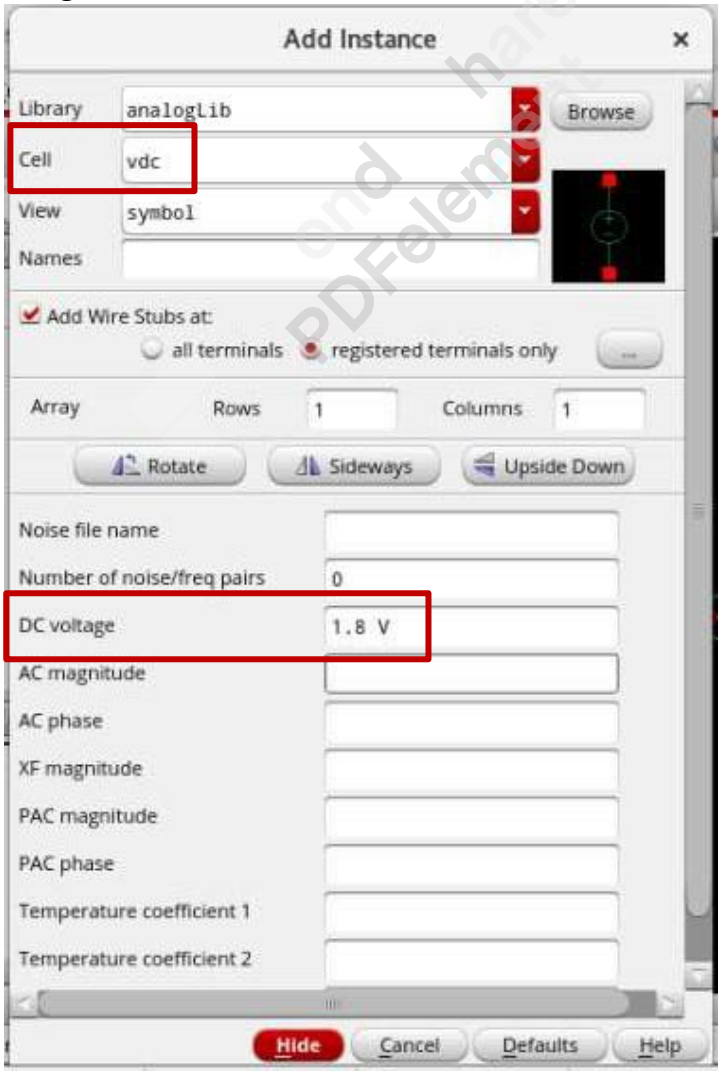


Figure – 1.38: Instantiating "vdc"

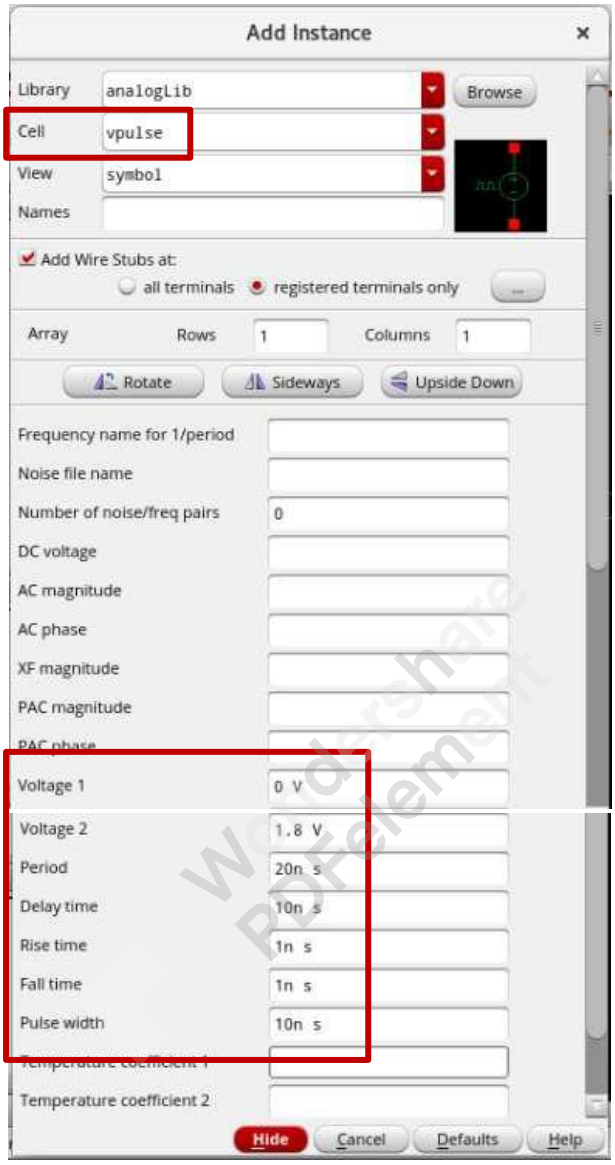


Figure – 1.39: Instantiating "vpulse"

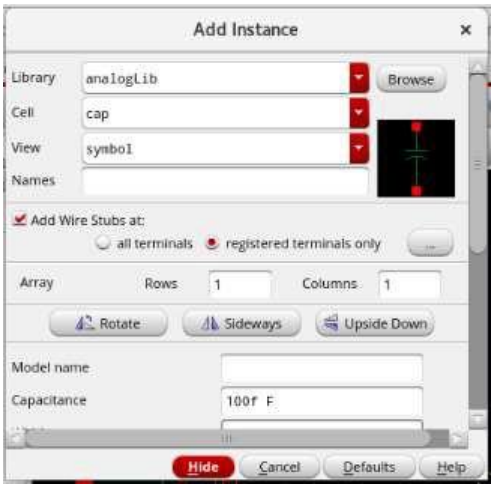


Figure – 1.40: Instantiating "cap"

Figure – 1.41: Instantiating "gnd"

The complete circuit after instantiating all the devices and interconnections is shown in Figure – 1.42.

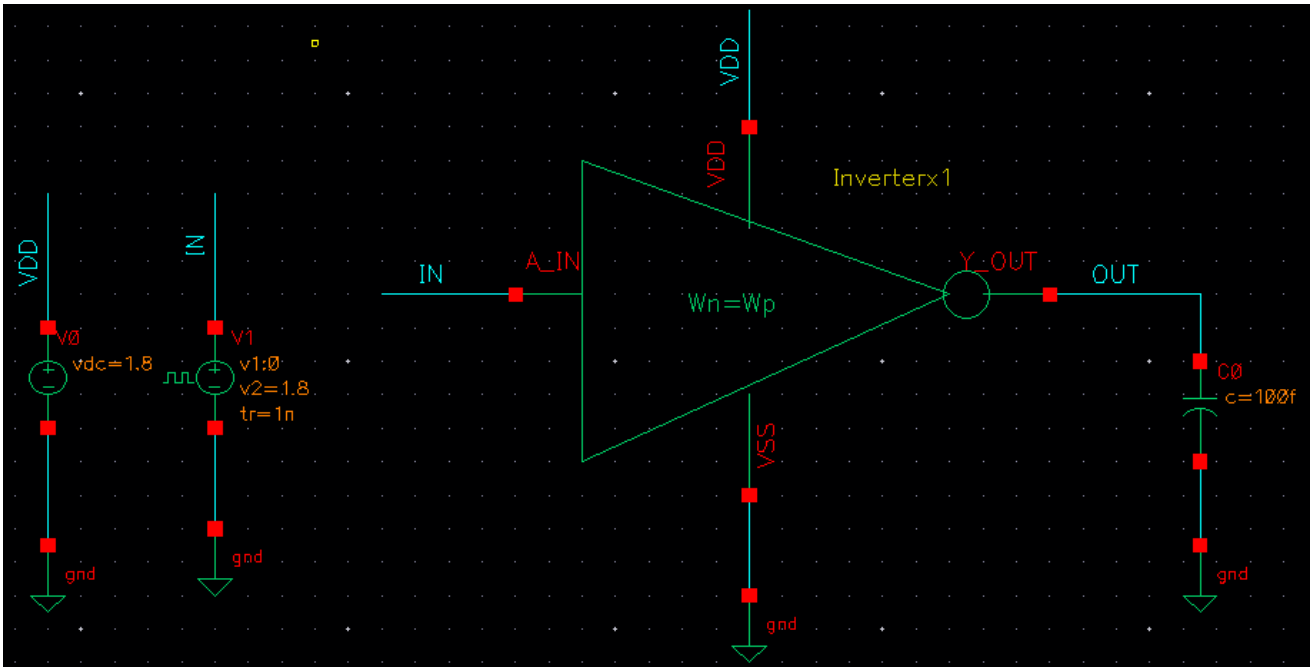


Figure – 1.42: Complete Test Schematic

To **Label** the nets, click on "**L**" in the keyboard. The "**Create Wire Name**" window pops up as shown in Figure – 1.43. Name the nets, different net names can be mentioned at the same instance of time by separating them with "**Spaces**", same net names can also be repeated as per the requirement and click on "**Hide**" as shown in Figure – 1.44.

Figure – 1.43: Create Wire Name Window

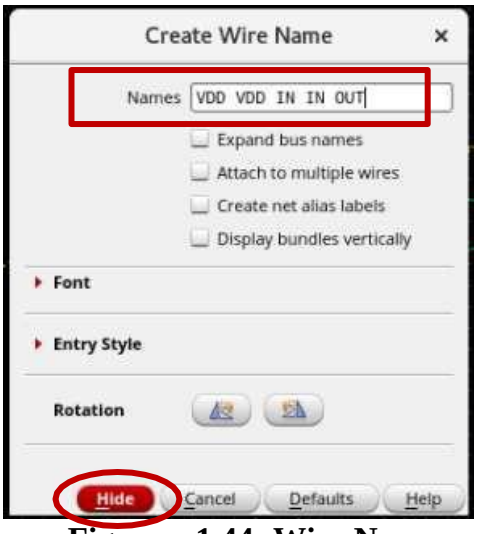


Figure – 1.44: Wire Names

The "**Wire Name before placement**" can be seen in Figure – 1.45. The "**Dot**" just under the wire name has to be placed over the "**wire**" and make a left mouse click to fix it. The "**Placed Wire Name**" can be seen in Figure – 1.45.

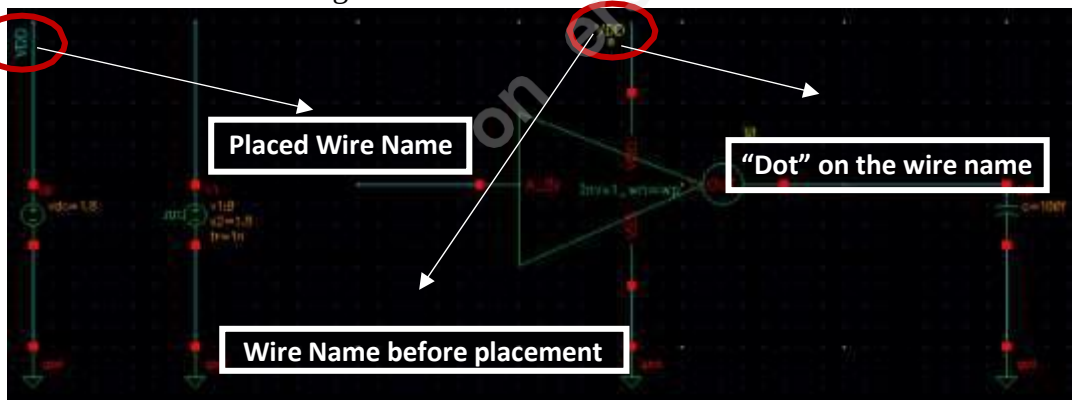


Figure – 1.45: Wire Name before and after its Placement

The complete schematic after placing all the wire names is shown in Figure – 1.46. "**Check and Save**" the Test Schematic.

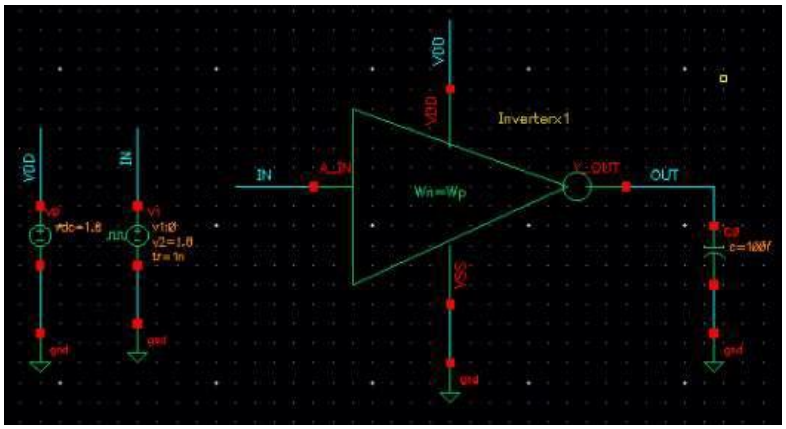


Figure – 1.46: Complete Test Schematic

FUNCTIONAL SIMULATION WITH SPECTRE:

To simulate the design and perform the DC Analysis and Transient Analysis for the CMOS Inverter, click on **"Launch ADE L"** from the top menu of the Test Schematic Cellview as shown in Figure – 1.47.

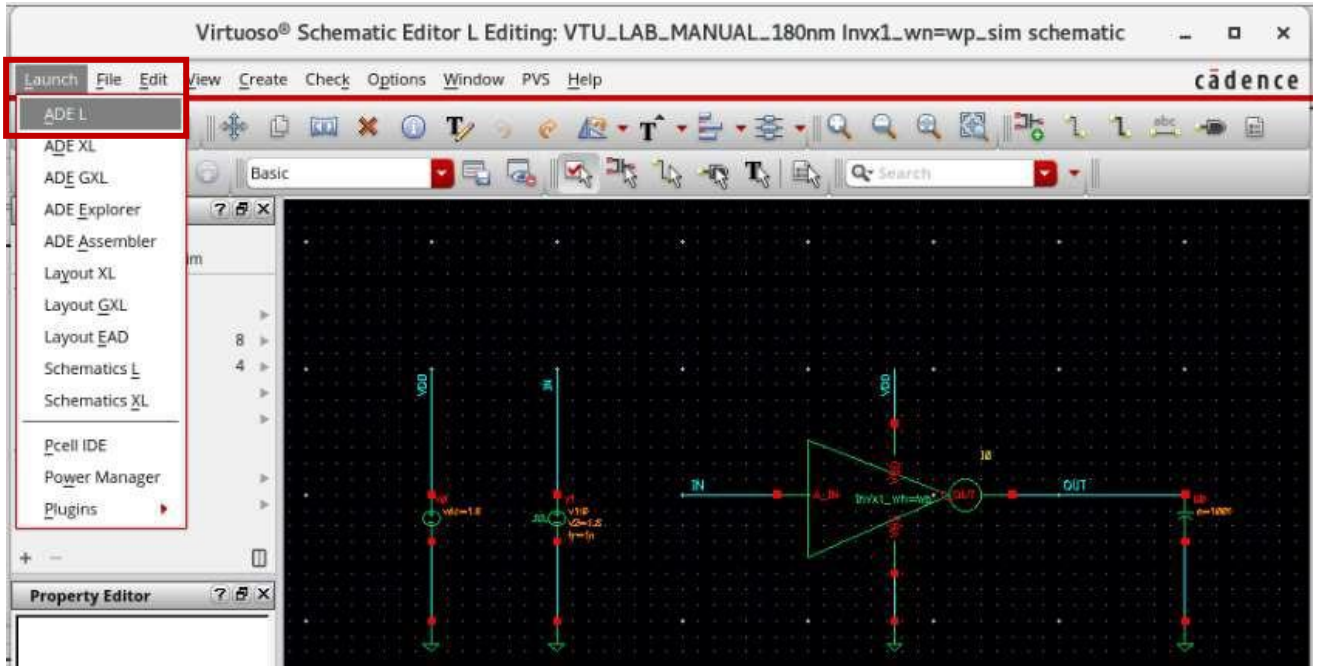


Figure – 1.47: Launch → ADE L

The "**ADE L**" window pops up as shown in Figure – 1.48.

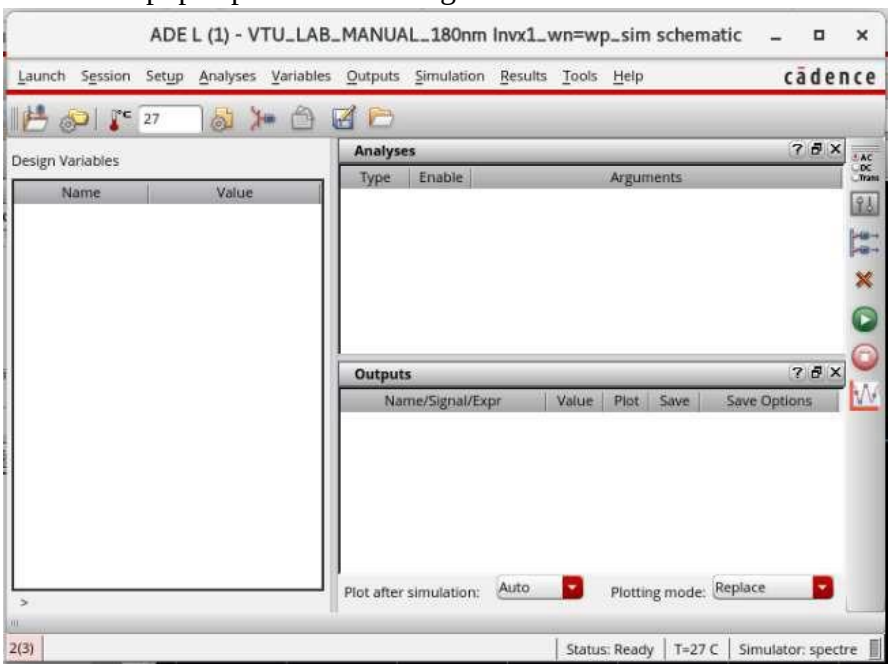


Figure – 1.48: ADE L Window

Before running the simulation, check for the Simulator and Model Libraries.

SELECTING THE SIMULATOR:

To select the Simulator, click on "**Setup** □ **Simulator/Directory/Host**" as shown in Figure – 1.49.

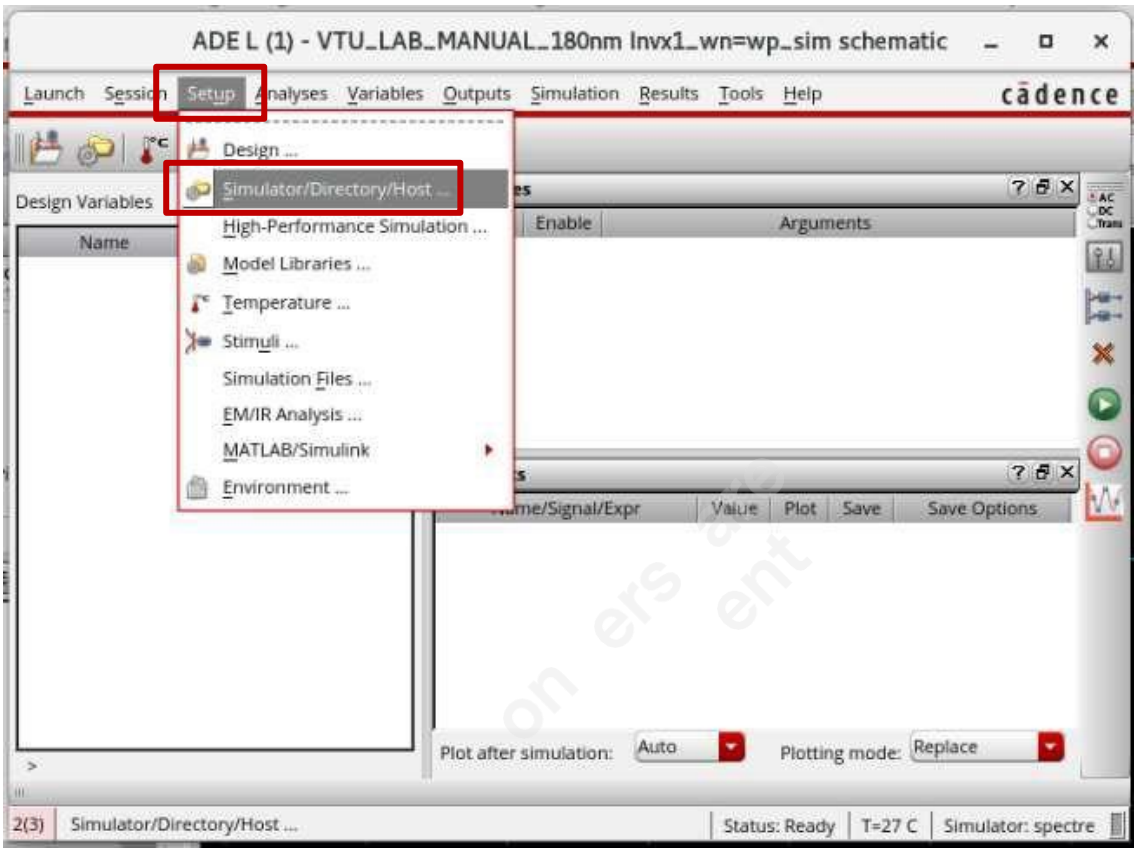


Figure – 1.49: Setup □ Simulator/Directory/Host..

The "**Choosing Simulator/Directory/Host**" window pops up. Select "**Simulator** □ **Spectre**" and click on "**OK**" as shown in Figure – 1.50.

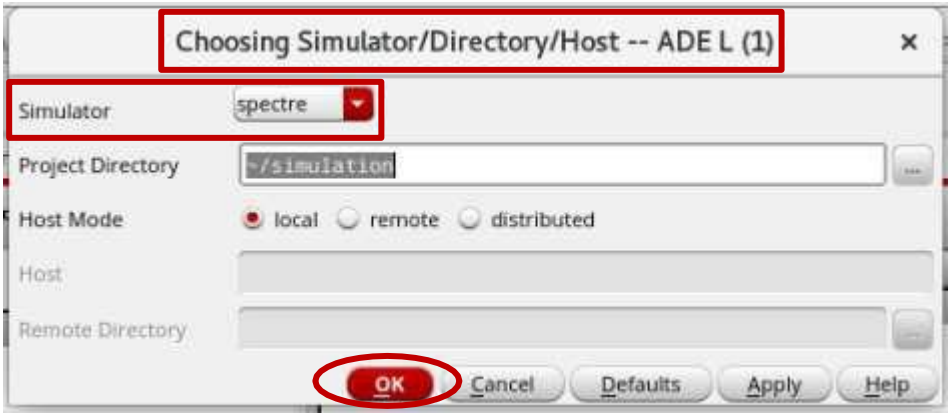


Figure – 1.50: Simulator □ Spectre

SELECTING THE MODEL LIBRARIES AND PROCESS CORNERS:

The Model Libraries and Process Corners are important to run the simulation.

To select the "**.scs**" file with respect to the technology node, select "**Setup □ ModelLibraries**" as shown in Figure – 1.51.

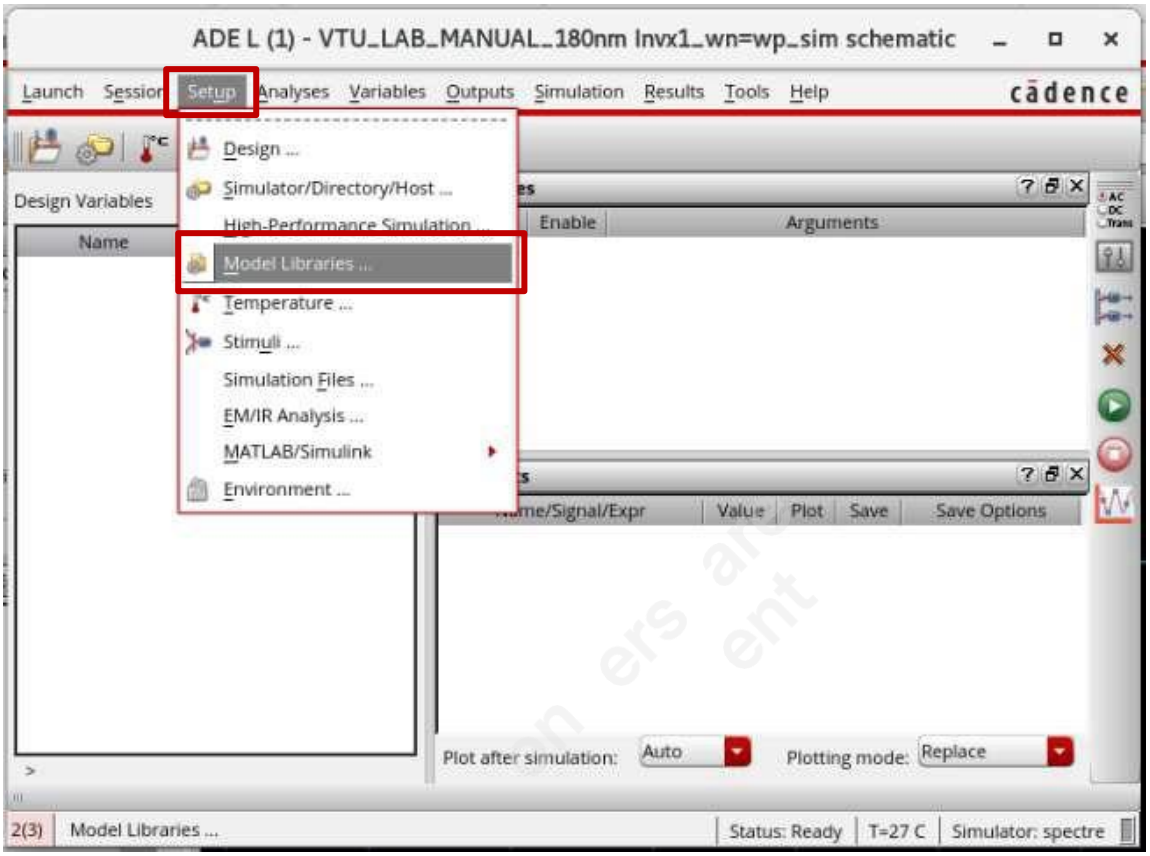


Figure – 1.51: Setup □ Model Libraries

The "**spectre0: Model Library Setup**" window pops up as shown in Figure – 1.52.

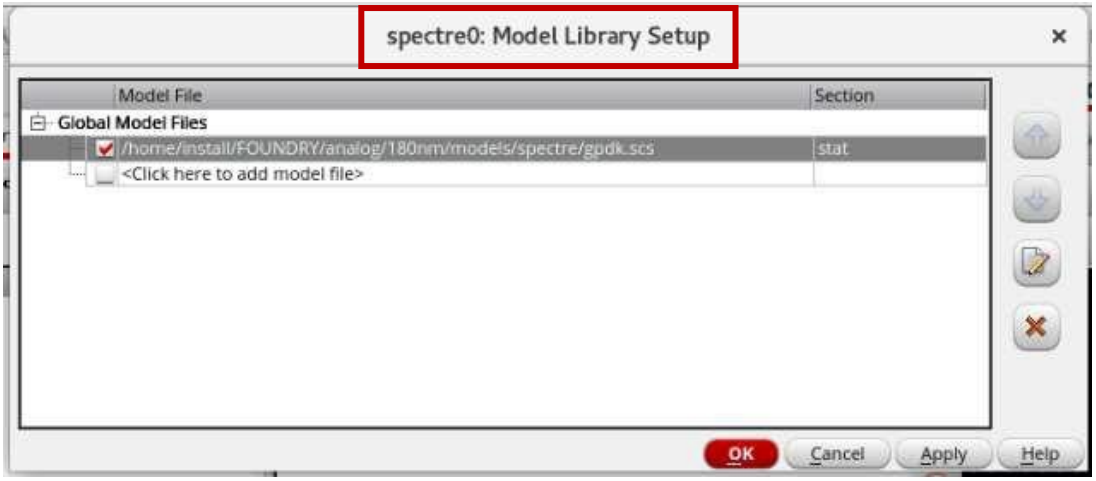


Figure – 1.52: "spectre0: Model Library Setup" Window

Select the respective "**.scs**" file and make a double click under "**Section**" to select the processing corner of interest using a Left Mouse Click on the drop down and click on "**OK**" as shown in Figure – 1.53.

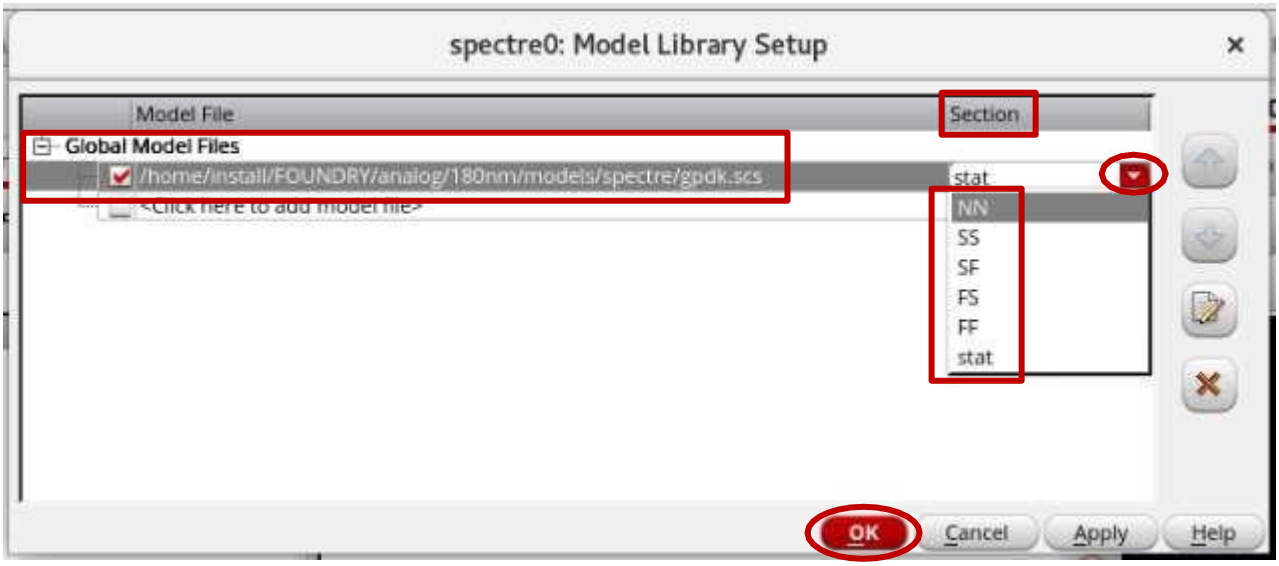


Figure – 1.53: ".scs" file and Processing Corner Selection

SELECTING THE ANALYSIS:

To select the analysis required to be performed on the Test Circuit, select "**Analyses** □ **Choose**" from the top menu in the ADE L window as shown in Figure – 1.54.

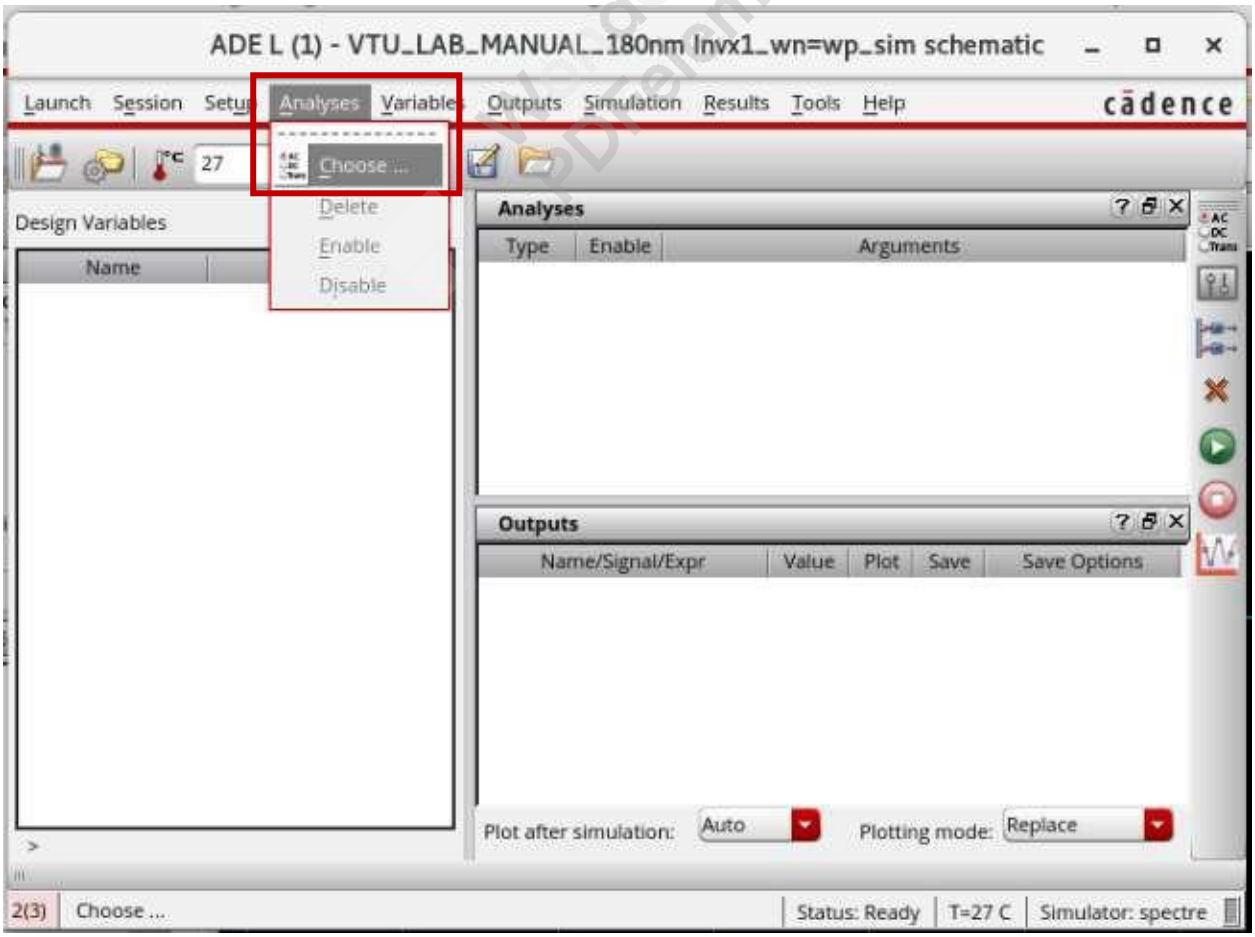


Figure – 1.54: Analyses □ Choose

The "**Choose Analyses – ADE L**" window pops up as shown in Figure – 1.55.

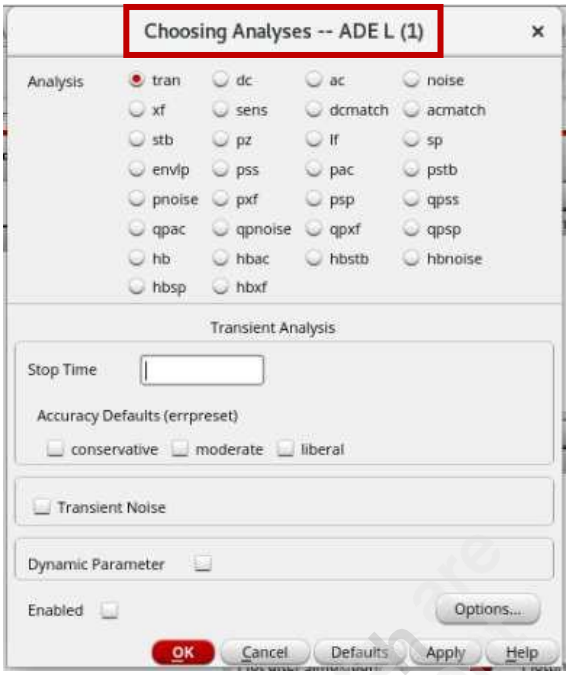


Figure – 1.55: "Choosing Analysis – ADE L" Window

TRANSIENT ANALYSIS:

To set up a "**Transient Analysis**", select "**tran**", mention the "**Stop Time**" (for example: 100n), select "**Accuracy Defaults**" (for example: moderate), click on "**Apply**" and click on "**OK**" as shown in Figure – 1.56.

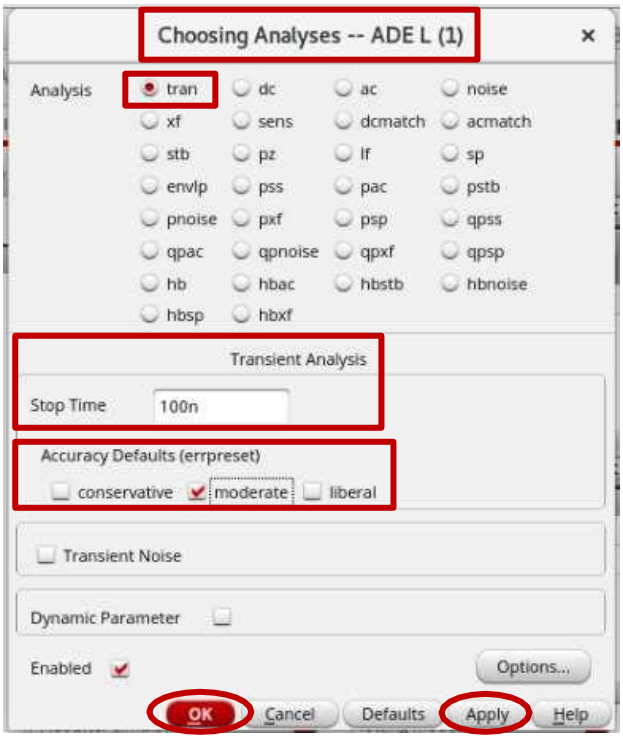
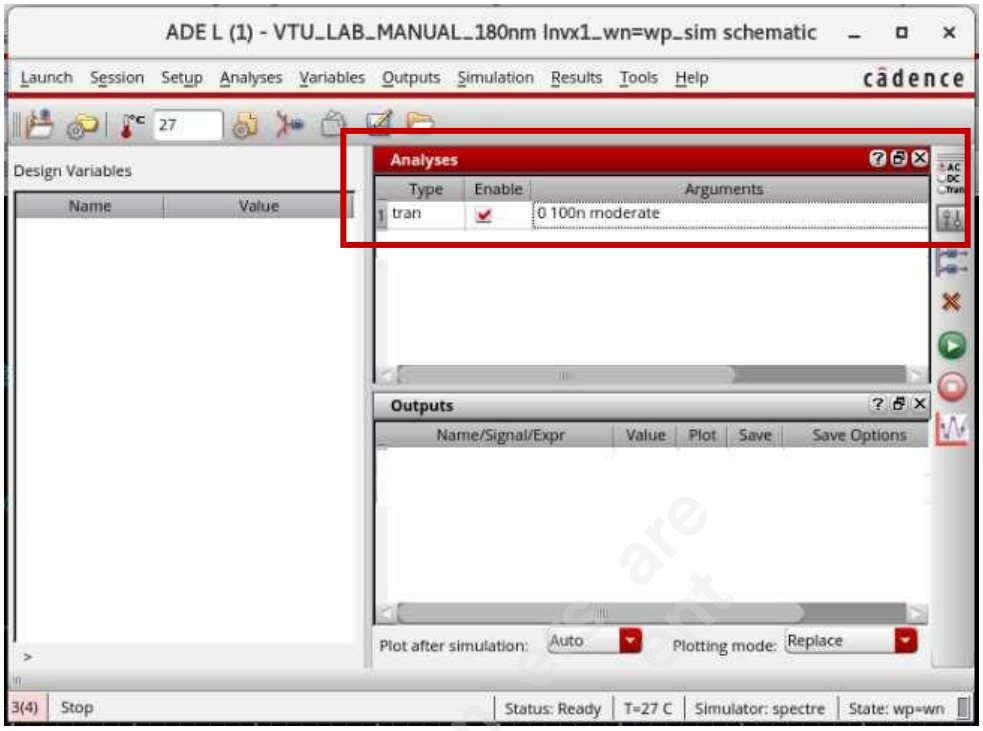


Figure – 1.56: Setup for Transient Analysis

The selected analysis and the arguments can be seen under the "**Analyses**" tab in the ADE L window as shown in Figure – 1.57.



DC ANALYSIS:

To set up a "**DC Analysis**", select "**dc**" and enable "**Save DC Operating Point**" as shown in

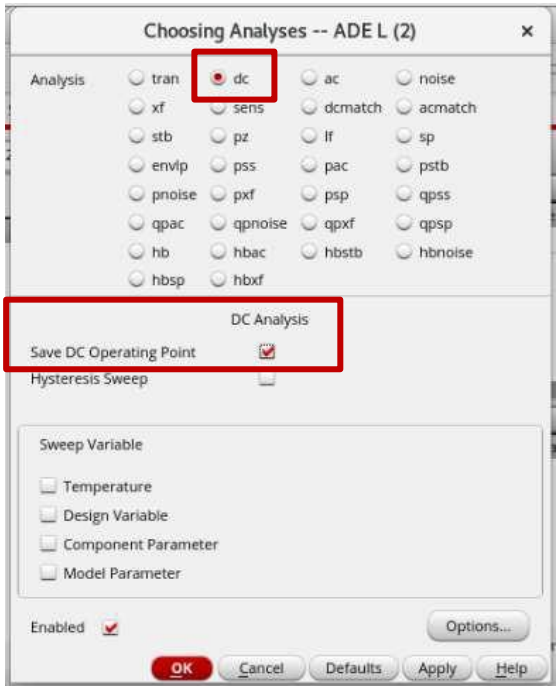


Figure – 1.58: Select "dc"

Enable "**Component Parameter**", click on "**Select Component**" as shown in Figure – 1.59.

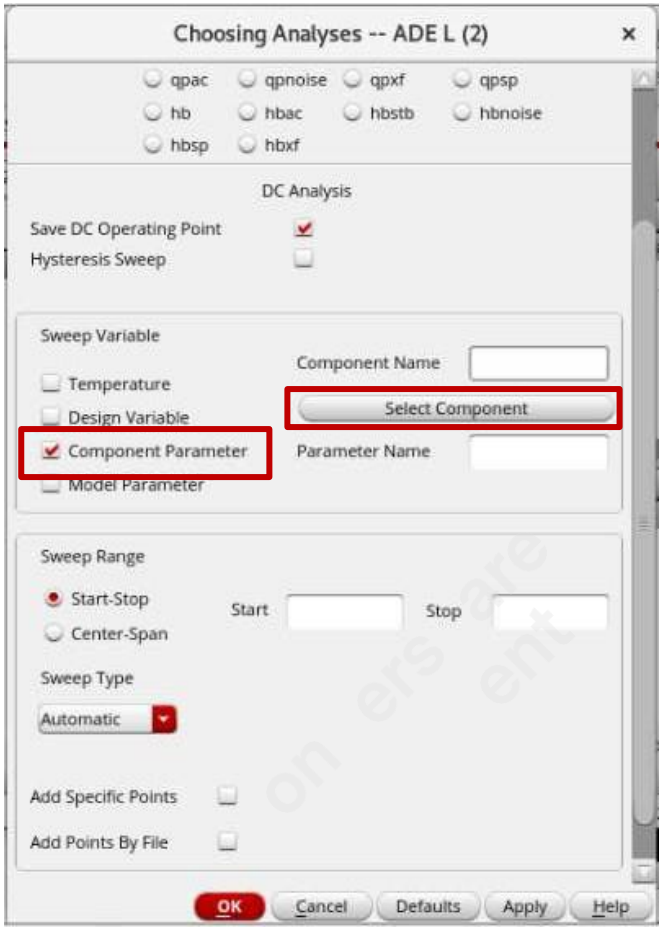


Figure – 1.59: Enable "Component Parameter"

Select the "**vpulse**" source from the Test Schematic as shown in Figure – 1.60.

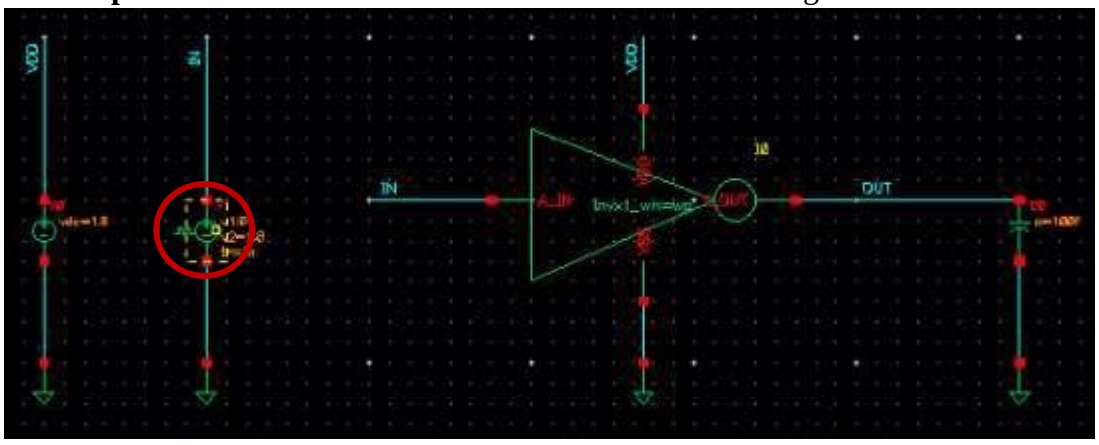


Figure – 1.60: Selecting "vpulse" from the Test Schematic

Select "**DC Voltage**" from the list of parameters as shown in the "**Select Component Parameter**" window and click on "**OK**" as shown in Figure – 1.61.

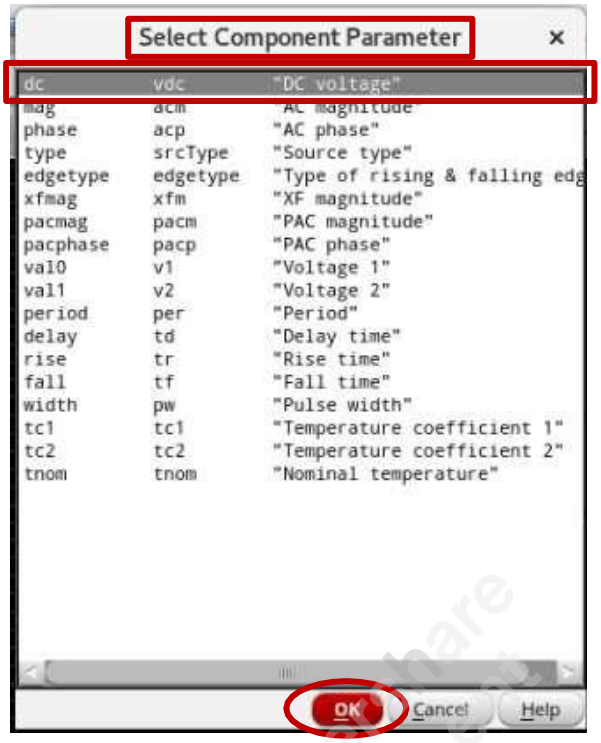


Figure – 1.61: "Select Component Parameter" window

From the "**Sweep Range**" option, select "**Start-Stop**" and mention the "**Start**" value as "**0**" and "**Stop**" value as "**1.8**", click on "**Apply**" and click on "**OK**" as shown in the Figure – 1.62.

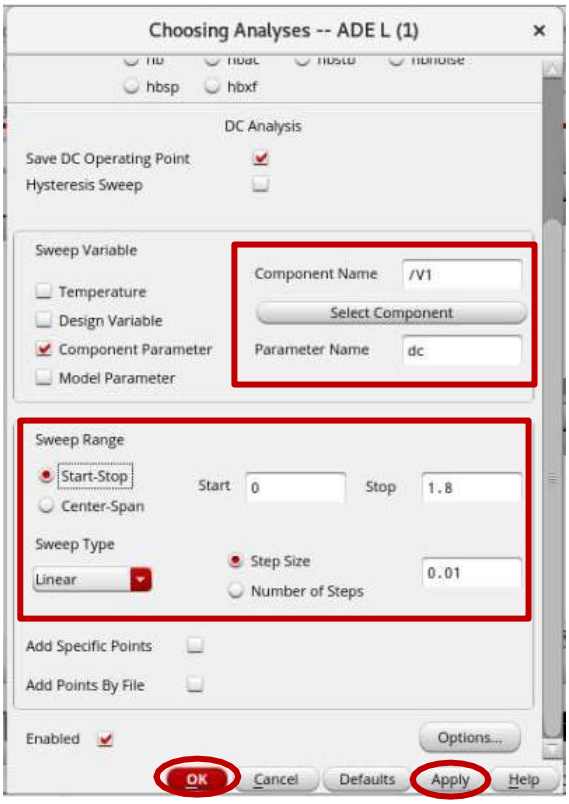


Figure – 1.62: Mention the Sweep Range

The ADE L window is now updated as shown in Figure – 1.63.

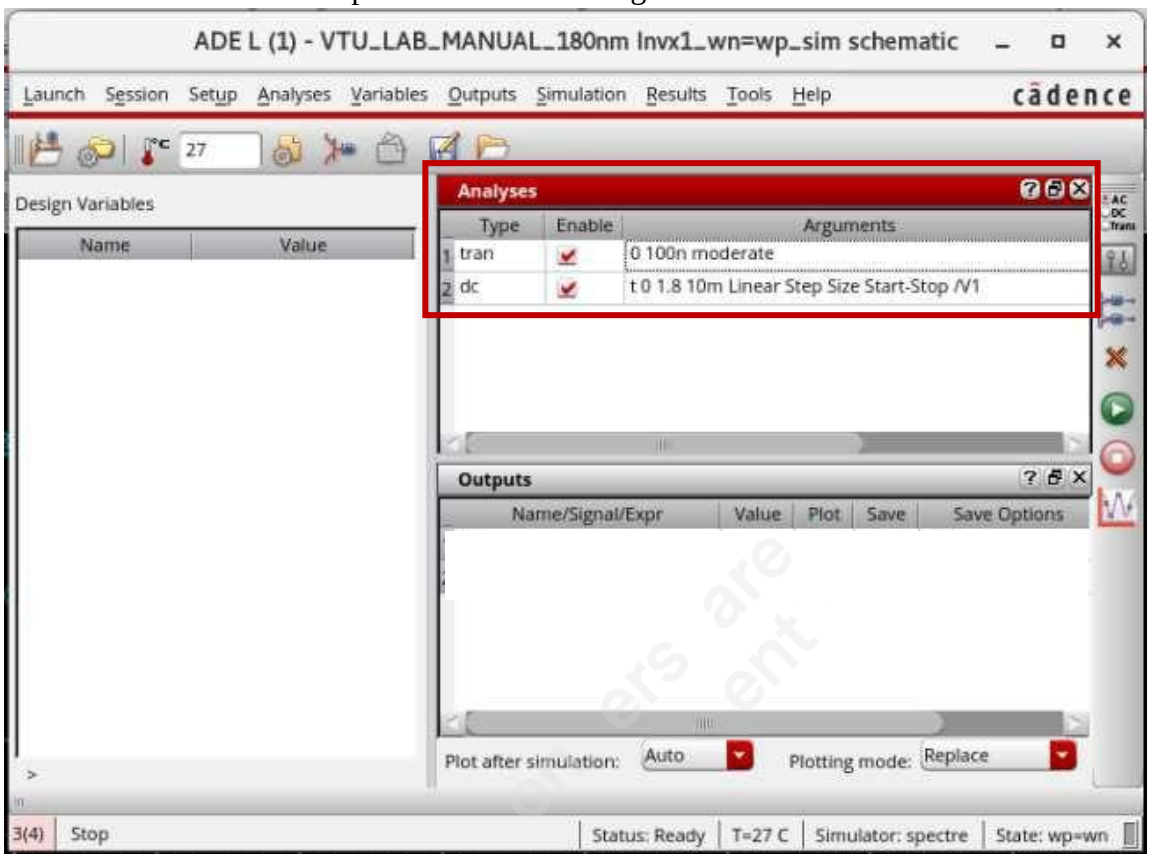


Figure – 1.63: Updated ADE L window

SELECTING THE SIGNALS TO BE PLOTTED:

To select the signals to be plotted, select "**Outputs □ Setup**" from the ADE L window as shown in Figure – 1.64.

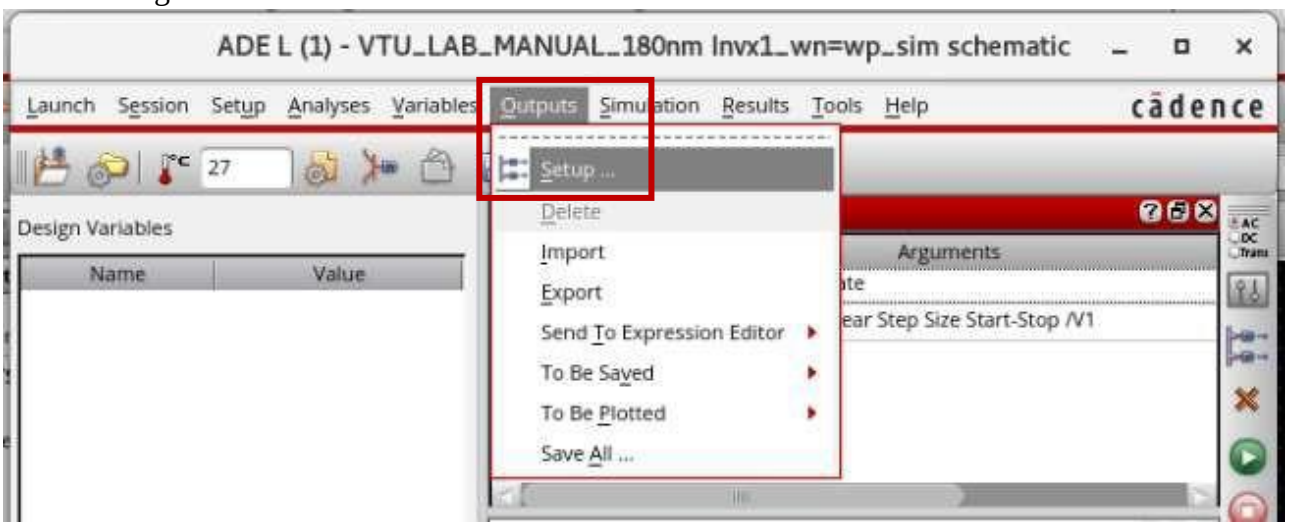


Figure – 1.64: Outputs □ Setup

The "**Setting Outputs – ADE L**" window pops up as shown in Figure – 1.65.

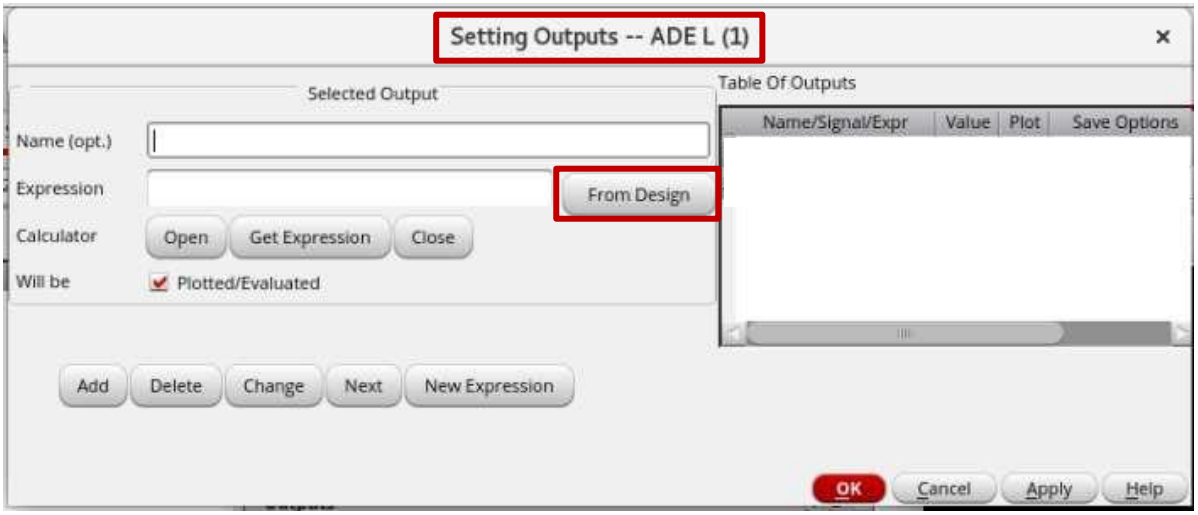


Figure – 1.65: Setting Outputs – ADE L window

Click on "**From Design**" as shown in the Figure – 1.65. This brings back the Test Schematic as shown in Figure – 1.66.

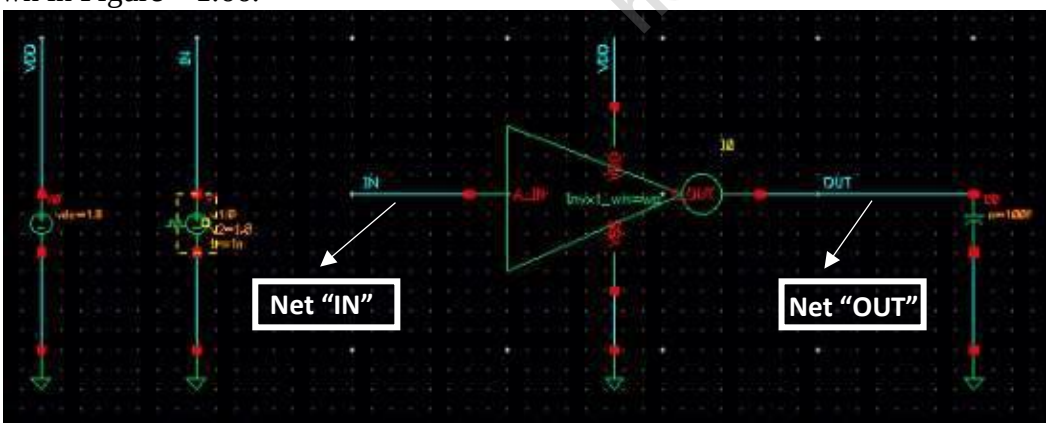


Figure – 1.66: Select the Net "IN" and Net "OUT"

Select the Input Net "**IN**" and the Output Net "**OUT**" as shown in Figure – 1.66. The selected Nets will be listed under "**Table of Outputs**" in the "**Setting Outputs – ADE L**" window as shown in Figure – 1.67. Click on "**OK**".

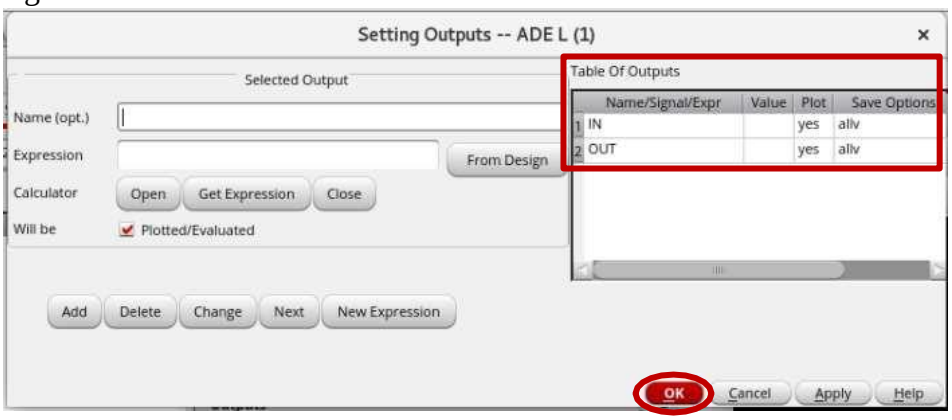


Figure – 1.67: Updated "Setting Outputs" window

The "**Outputs**" column in the "**ADE L**" window will be updated as shown in Figure – 1.68.

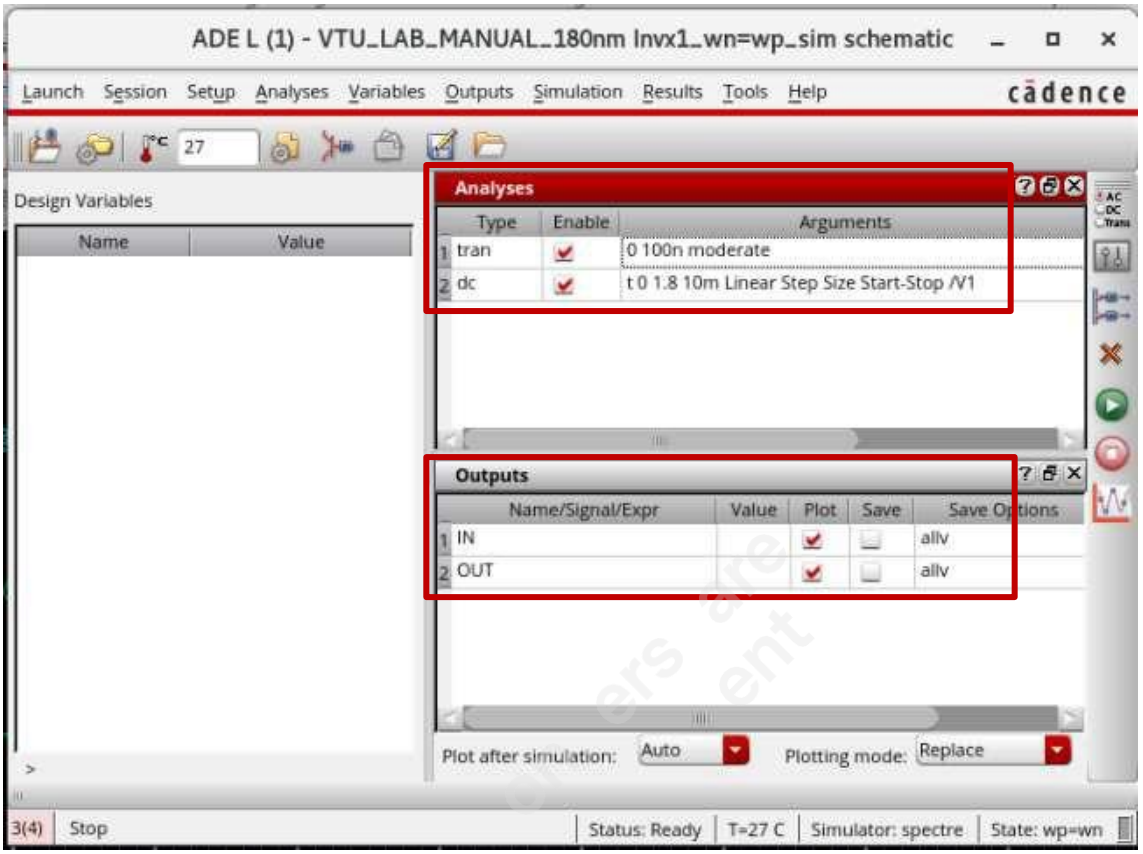


Figure – 1.68: "Outputs" in ADE L window

RUNNING THE SIMULATION:

To run the simulation, click on "**Simulation □ Netlist and Run**" from ADE L window as shown in Figure – 1.69.

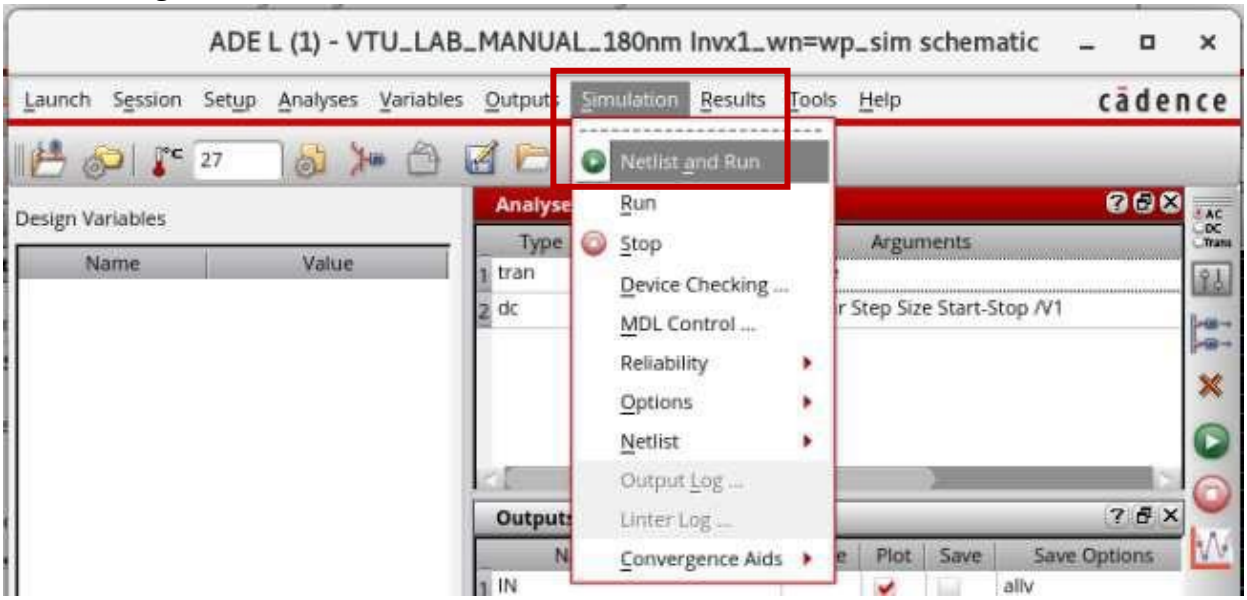


Figure – 1.69: Simulation □ Netlist and Run

The simulated waveforms can be seen on the "**Virtuoso Visualization and Analysis XL**" window as shown in Figure – 1.70.

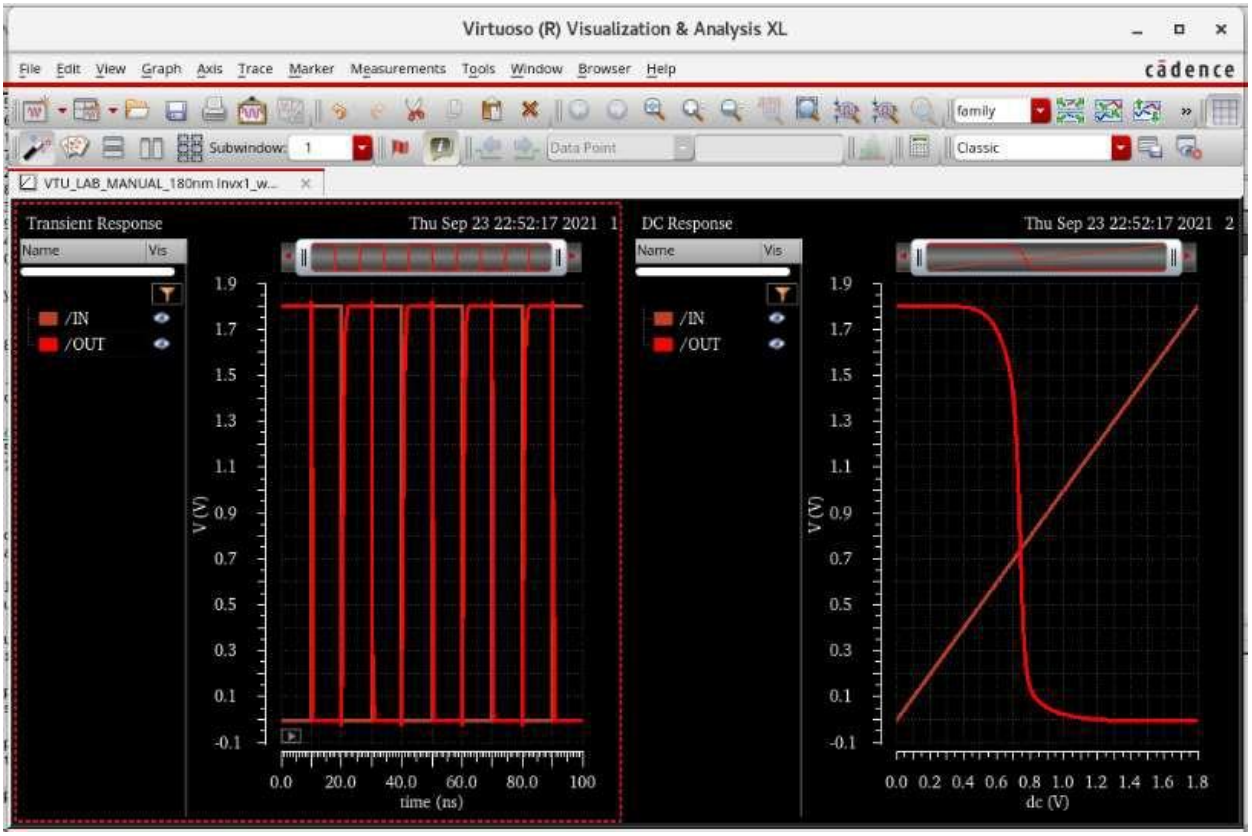


Figure – 1.70: Simulated waveforms

The Input and Output Signals can be split up by selecting "**Graph □ Split All Strips**" as in Figure – 1.71.

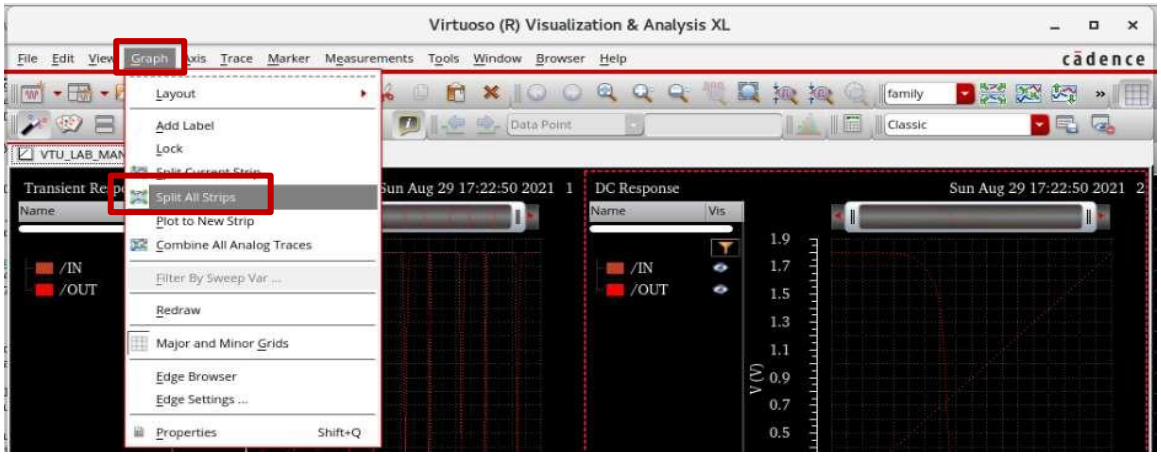


Figure – 1.71: Graph □ Split All Strips

SAVING THE ADE L STATE:

To save the current ADE L state, click on "**Session □ Save State**" as shown in Figure – 1.72.

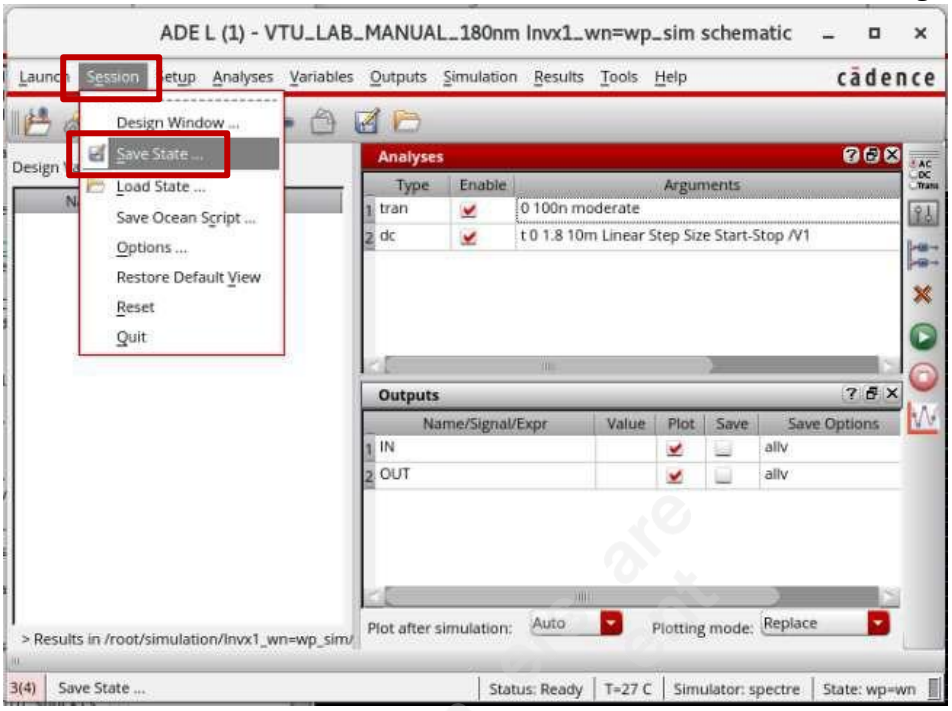


Figure – 1.72: Session □ Save State

The "**Saving State – ADE L**" window pops up. Select the "**Save State Option □ Cellview**" and click on "**OK**" as shown in Figure – 1.73.

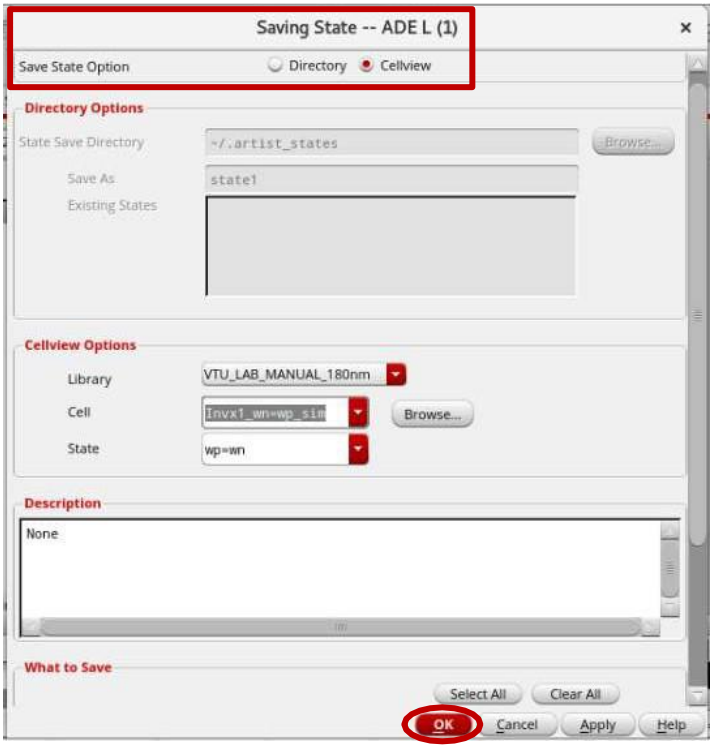
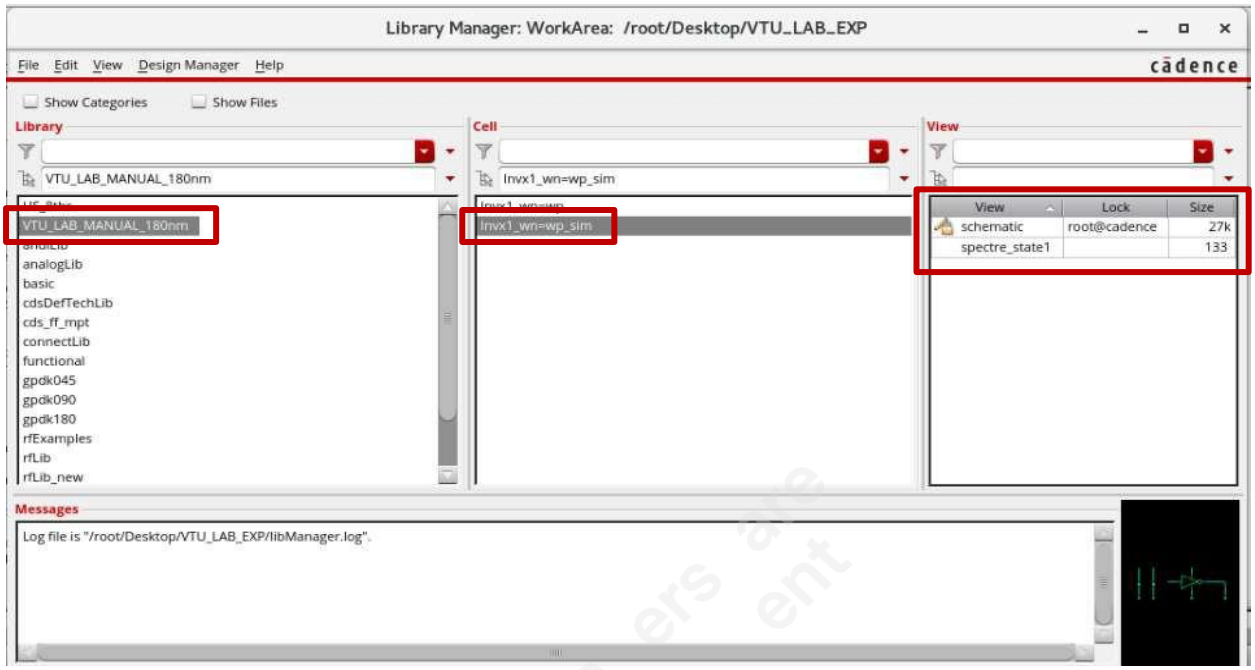


Figure – 1.73: Saving State

The Test Schematic and the State can be seen in the Library Manager as shown in Figure – 1.74.



OPEN THE SAVED ADE L STATE:

To open the saved state, click on "**Session □ Load State**" as shown in Figure – 1.75.

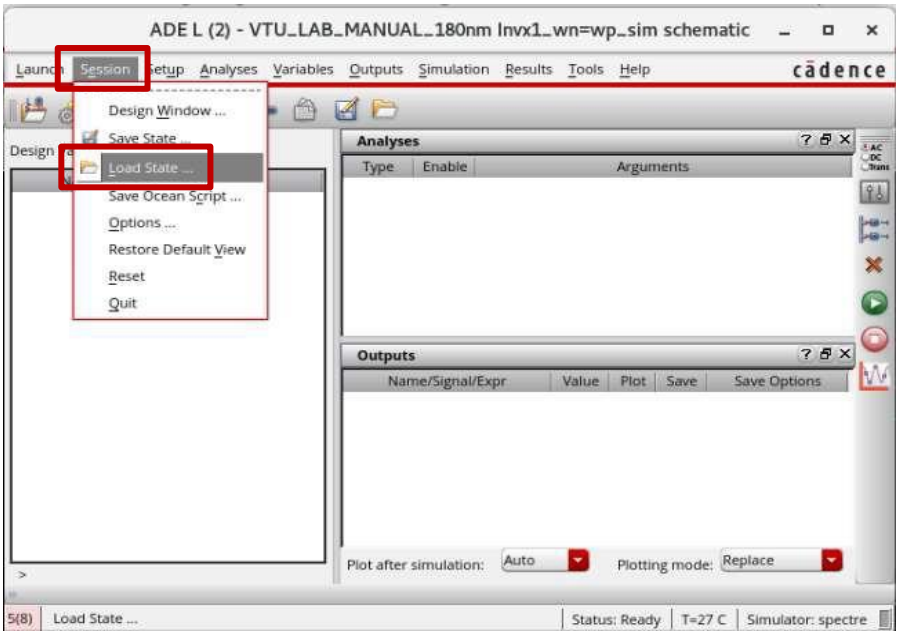


Figure – 1.75: Session □ Load State

The "**Loading State – ADE L**" window pops up. Select the "**Load State Option □ Cellview**" and click on "**OK**" as shown in Figure – 1.76.

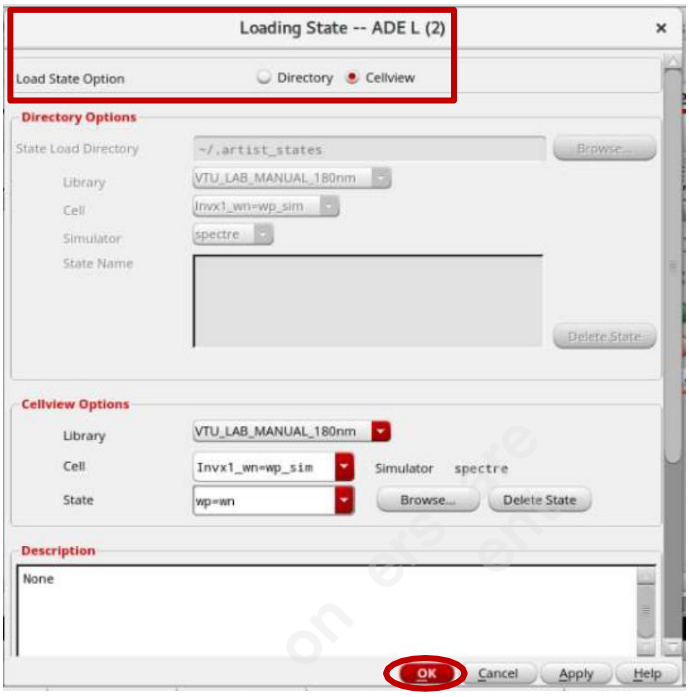


Figure – 1.76: "Loading State – ADE L" window

The Saved ADE L state is loaded as shown in Figure – 1.77.

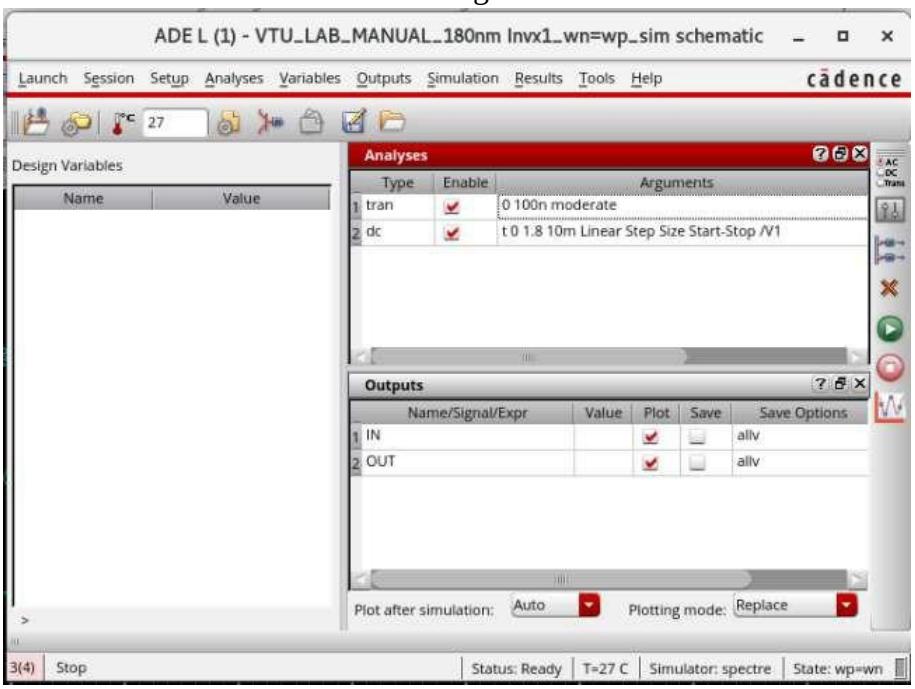


Figure – 1.77: Reloaded ADE L State

(2) CALCULATION OF t_{pHL}, t_{pLH} AND t_{PD}:

To calculate the Propagation Delay (t_{PD}), the formula used is

$$t_{PD} = \frac{(tp_{LH} + tp_{HL})}{2}$$

where, tp_{LH} □ Low – High Propagation Delay and tp_{HL} □ High – Low Propagation Delay. To calculate tp_{LH} and tp_{HL} use the Calculator option from the "**Virtuoso (R) Visualizationand Analysis**" window. So, select "**Tools** □ **Calculator**" or click on the icon as shown inFigure – 1.78.

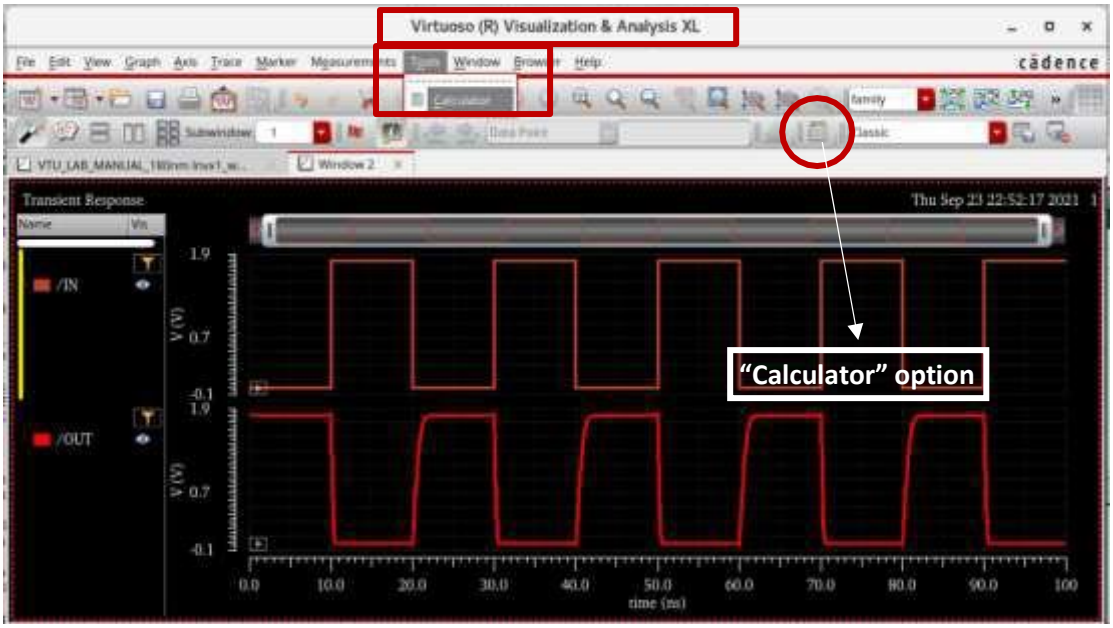


Figure – 1.78: Tools □ Calculator

The "**Virtuoso (R) Visualization and Analysis XL calculator**" window pops up as shown in Figure – 1.79.

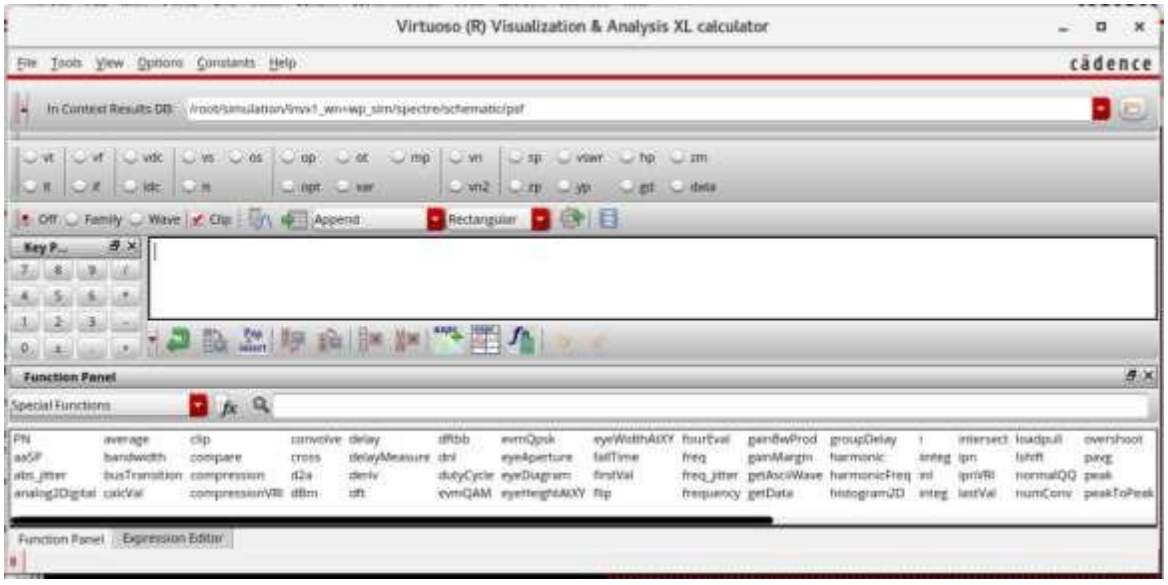
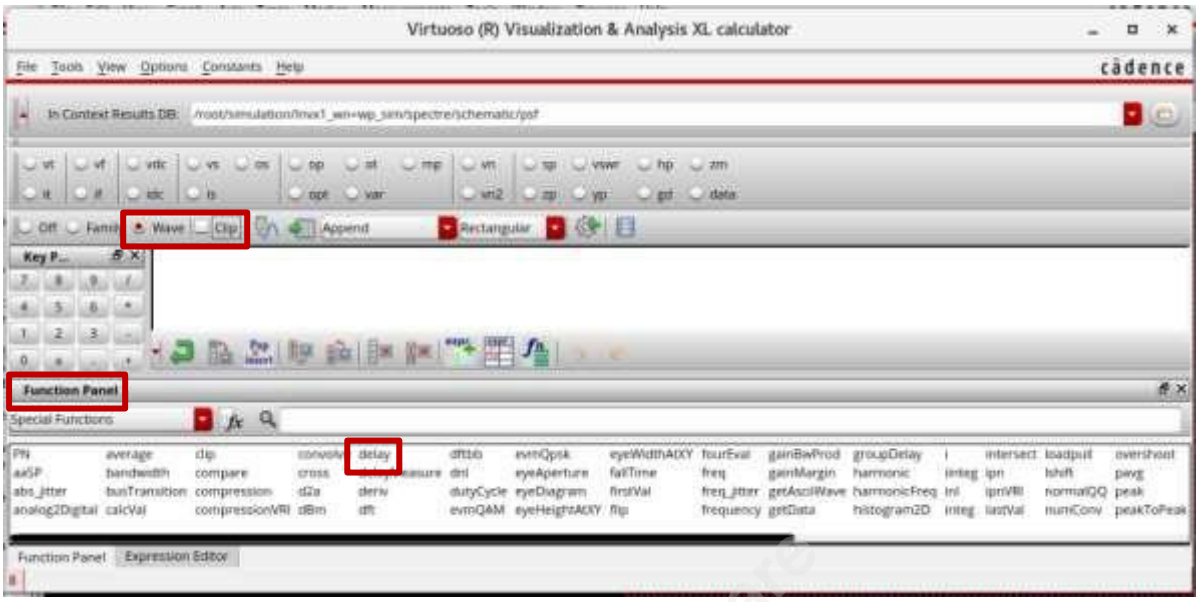


Figure – 1.79: Virtuoso (R) Visualization and Analysis XL calculator window

"**Enable □ Wave**" and "**Disable □ Clip**" as shown in Figure – 1.80.



gets updated as shown in Figure –

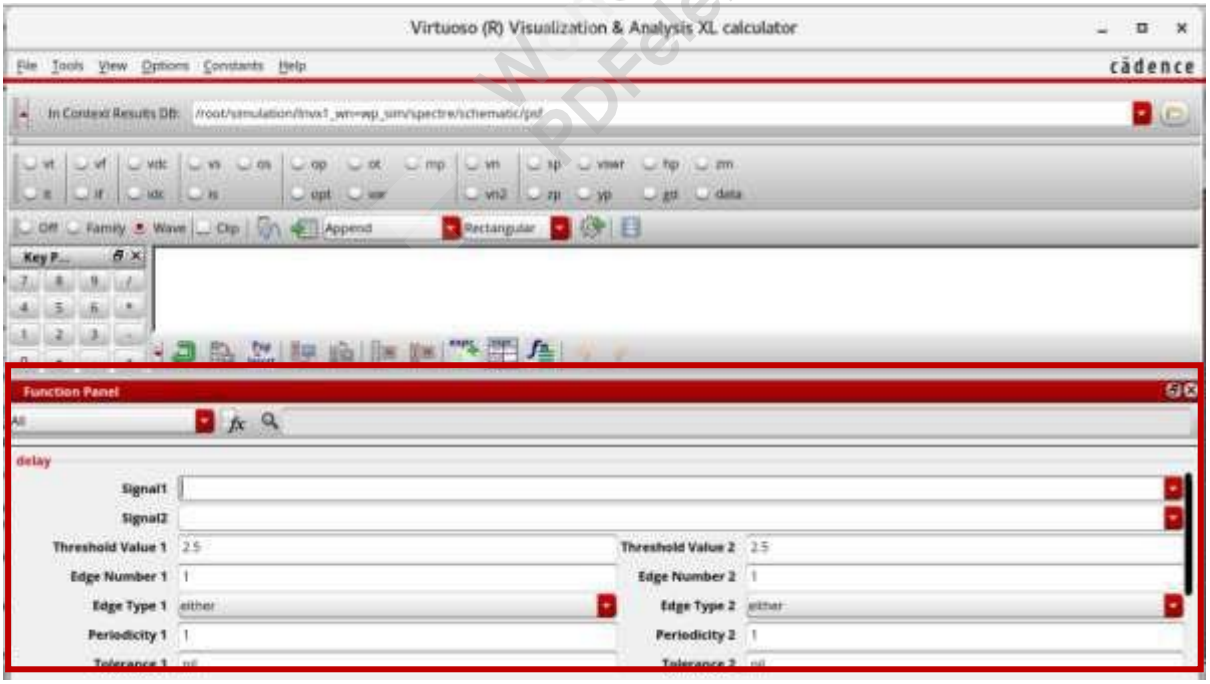
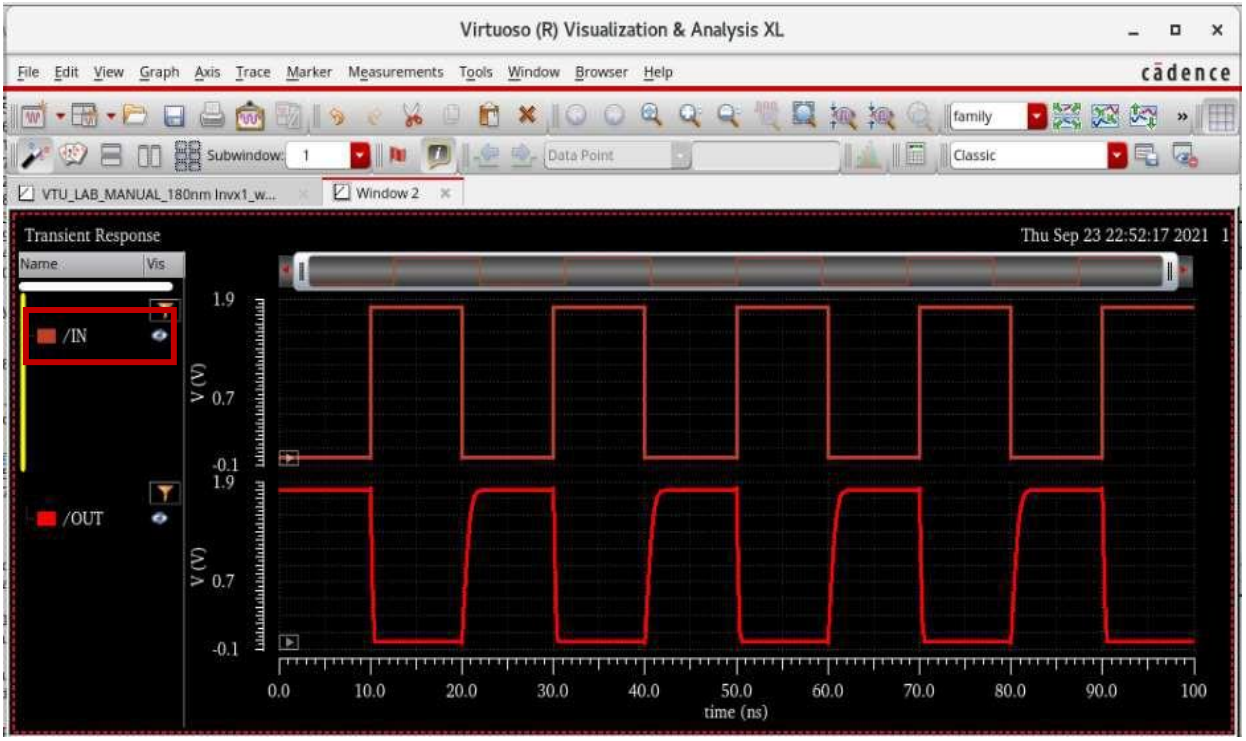


Figure – 1.81: Updated Function Panel

Place the cursor in "**Signal 1**", select the signal "**IN**" from the waveform window as shown in Figure – 1.82.



Similarly, place the cursor in "**Signal 2**" and select the "**OUT**" signal. The Function Panel gets updated as shown in Figure – 1.83.

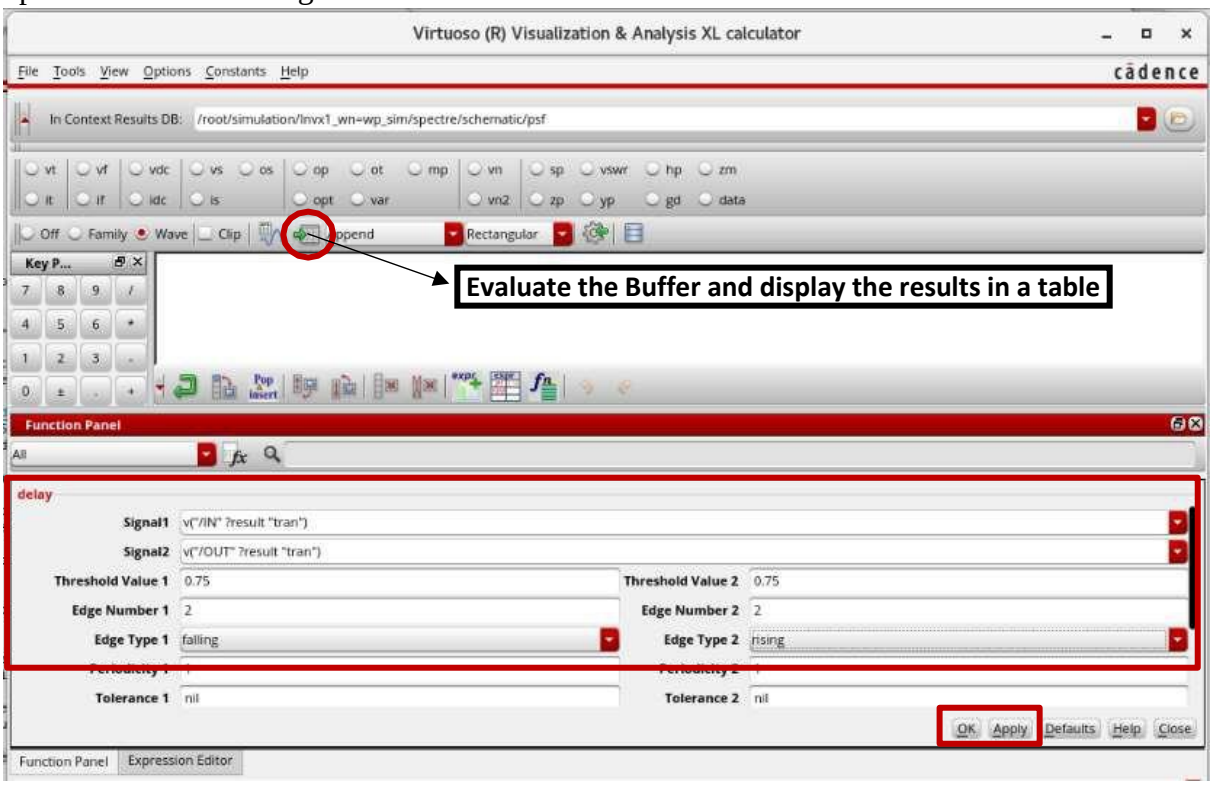


Figure – 1.83: "Signal 2" and other parameter selection

The value of "**Switching Potential**" should be mentioned under "**Threshold Value 1**" and "**Threshold Value 2**".

Note:

What is Switching Potential?

Switching Potential is defined as the value of Input Voltage for which the Output Voltage is equal to the Input Voltage.

How to obtain the value of Switching Potential?

To obtain the Switching Potential, use the "**intersect**" option from the **Function Panel**, select the "**Signal 1**" and "**Signal 2**" from the **DC Analysis** waveform window, click on "**Apply**", click on "**OK**" and click on "**Evaluate the buffer and display the resultsin a table**" icon as shown in Figure – 1.83 to obtain the value.

Select "**Edge Number 1**" and "**Edge Number 2**" as "**2**" (for example). Select "**Edge Type 1 □ falling**" and "**Edge Type 2 □ rising**" to obtain the value of "tp_{LH}" and "**Edge Type 1 □ rising**" and "**Edge Type 2 □ falling**" to obtain the value of "tp_{HL}".
After the above mentioned selections, click on "**Apply**" and click on "**OK**" to see the "**Buffer**" window in the calculator getting updated as shown in Figure – 1.84.

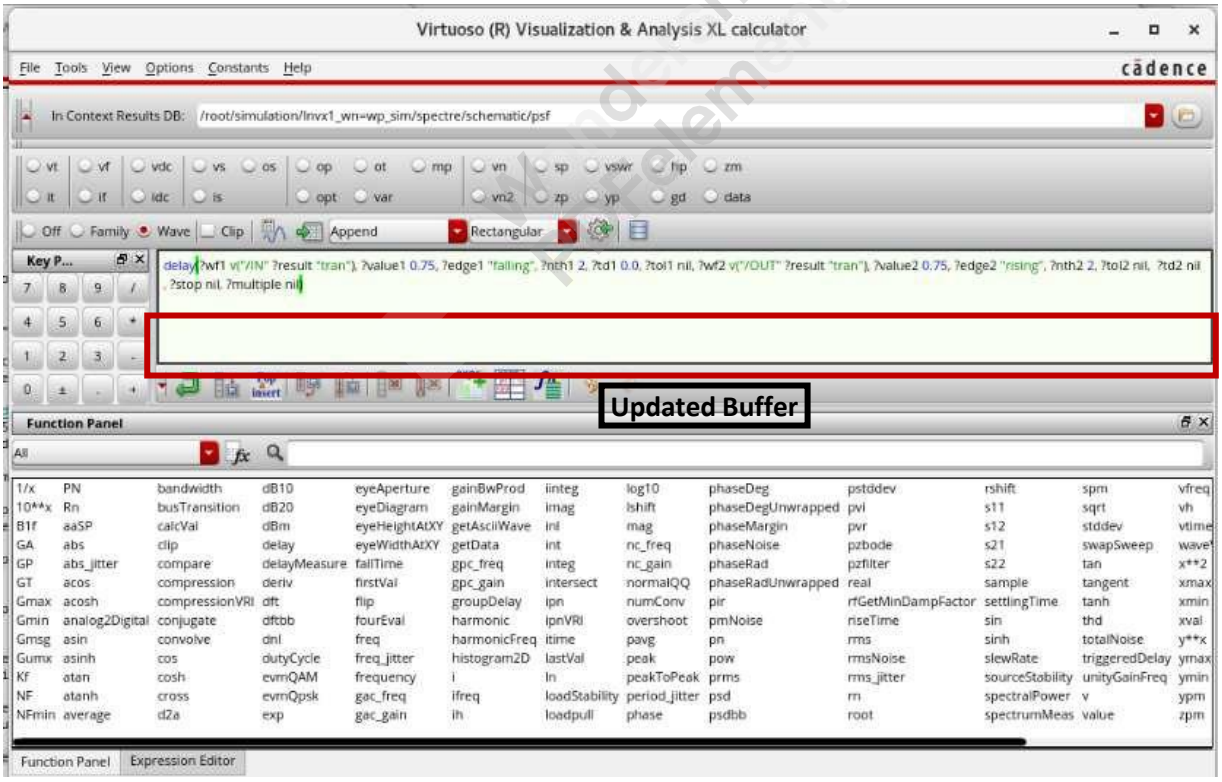


Figure – 1.84: Updated Buffer

Click on the icon "**Evaluate the buffer and display the results in a table**" as shown in Figure – 1.83 to obtain the value of tp_{HL} / tp_{LH}. Use the formula mentioned above to obtain the t_{PD}.

Obtain the values of tp_{HL}, tp_{LH} and t_{PD} for all the three geometrical settings of Width.

(3) TABULATED VALUES OF DELAY:

The results of tp_{HL}, tp_{LH} and t_{PD} for all the required geometrical settings are tabulated below in Table – 5.

Table – 5: Values of tp_{HL}, tp_{LH} and t_{PD} for different geometries

Width Settings	MOSFET	Width	tpLH	tpHL	tpd
Wp =Wn	PMOS	850n	3.233E-10	7.049E-10	5.141E-10
	NMOS	850n			
Wn =2Wp	PMOS	850n	3.337E-10	4.700E-10	4.019E-10
	NMOS	1.7u			
Wn = Wp/2	PMOS	425n	1.141E-09	3.154E-10	7.282E-10
	NMOS	850n			

(b) Layout of CMOS Inverter with $\frac{WP}{WN} = \frac{40}{20}$

Objective:

To draw the Layout of CMOS Inverter with $\frac{WP}{W_N} = \frac{40}{20}$ using optimum Layout Methods. Verify for DRC and LVS, extract the Parasitics and perform the Post-Layout Simulations, compare the results with Pre-Layout Simulations and record the observations.

SCHEMATIC CAPTURE:

Create a New Library, Create a Cellview and instantiate the required devices through "Create Instance" option. The parameter for PMOS and NMOS Transistors are listed in Table – 6 shown below.

Table – 6: Parameters for NMOS and PMOS Transistors

Library Name	Cell Name	Comments / Properties
gpdk180	Nmos	Width, W_N = 20 u Length, L = 180 n
gpdk180	Pmos	Width, W_P = 40 u Length, L = 180 n

Follow the techniques demonstrated in Lab – 01 to complete the Schematic. The completed CMOS Inverter circuit is shown in Figure – 1.85.

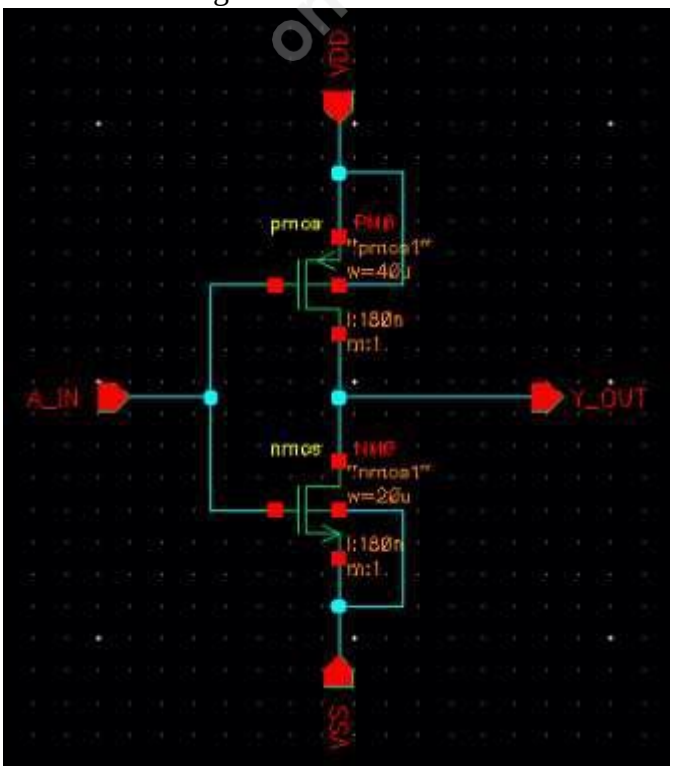


Figure – 1.85: Schematic for CMOS Inverter with $\frac{WP}{W_N} = \frac{40}{20}$

The symbol for the CMOS Inverter is shown in Figure – 1.86.

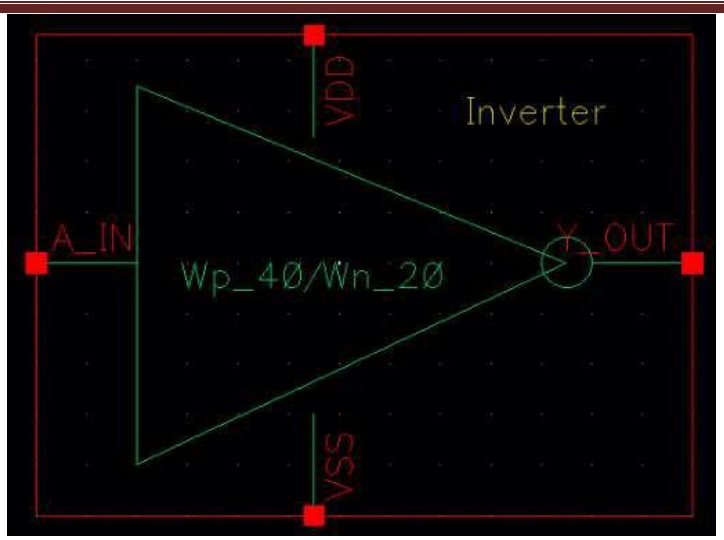


Figure – 1.86: Symbol for CMOS Inverter with $\frac{WP}{W_N} = \frac{40}{20}$

Create a New Cellview and capture the Test Schematic using the symbol shown in Figure – 1.86. The Test Schematic is shown in Figure – 1.87.

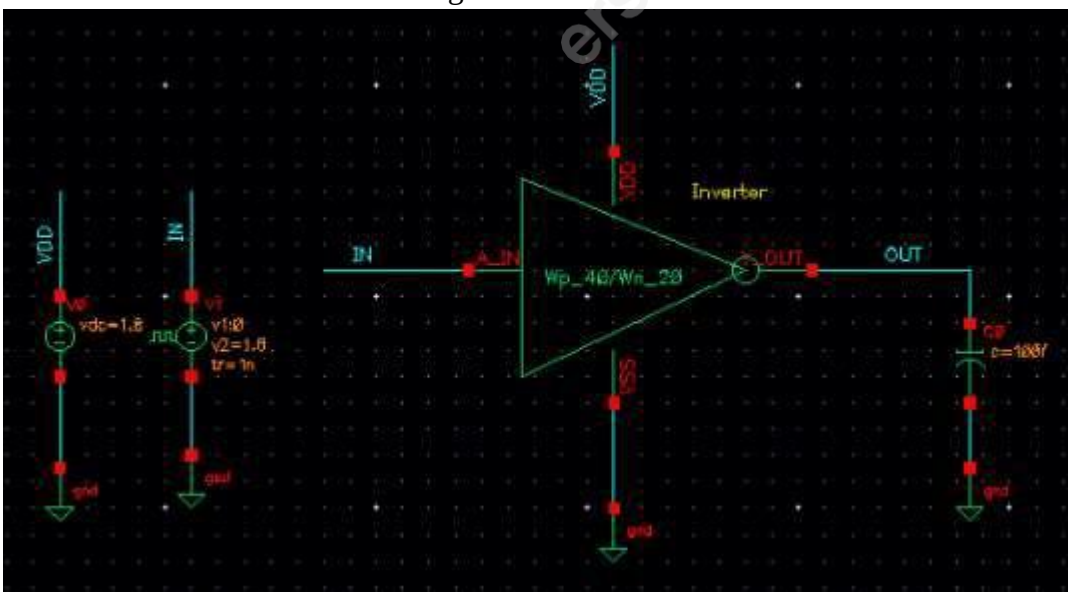


Figure – 1.87: Test Schematic for CMOS Inverter with $\frac{WP}{W_N} = \frac{40}{20}$

The parameters for "vdc", "vpulse" and the "capacitor" are the same as shown in Table – 4. Check and Save the design.

SIMULATION:

Launch the ADE L window from the Test Circuit, setup the Simulator, Model Libraries and the Process Corner as shown in Figure – 1.50, Figure – 1.52 and Figure – 1.53.

Setup the DC Analysis and Transient Analysis through the "Choose Analysis" option and the parameters are the same as shown in Figure – 1.56, Figure – 1.58 and Figure – 1.62.

Select the signals to be plotted and the updated ADE L window is shown in Figure – 1.88.

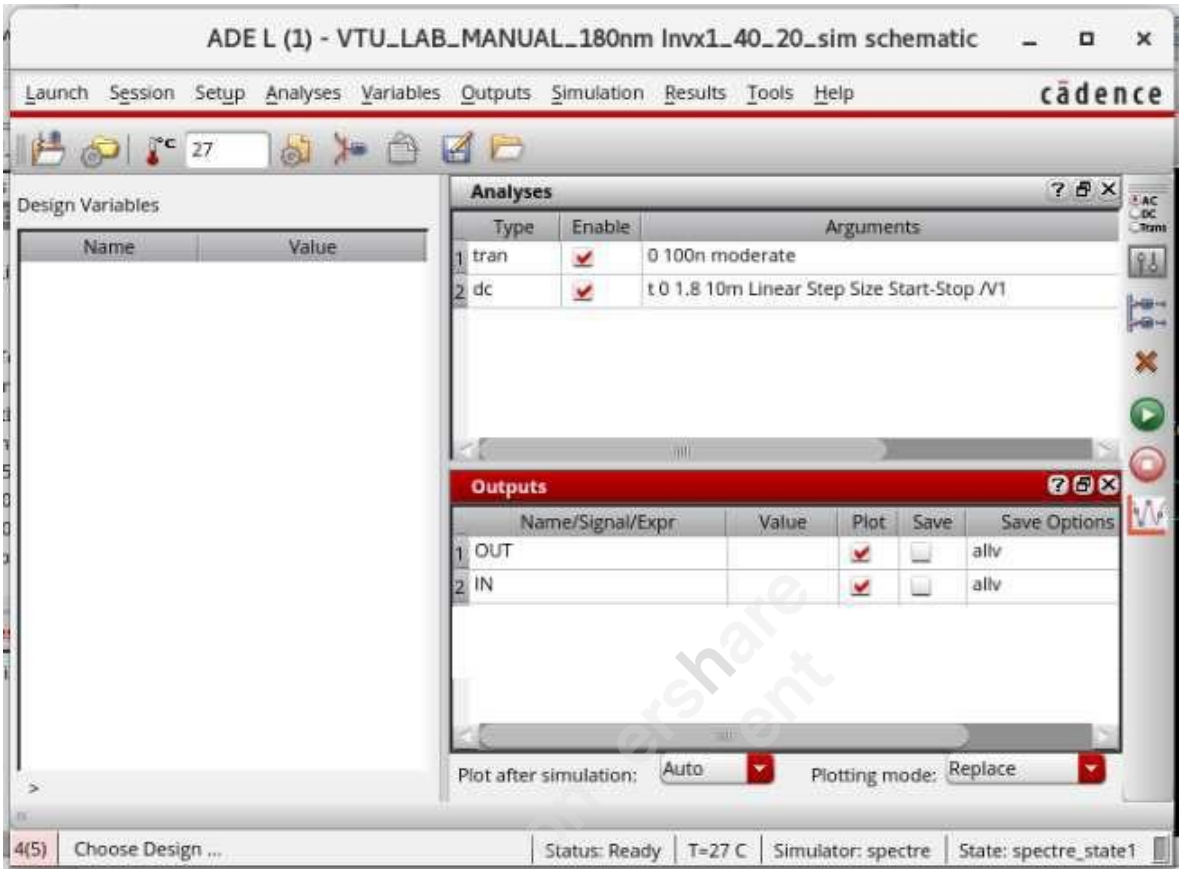


Figure – 1.88: Updated ADE L window

The signals plotted as a result of Transient Analysis is shown in Figure – 1.89.

Figure – 1.89: Transient Analysis for CMOS Inverter with $\frac{W}{W_N} = \frac{40}{20}$

Similarly, the signals plotted after DC Analysis are shown in Figure – 1.90.

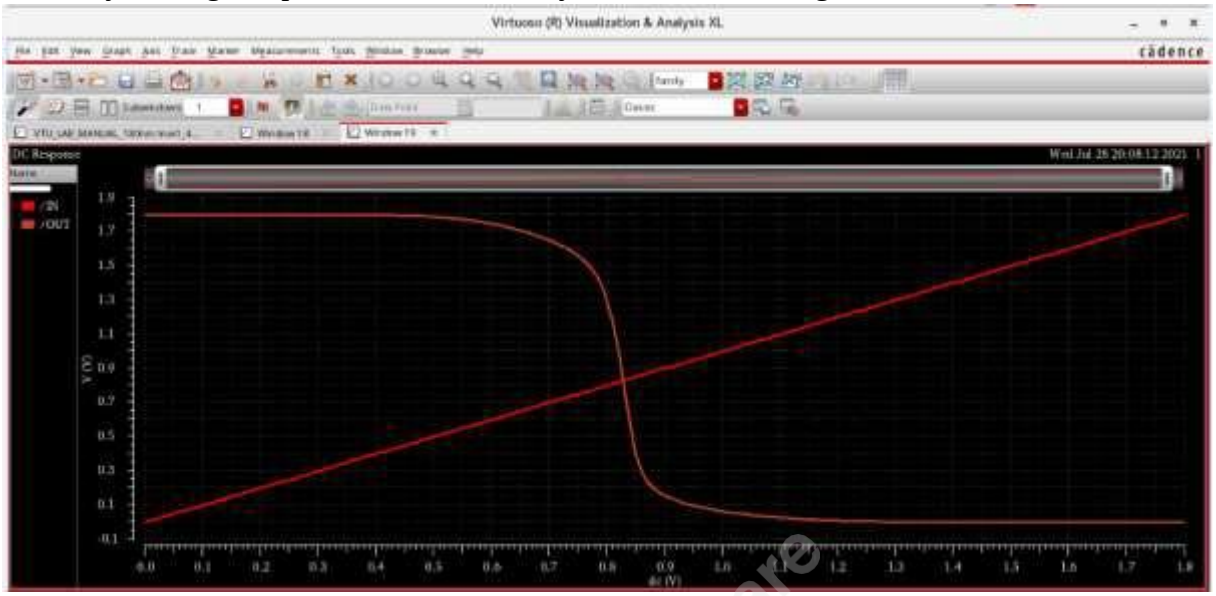


Figure – 1.90: DC Analysis for CMOS Inverter with $\frac{WP}{W_N} = \frac{40}{20}$

VALUES OF tp_{HL}, tp_{LH} AND t_{PD}:

Obtain the values of tp_{HL}, tp_{LH} and t_{PD} by referring to "**CALCULATION OF t_{PHL}, t_{PLH} AND t_{PD}**" in the previous section. The values are tabulated as shown in Table – 7.

Table – 7: Values of tp_{HL}, tp_{LH} and t_{PD} for CMOS Inverter with $\frac{WP}{W_N} = \frac{40}{20}$

MOSFET	Length	Width	tpLH	tpHL	tpd
PMOS	180n	40u	1.228E-10	3.550E-11	7.920E-11
NMOS	180n	20u			

LAYOUT FOR CMOS INVERTER WITH $\frac{W_P}{W_N} = \frac{40}{20}$

From the Virtuoso Schematic Editor as shown in Figure – 1.85, select "**Launch □ Layout XL**" as shown in Figure – 1.91.

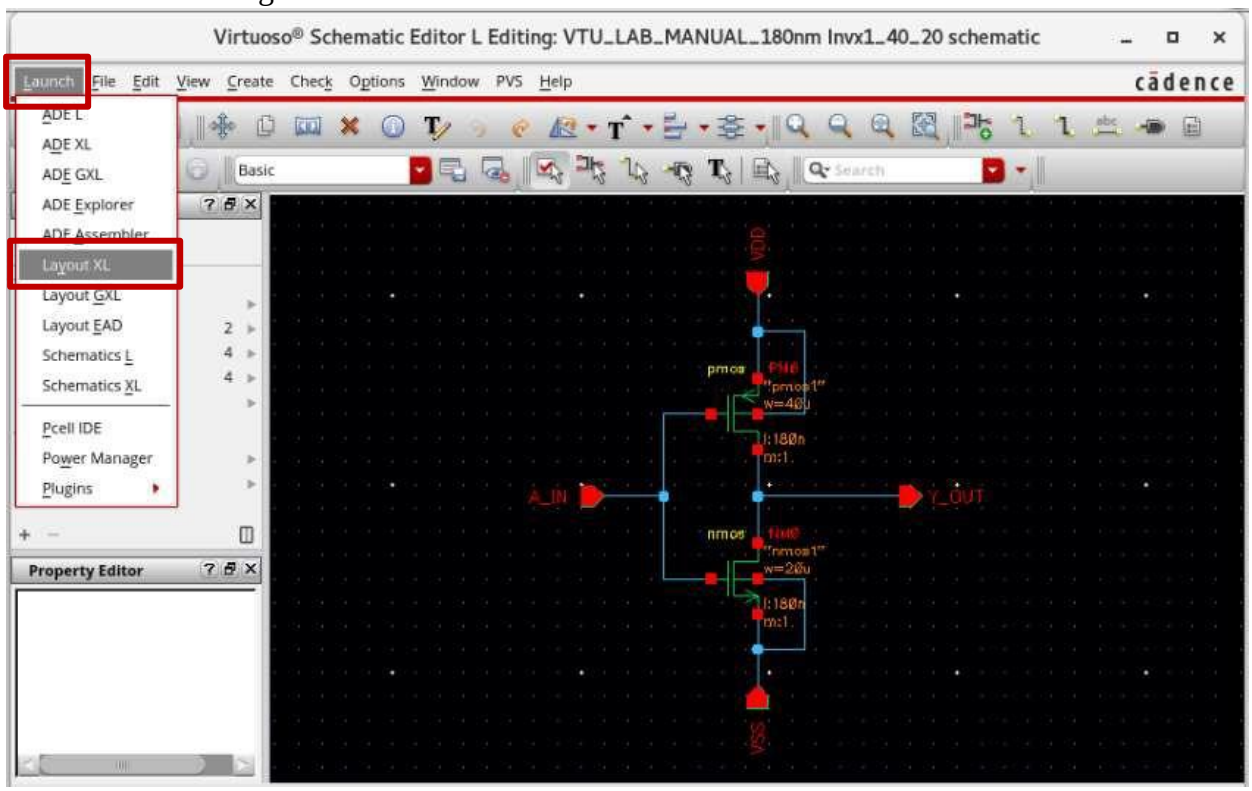
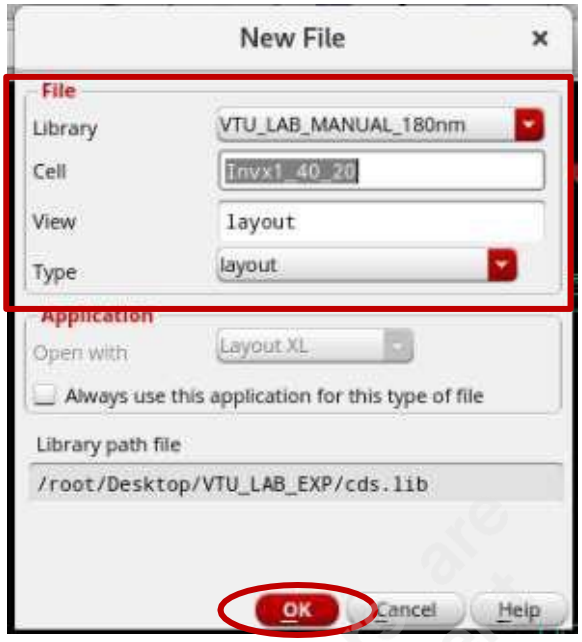


Figure – 1.91: Launch □ Layout XL

The "**Startup Option**" window pops up as shown in Figure – 1.92. Select "**Layout □ Create New**" and "**Configuration □ Automatic**" and click on "**OK**".

Figure – 1.92: Startup Option

The "**New File**" window pops up. Verify the Library Name and Cell Name. "**View**" and "**Type**" should be "**layout**". Click on "**OK**" as shown in Figure – 1.93.



The "**Virtuoso Layout Suite XL Editing**" window pops up as shown in Figure – 1.94. Click on "**F**" to fit the cross wire to the center of the Virtuoso Layout Editor.

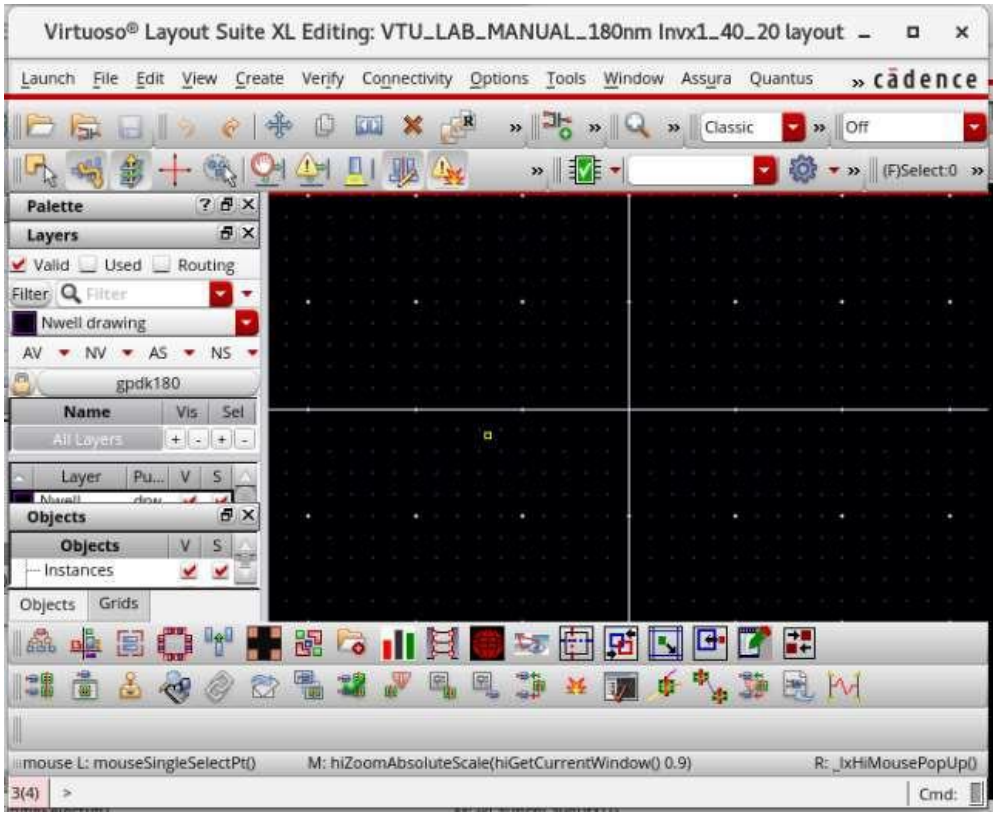
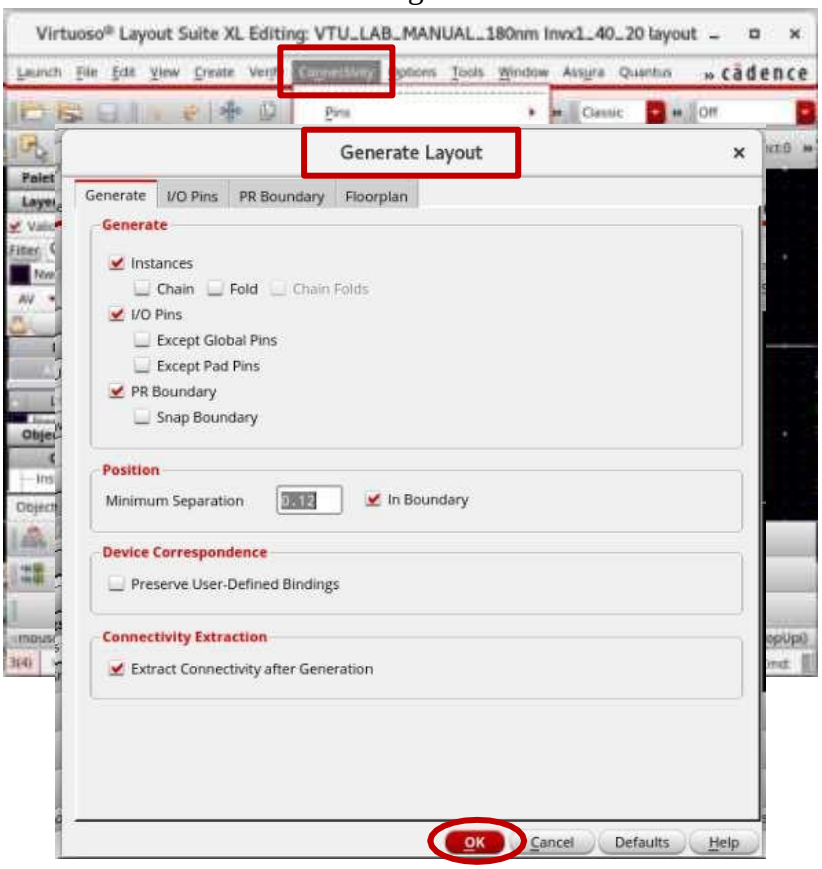


Figure – 1.94: Virtuoso Layout Suite XL Editing

Note:

We have created a Template for gpdk180.

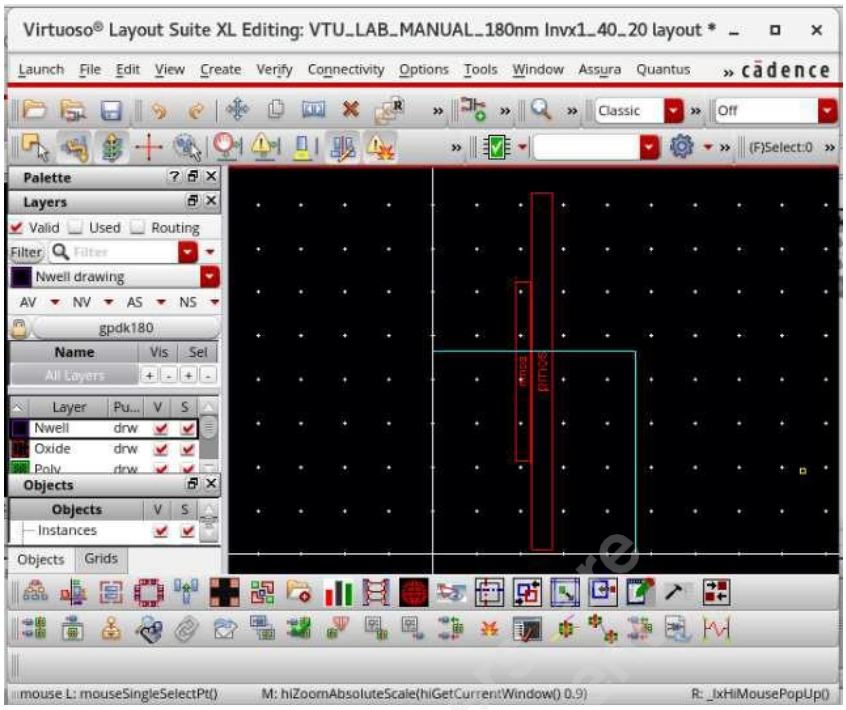
To instantiate all the devices from the Virtuoso Schematic Editor, select "**Connectivity** □ **Generate** □ **All From Source**" as shown in Figure – 1.95.



The "**Generate Layout**" window pops up. Click on "**OK**" as shown in Figure – 1.96.

Figure – 1.96: Generate Layout window

The Virtuoso Layout Editor gets updated as shown in Figure – 1.97.



To view the terminals of the devices, click on "**Shift + F**" and the devices in the Virtuoso Layout Editor gets updated as shown in Figure – 1.98.

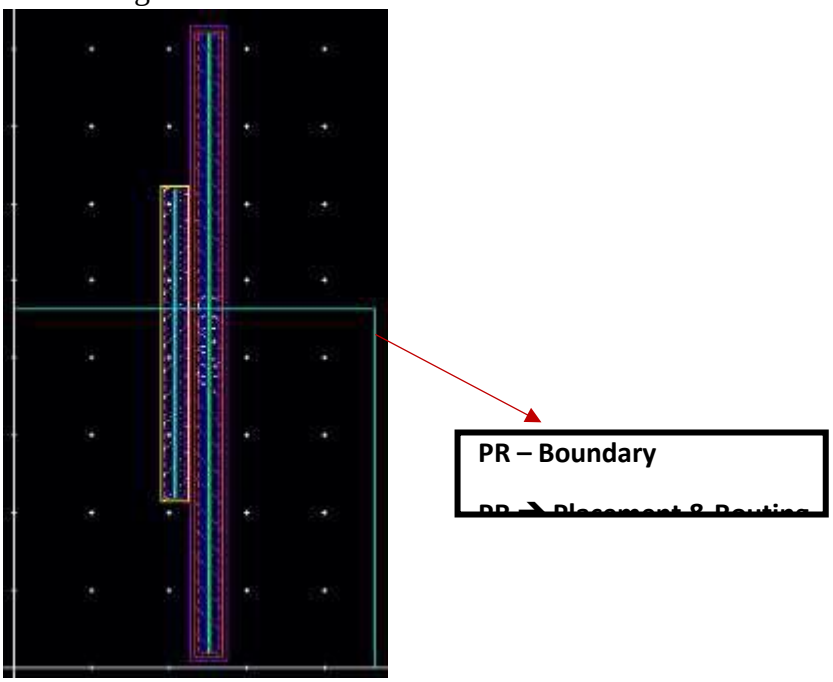


Figure – 1.98: Updated devices after "Shift + F"

The blue colored box that is seen in the Layout is the **PR – Boundary** (**PR Placement and Routing**). Since the devices have to be fixed within the size of the Template, the properties of NMOS and PMOS transistors have to be changed.

To change the device properties, select the device using a Left Mouse Click (for eg: NMOS transistor) and it gets highlighted as shown in Figure – 1.99.

Figure – 1.99: NMOS Transistor after Left Mouse Click

To edit the device properties, use a Right Mouse Click and select "**Properties**" as shown in Figure – 1.100 (or) use the bind key "**Q**".

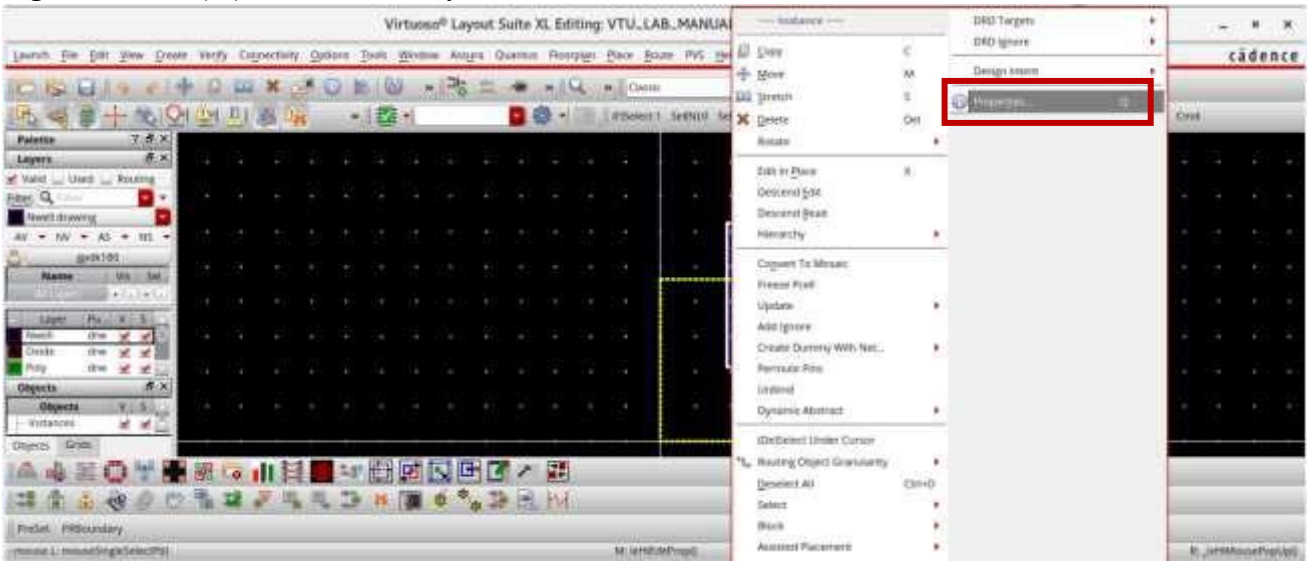


Figure – 1.100: Select "Properties" after a Right Mouse Click

The "**Edit Instance Properties**" window pops up as shown in Figure – 1.101.

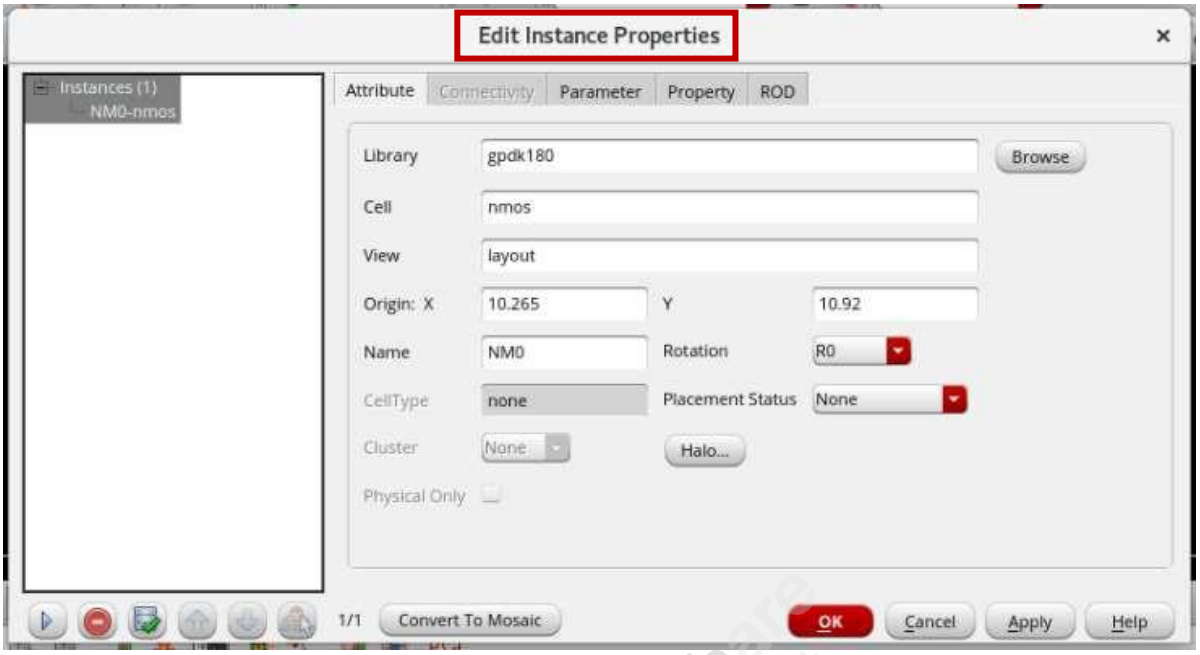


Figure – 1.101: Edit Instance Properties window

Click on "**Parameter**" tab to visualize the parameters of the selected device (for example: NMOS transistor) like Length, Multiplier, Total Width, Finger Width, Fingers and most importantly Bodytie Type as shown in Figure – 1.102. Initially, the "**Bulk**" won't be included to the layout view of Transistors and the "**Bodytie Type**" option helps in including that.

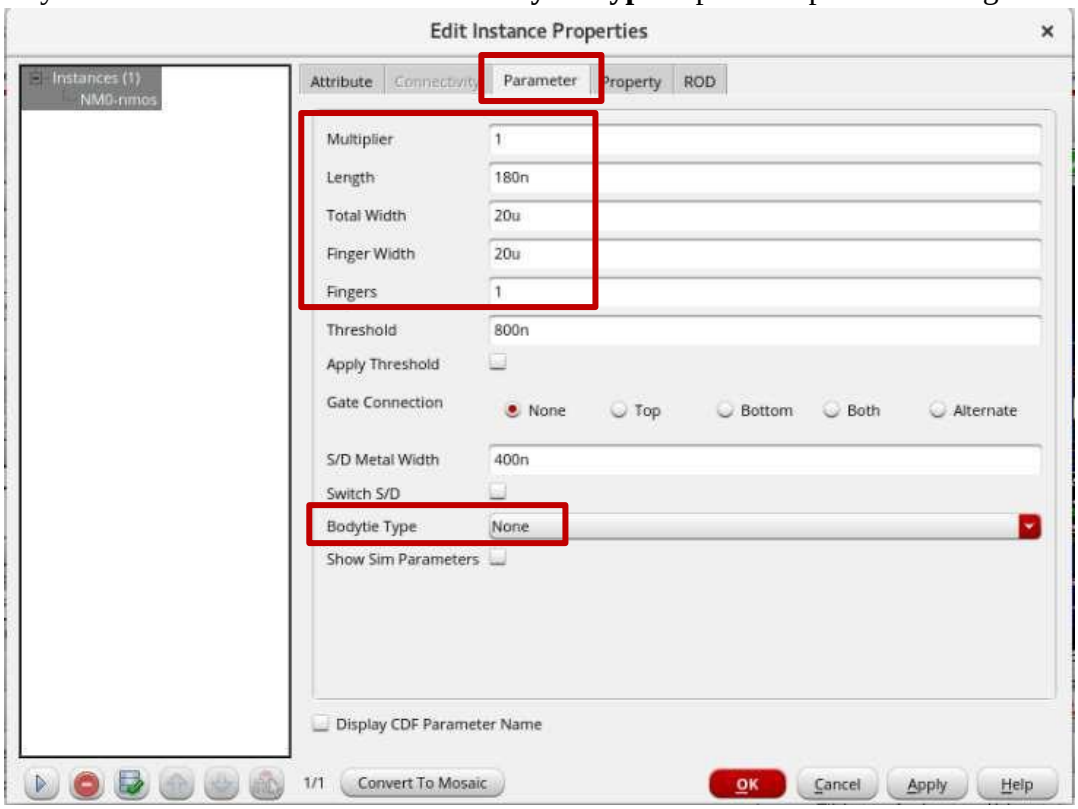
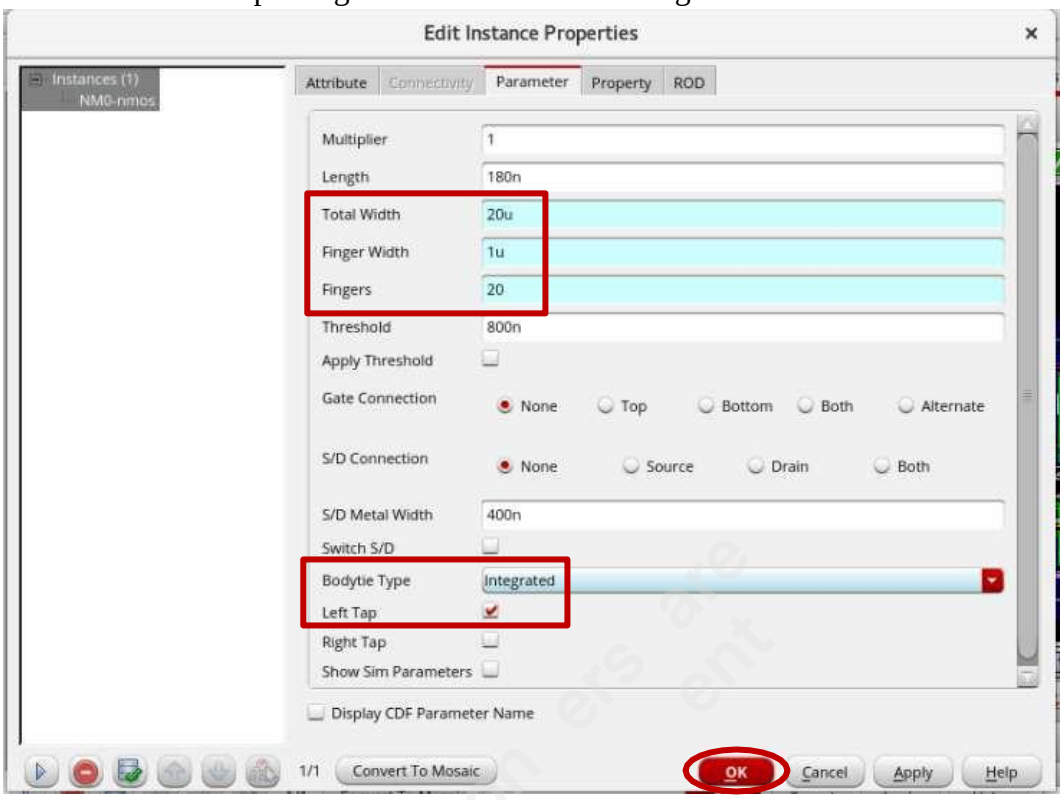


Figure – 1.102: Parameter tab before update

The Parameter tab after updating the values is shown in Figure – 1.103.



In order to fix the device within the Template, the parameter "**Finger □ 20**" and "**Finger Width □ 1u**" are changed but the "**Total Width**" should remain the same. The "**Bodytie Type**

□ **Integrated**" option will have the Bulk terminal integrated to the Source terminal of the device on the left side "**Left Tap**" of the device. Click on "**OK**" and the device gets updated as shown in Figure – 1.104.

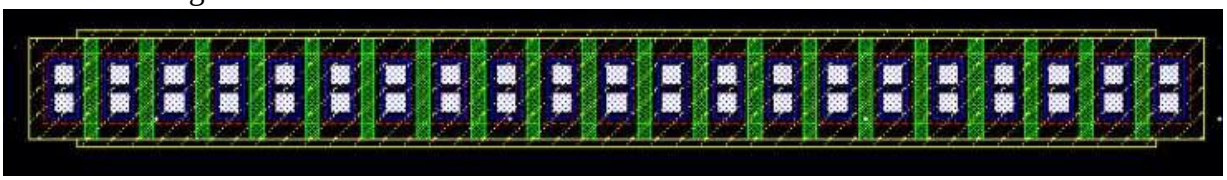
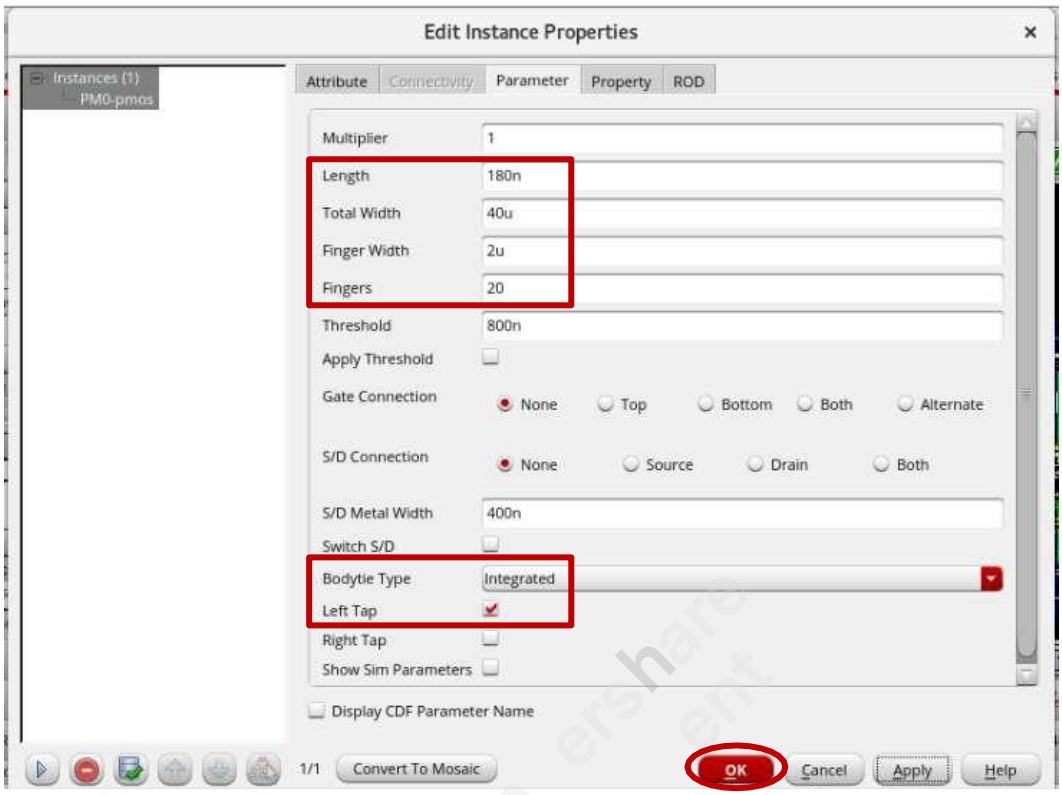


Figure – 1.104: Updated NMOS Transistor

Similarly, the parameters for the PMOS transistors are given as "**Finger □ 20**", "**Finger Width □ 2u**" and "**Bodytie Type □ Integrated**". The updated parameter tab is shown in Figure – 1.105.



Click on "**OK**" and the device gets updated as shown in Figure – 1.106.

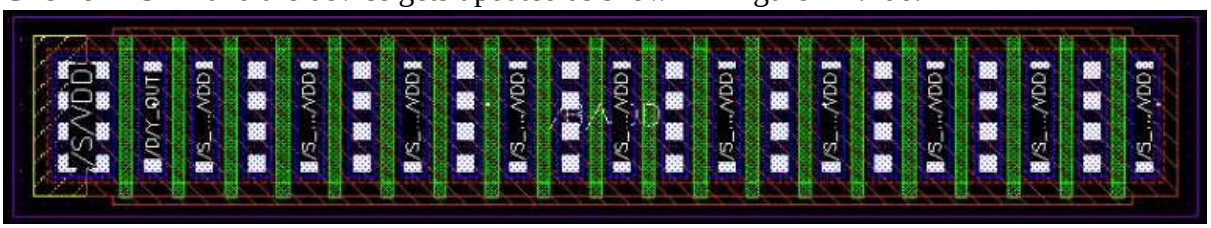


Figure – 1.106: Updated PMOS Transistor

To have an idea of the interconnections, select the layout using "**Ctrl + A**". The entire layout gets highlighted as shown in Figure – 1.107.

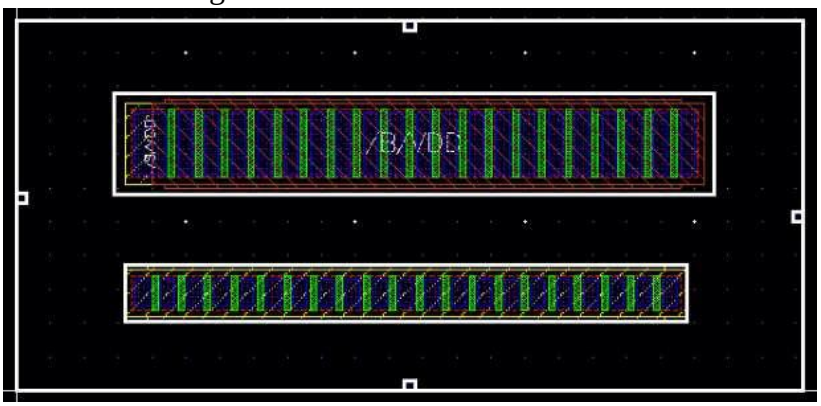
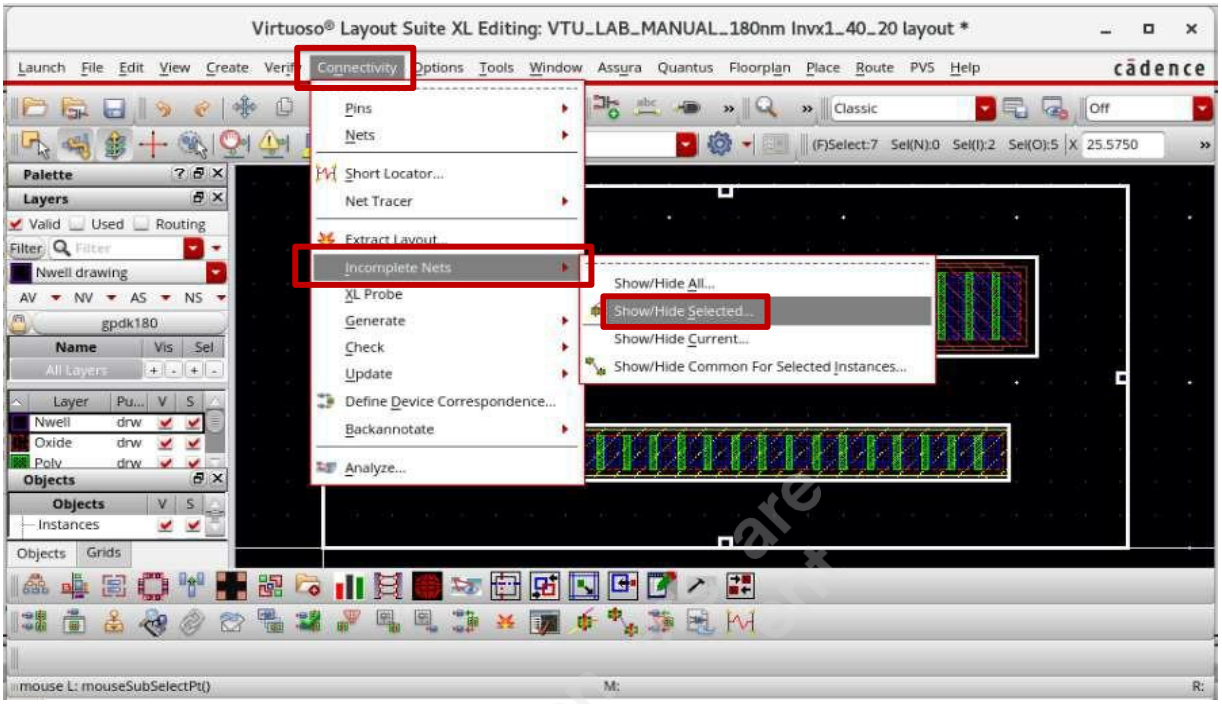


Figure – 1.107: Selecting the layout using "Ctrl + A"

Select "**Connectivity □ Incomplete Nets □ Show/Hide Selected..**" as shown in Figure – 1.108.



The layout gets updated as shown in Figure – 1.109. The lines visible in the layout are called the Rat Lines. These lines give an idea about the missing interconnections in the layout.

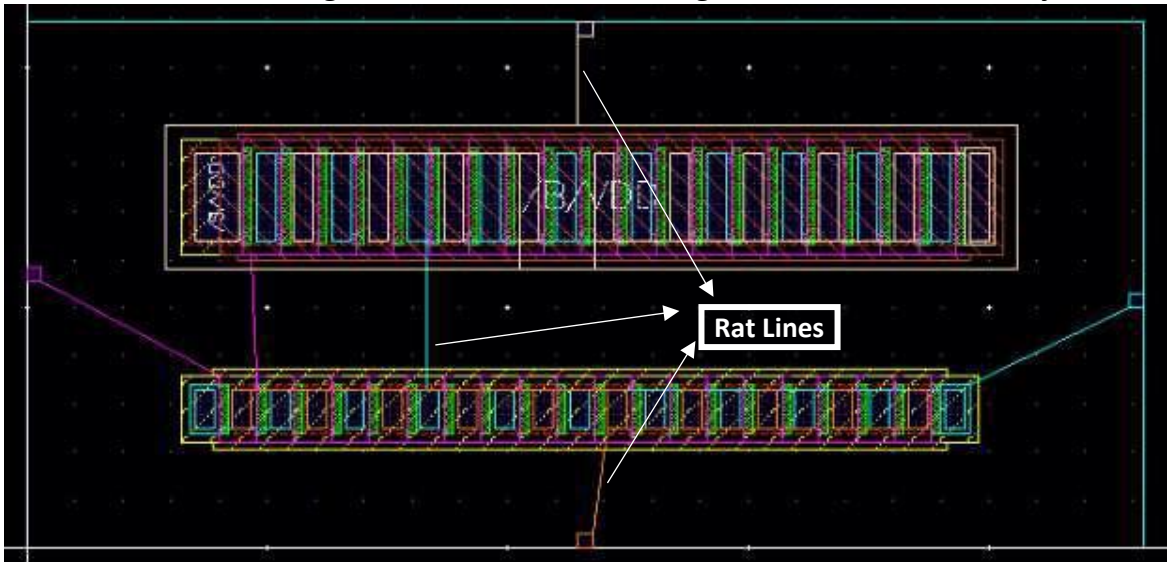


Figure – 1.109: Layout with Rat Lines

Use the bind key "**P**" for the interconnections. After the click on "**P**" in the keyboard, if the mouse pointer is taken close to the terminals in the transistor, it gets highlighted as shown in Figure – 1.110.

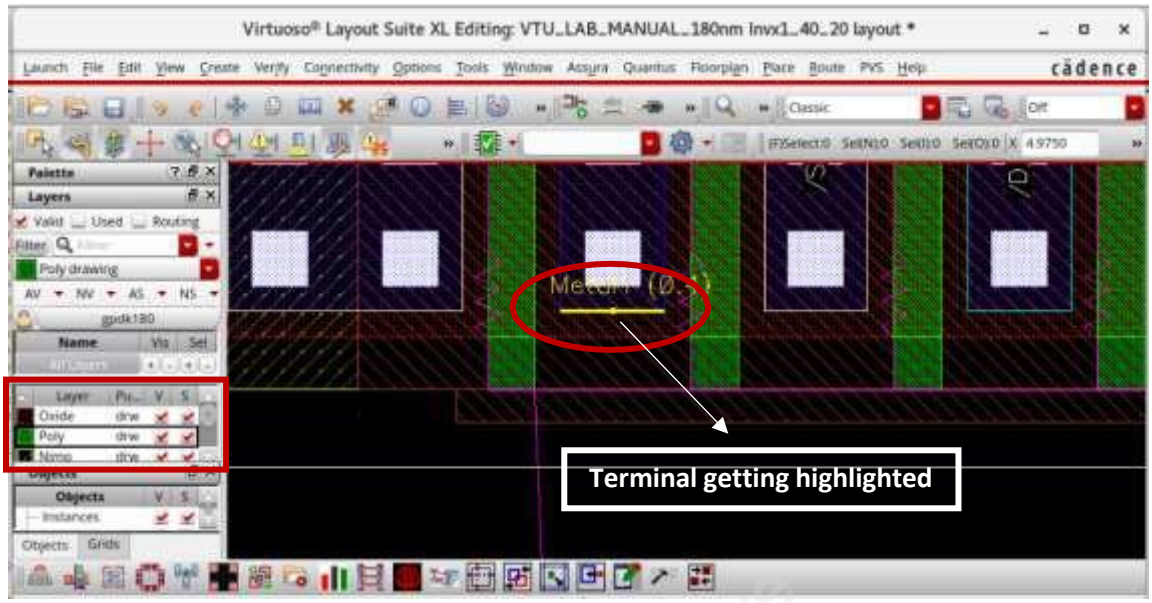


Figure – 1.110: Terminal getting highlighted

Use Left Mouse Click to start the interconnection from the terminal as shown in Figure – 1.111. The option "**Rectangle**" with bind key "**R**" can also be used for interconnections. In case of "**Rectangle**" option, select the respective layer from the "**Layer Palette**" as shown in Figure – 1.110 and then click on "**R**" in the keyboard, use Left Mouse Click to draw the respective layers. Similarly, use the option "**Shift + P**" to create "**Polygon**" which is useful in creating the layers in different shapes other than Rectangle and Square.

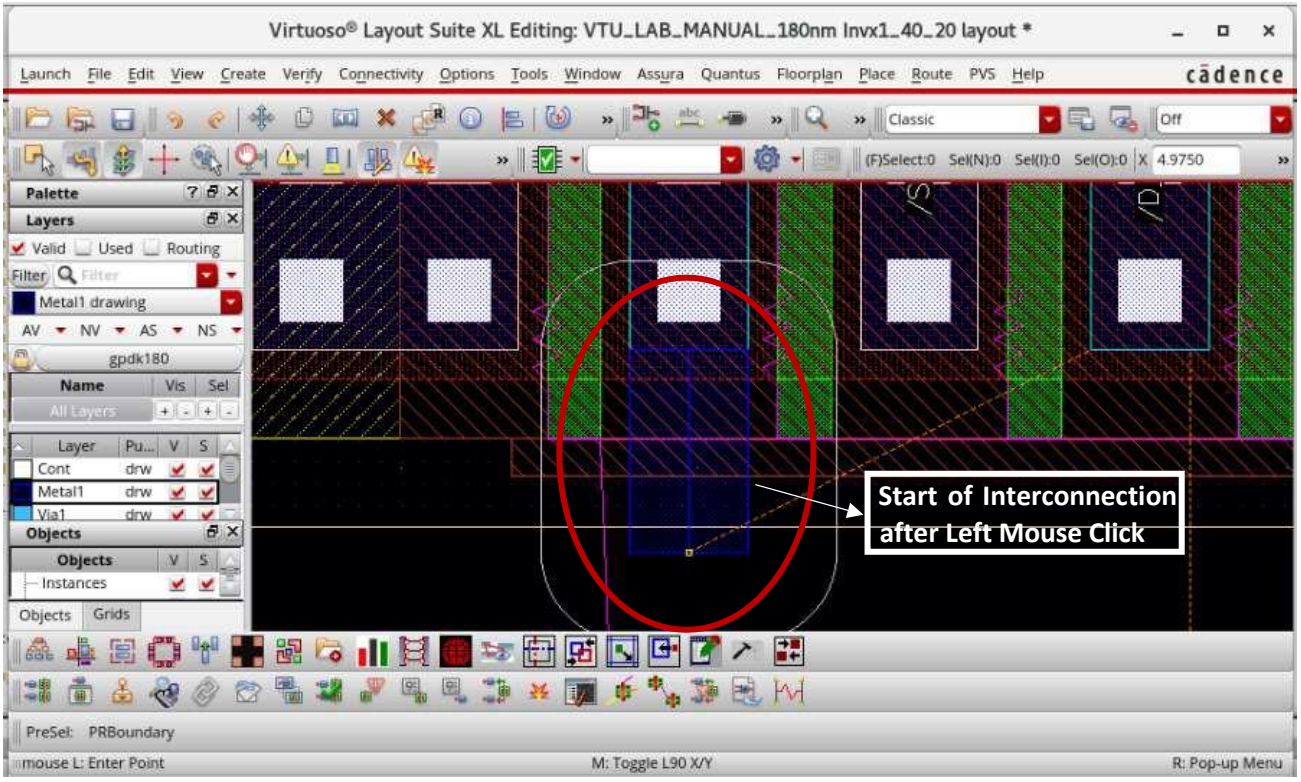
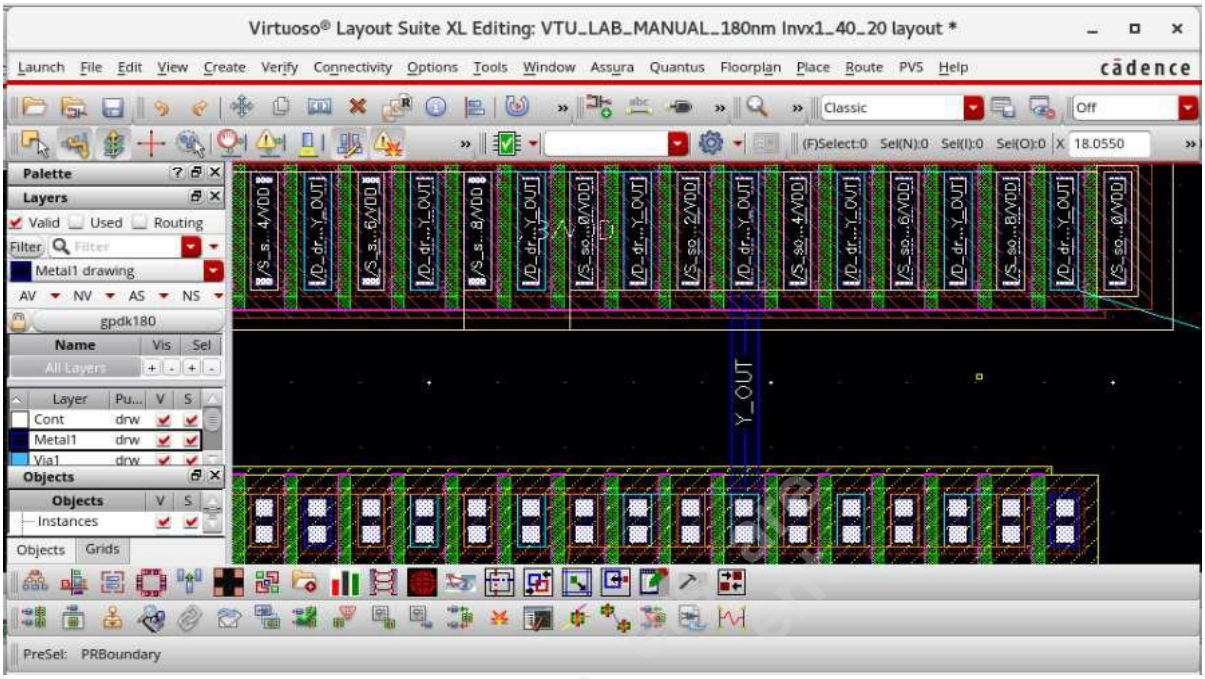


Figure – 1.111: Start of Interconnection after Left Mouse Click

Use the Left Mouse Click at the point where the interconnection has to end. The layout is updated as shown in Figure – 1.112.



Repeat the steps to complete the remaining connections in the layout.

For interconnections between two different layers, use a "**Via**". Via has to be selected according to the layers present at the Start and End of the interconnections. To include a Via in the layout, select "**Create** □ **Via**" as shown in Figure – 1.113.

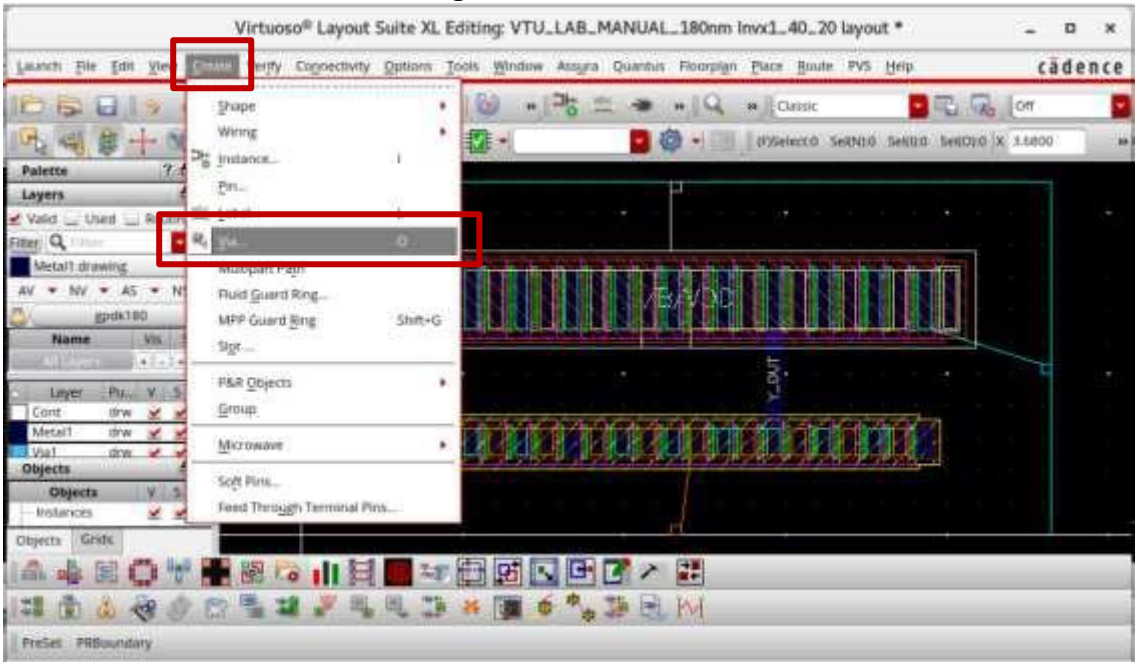
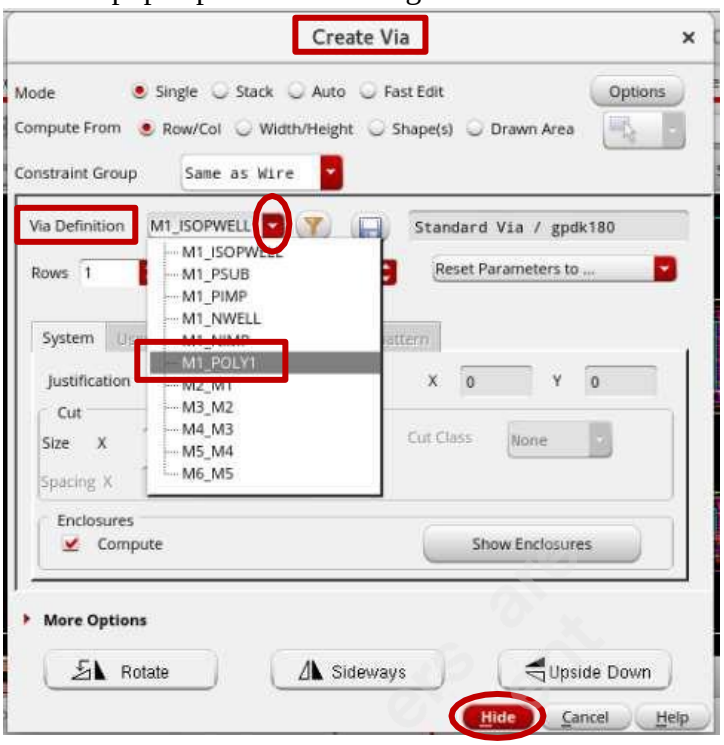


Figure – 1.113: Create □ Via

The "**Create Via**" window pops up as shown in Figure – 1.114.



The required Via can be selected through the "**Via Definition**" option as shown in Figure – 1.114. Click on the drop down and select the required Via. For example, in the CMOS Inverter design, the Input pin "**A_IN**" is of **Metal 1** layer and it has to be connected to the Gate terminal of PMOS and NMOS Transistors which is a **Poly** layer. So, from the Via Definition, **M1_POLY1** is selected as shown in Figure – 1.114. Click on "**Hide**" to visualize the Via on the Virtuoso Layout Editor as shown in Figure – 1.115. Use a Left Mouse Click to place the Via.

Figure – 1.115: Selected Via

Use the bind key "**P**" to complete the connections between the Input Pin and Via and between Via and Gate Terminal of the transistor. The completed layout can be visualized as shown in Figure – 1.116(a).

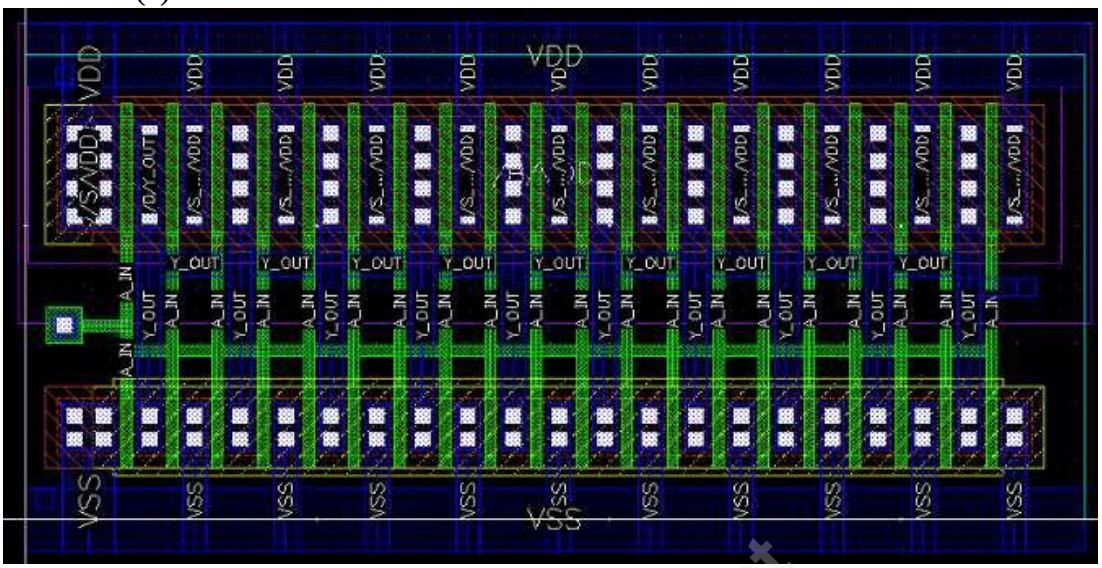


Figure – 1.116(a): Completed Layout

With the Template shown, the layout can be visualized as shown in Figure – 1.116(b).

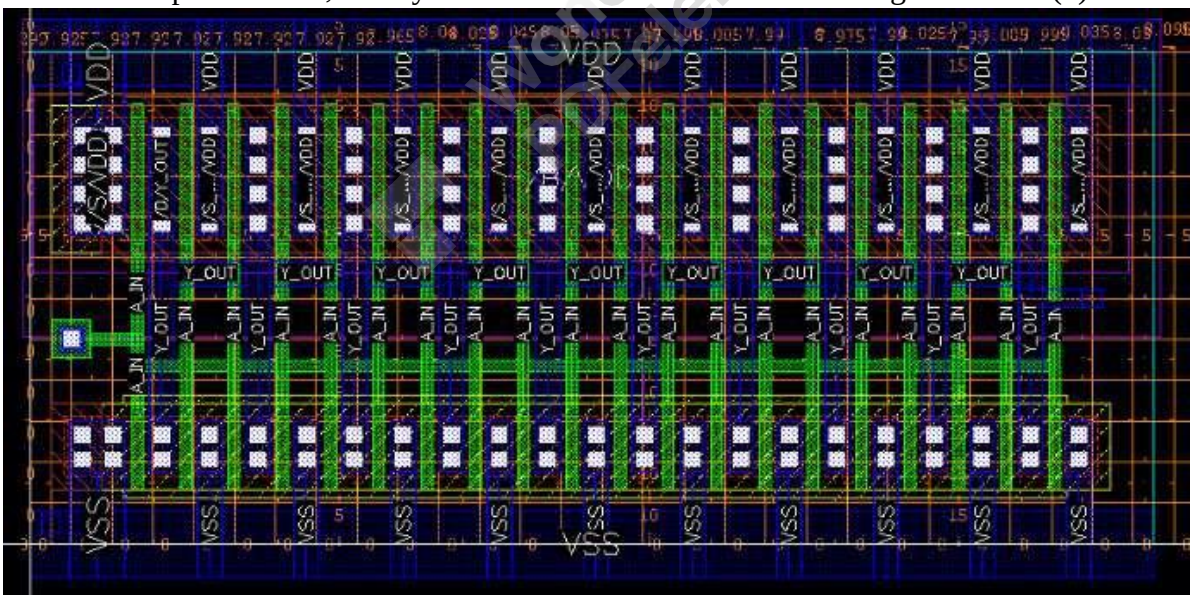


Figure – 1.116(b): Completed Layout with Template

Click on "**File □ Save**" to **Save** the layout.

PHYSICAL VERIFICATION WITH ASSURA:

Physical Verification involves **DRC** (Design Rule Check) and **LVS** (Layout versus Schematic) checks on the layout. These checks are performed using Assura.

TECHNOLOGY LIBRARY (assura_tech.lib) MAPPING:

To map the library file, select "**Assura □ Technology..**" as shown in Figure – 1.117.

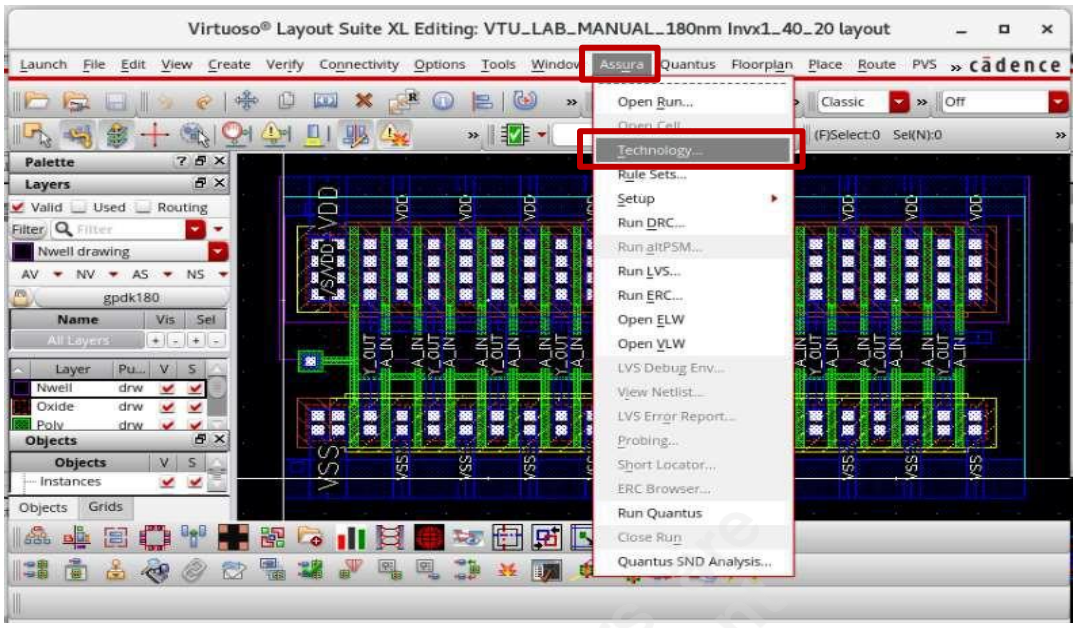
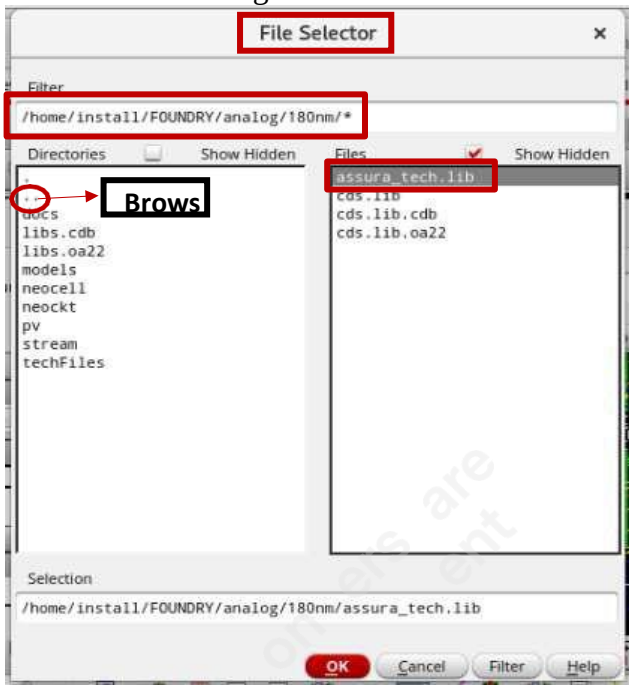


Figure – 1.117: Assura ➔ Technology..

`The "**Assura Technology Lib Select**" window pops up as shown in Figure – 1.118.

Figure – 1.118: Assura Technology Lib Select

Click on "**Browse**" option as shown in Figure – 1.118. The "**File Selector**" window pops up



as shown in Figure–1.119.

Figure – 1.119: File Selector window

Click on the "**Browse**" as shown in Figure – 1.119 to select the file "**assura_tech.lib**" from the location "**/home/install/FOUNDRY/analog/180nm/** ". Once a double-mouse click in done on "**assura_tech.lib**", the path gets completed as shown in Figure – 1.120. Click on "**OK**".

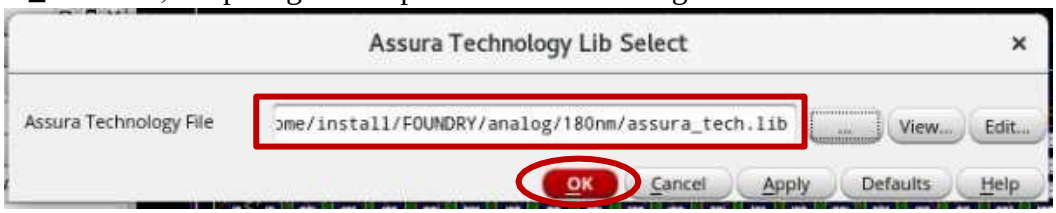


Figure – 1.120: Assura Technology Lib Select window after file selection

DRC (DESIGN RULE CHECK):

To run DRC check using Assura, select "**Assura** □ **Run DRC**" as shown in Figure – 1.121.

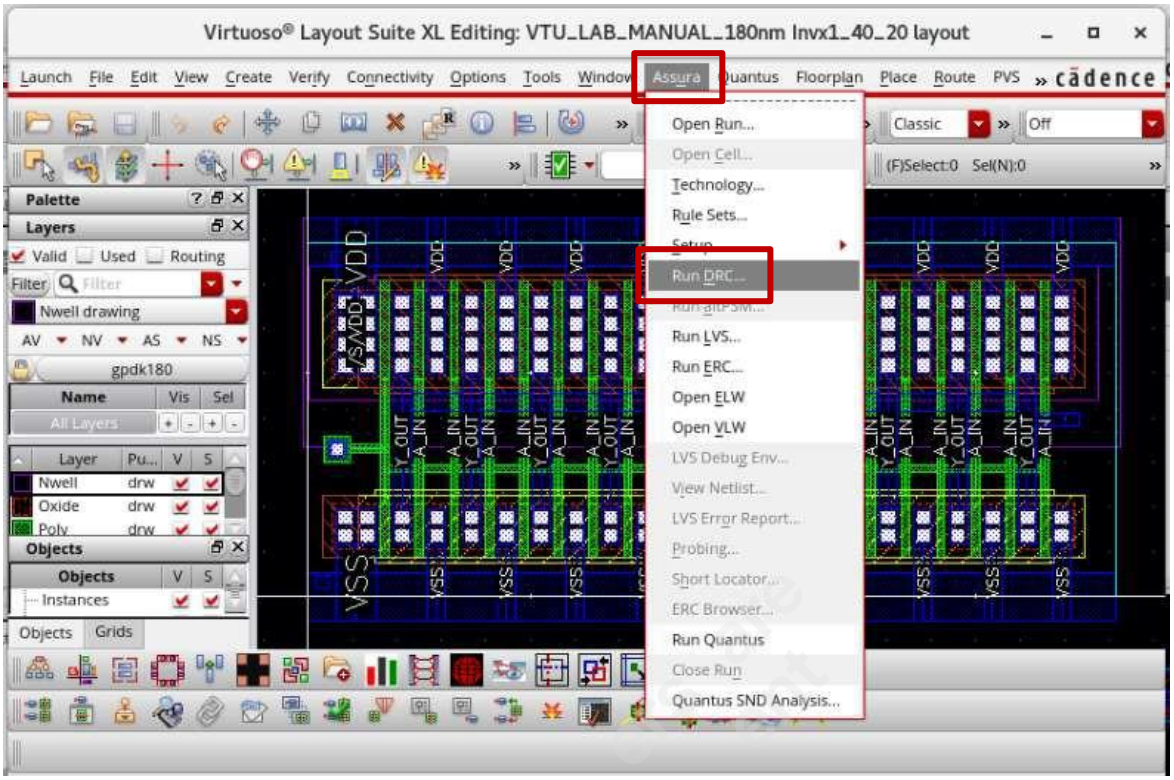


Figure – 1.121: Assura ➔ Run DRC

The "**Run Assura DRC**" window pops up as shown in Figure – 1.122.

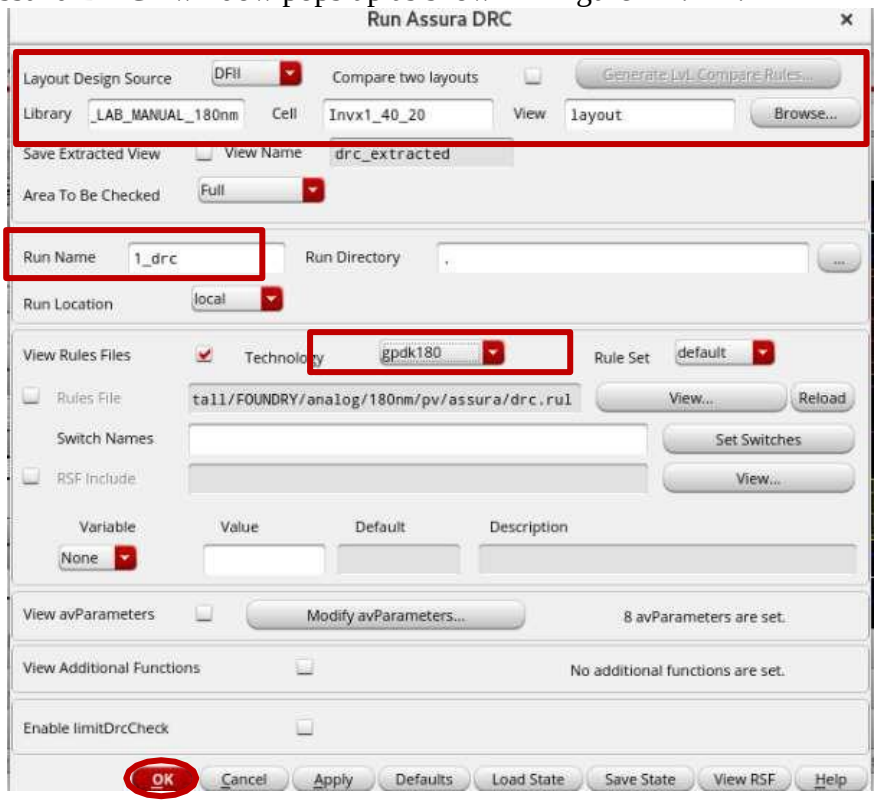


Figure – 1.122: Run Assura DRC window

Check for the "**Layout Design Source**", mention a "**Run Name**" (it can be any name) and select "**Technology □ gpdk180**" from the drop down and click on "**OK**" as shown in Figure – 1.122. The "**Progress**" window pops up as shown in Figure – 1.123.

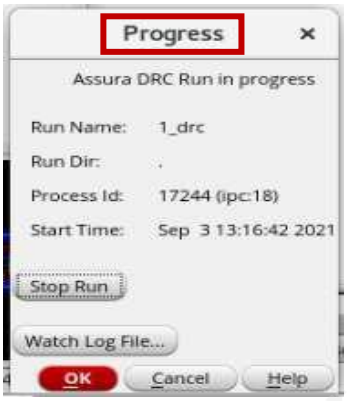


Figure – 1.123: "Progress" window

Once the DRC check is over, we get the DRC check completion window as shown in Figure – 1.124.

Figure – 1.124: DRC Check completion window

Click on "**Yes**" to get the results of DRC Check as shown in Figure – 1.125.

Figure – 1.125: Result of DRC Check if the layout is DRC clean

In case of errors, the error information will be shown. If the error is selected, it points out the issue which has to be reworked on the layout. Once done, Save the layout and re-run the DRC check to make sure that the layout is DRC clean.

LVS (LAYOUT VERSUS SCHEMATIC):

To run the LVS check using Assura, select "**Assura □ Run LVS**" as shown in Figure – 1.126.

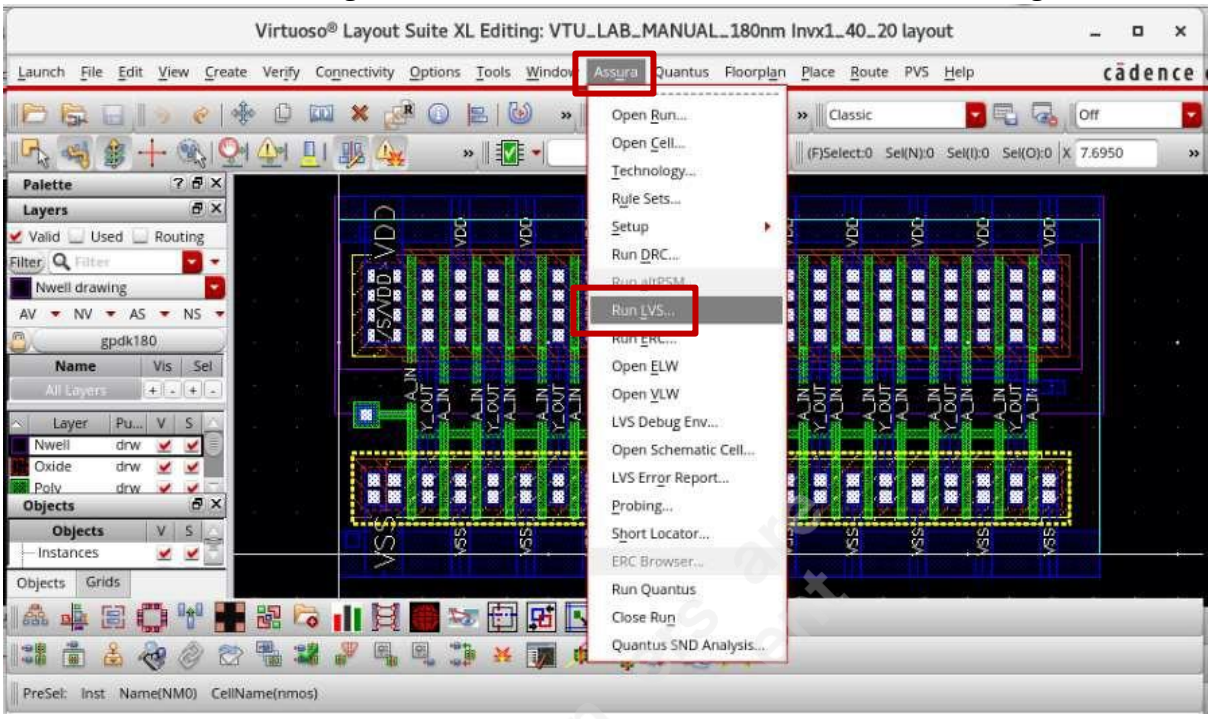


Figure – 1.126: Assura ➔ Run LVS

The "**Run Assura LVS**" window pops up as shown in Figure – 1.127.

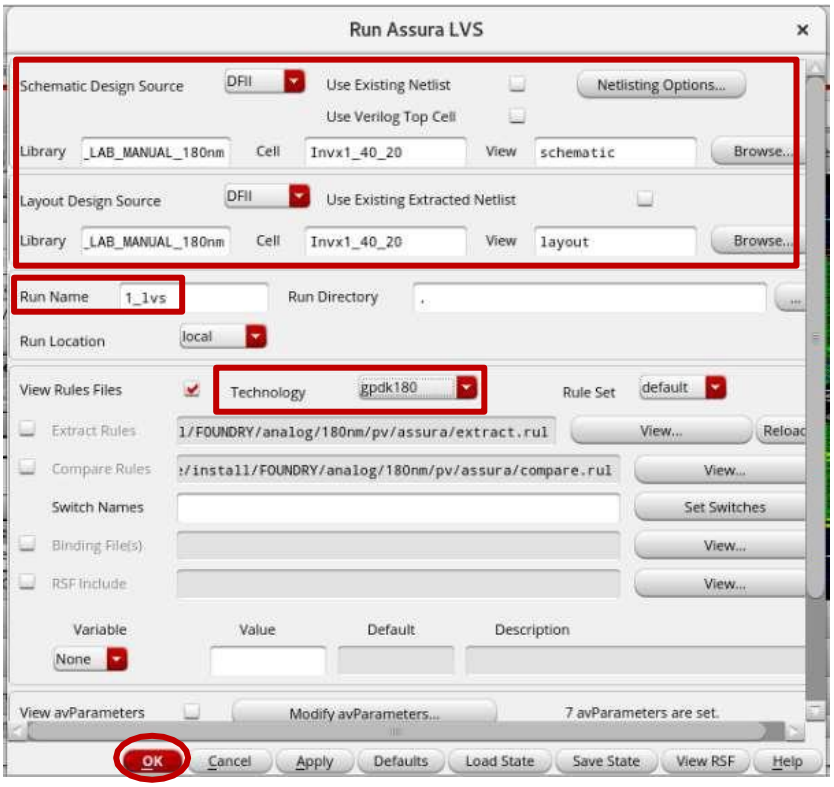


Figure – 1.127: Run Assura LVS window

Check for the correctness of the Schematic and Layout to be compared in the "**Schematic Design Source**" and the "**Layout Design Source**", mention a "**Run Name**" (it can be any name but avoid space) and select the **Technology** (for example: gpdk180) as shown in Figure – 136. Click on "**OK**". The progress of "**LVS**" check can be seen as shown in Figure – 137.

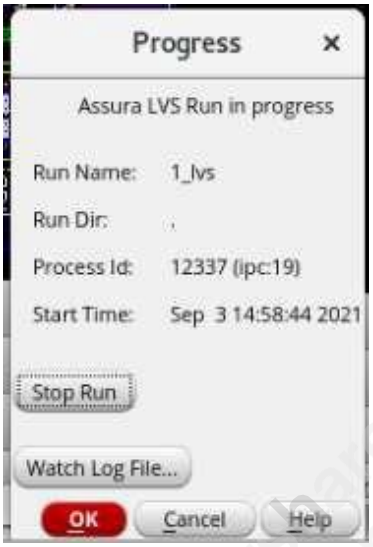


Figure – 1.128: LVS "Progress" window

After the LVS check gets completed, the "**Run: "1_lvs"**" window pops up. In case of violations in LVS check, the total number of violations can be seen as shown in Figure – 1.129. Since there are no violations, it shows as "**0**".

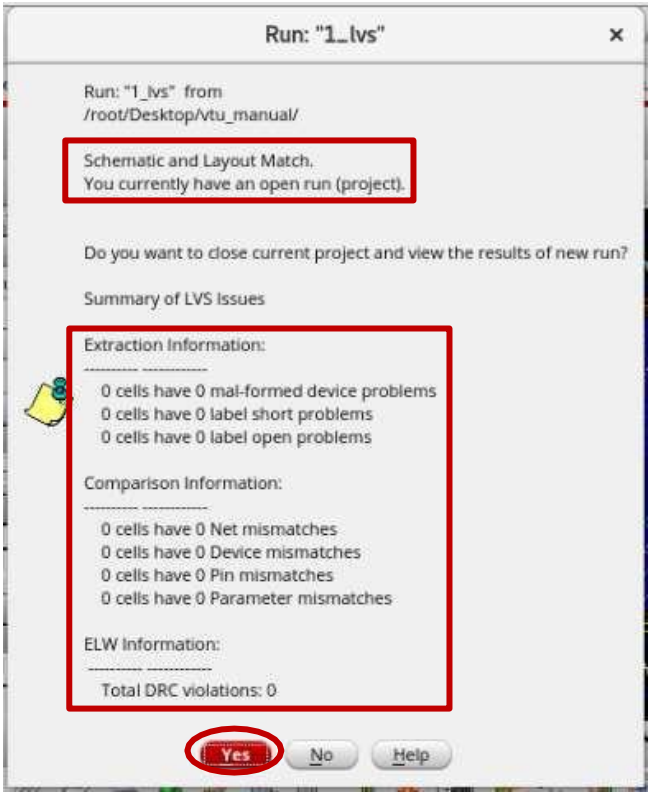


Figure – 1.129: "Run: "1_lvs"" window

Click on "**Yes**" to see the result in the "**LVS Debug**" window as shown in Figure – 1.130.

Figure – 1.130: "LVS Debug" window

Since the design is LVS clean, the message "**Schematic and Layout Match**" can be seen. In case of violations, the respective messages are listed out.

QRC (RC / PARASITIC EXTRACTION):

The tool used for Parasitic Extraction process is "**Quantus**". Select "**Assura □ Run Quantus**" as shown in Figure – 1.131 to invoke the tool and enter the "**Quantus (Assura) Parasitic Extraction Run Form**".

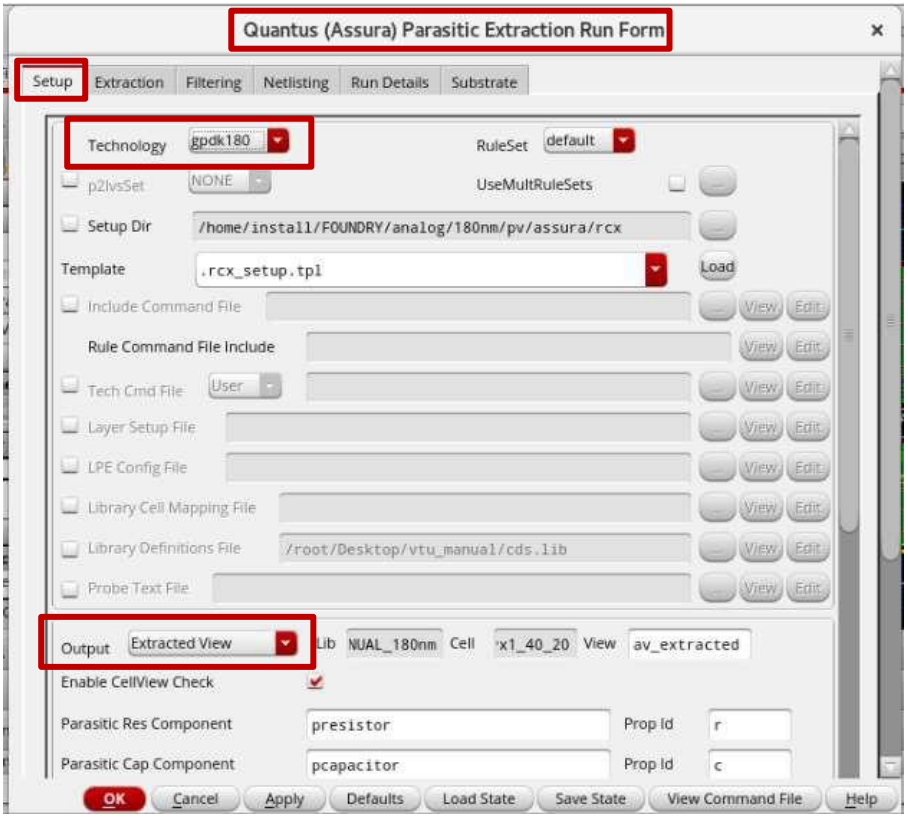


Figure – 1.131: Quantus (Assura) Parasitic Extraction Run Form

Click on the "**Setup**" tab, check for "**Technology gpdk180**" and select "**Output Extracted View**" as shown in Figure – 1.131. Click on "**Extraction**" tab and the options can be seen as shown in Figure – 1.132.

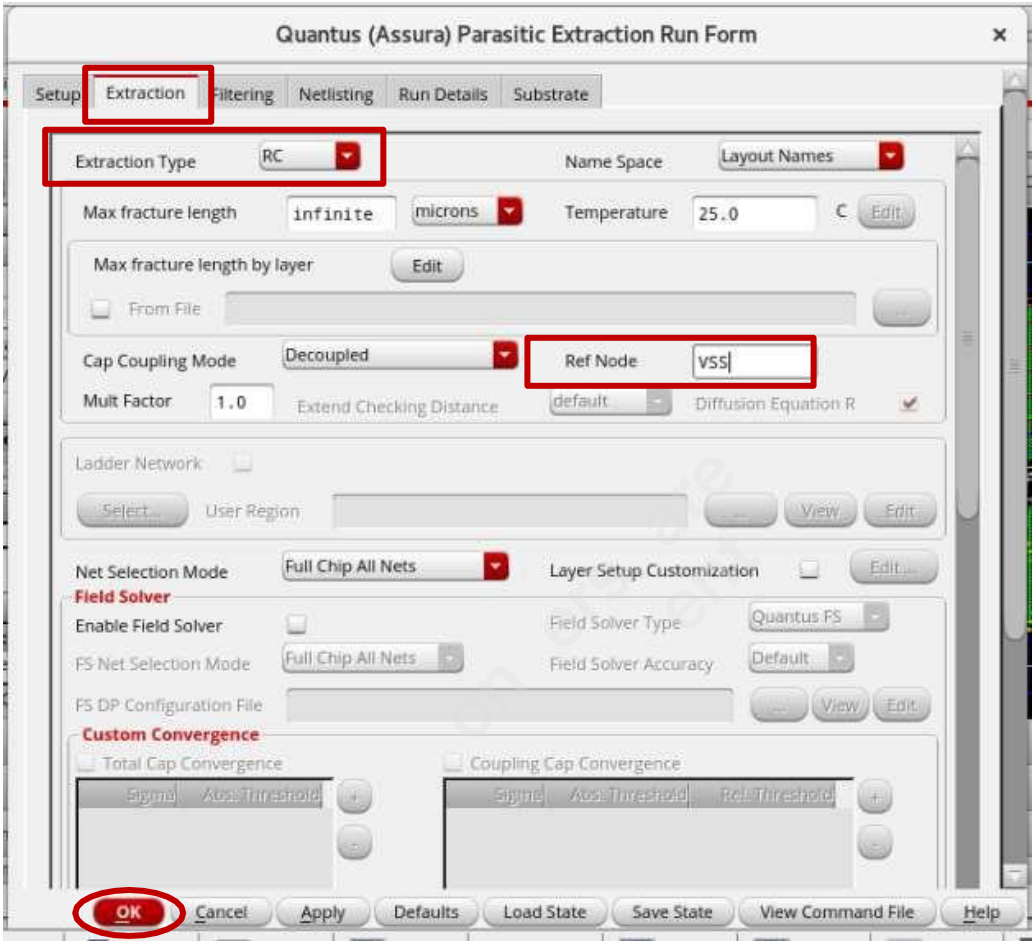


Figure – 1.132: "Extraction" Tab

Select "**Extraction Type RC**" and the other options like "**R only**", "**C only**" and others can be checked as per the requirements. Select the "**Ref Node VSS**" and click on "**OK**" as shown in Figure – 1.132. The "**Quantus Progress Form**" can be seen as shown in Figure – 1.133.

Figure – 1.133: Quantus Progress Form

After Extraction, the "**Quantus Run**" form pops up with the "**av_extracted**" file's location as shown in Figure – 1.134.

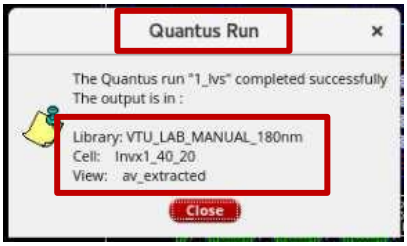


Figure – 1.134: Quantus Run form

The details of Extracted Parasitics are available with the av_extracted view and the file can be opened from the Library Manager as shown in Figure – 1.135.

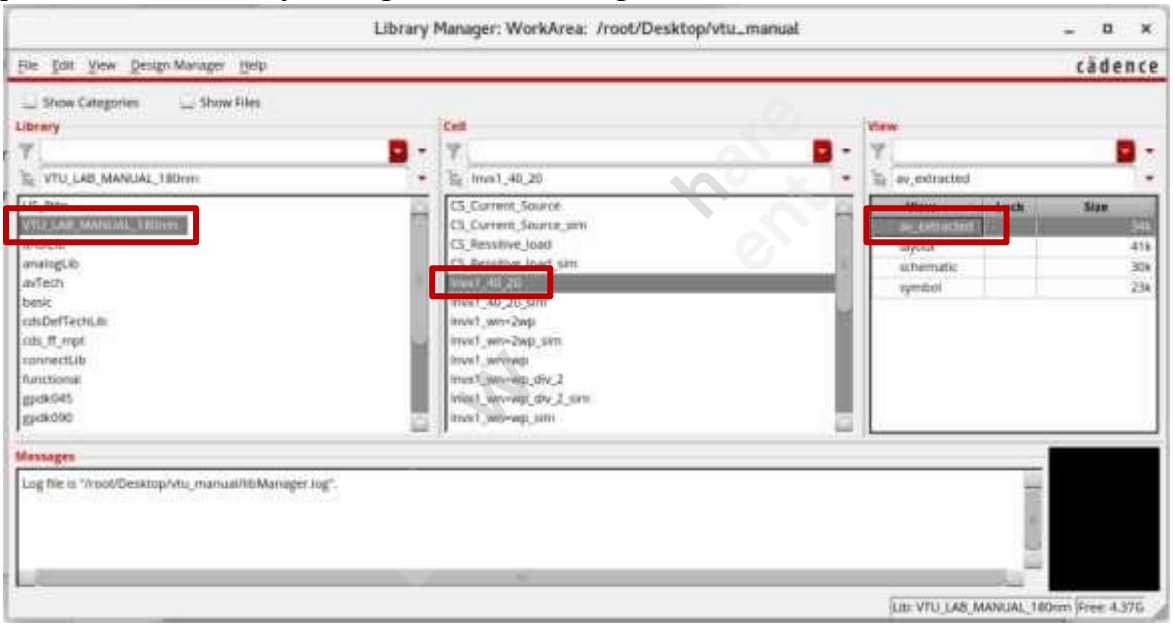


Figure – 1.135: "av_extracted" view from Library Manager

Double Click on "av_extracted" view to see the Extracted View of the layout as shown in Figure – 1.136.

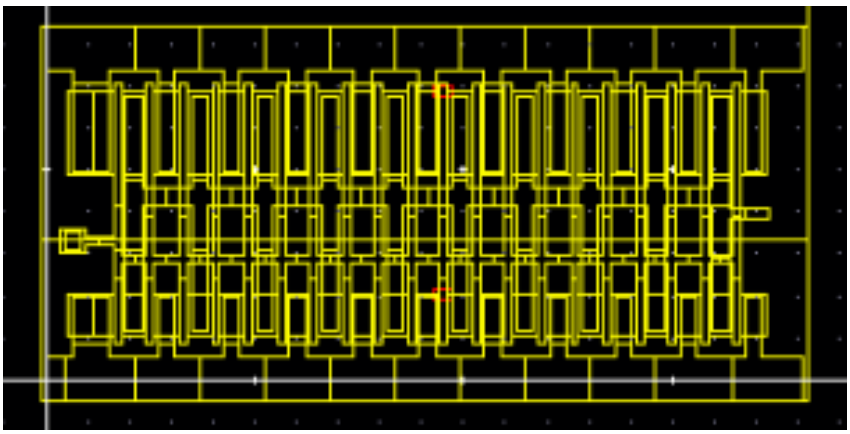
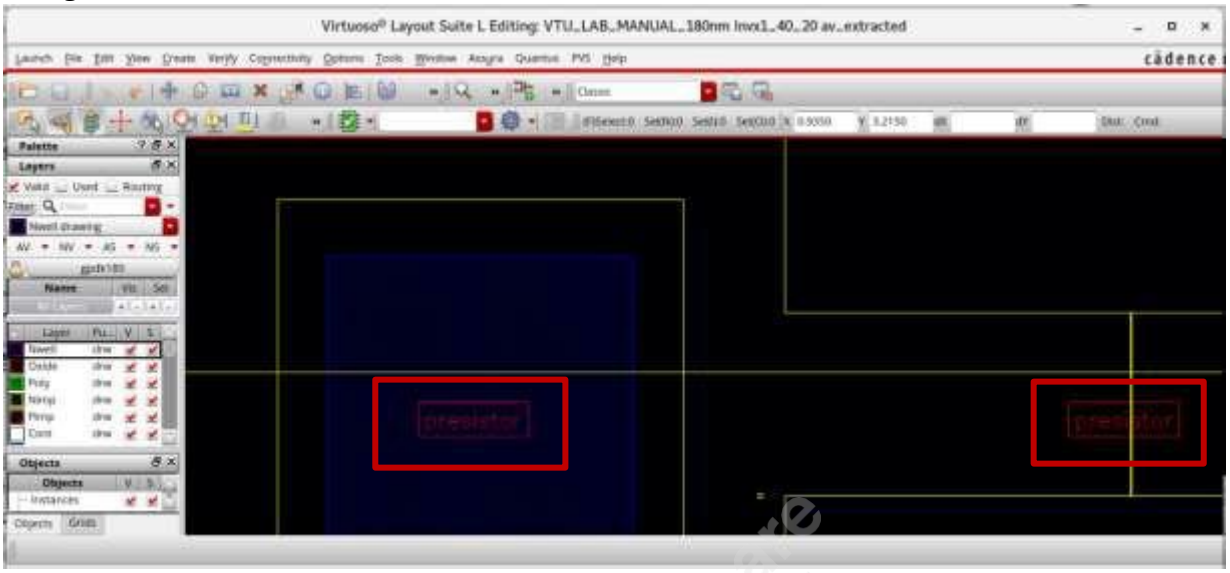


Figure – 1.136: Extracted View

Use the Mouse Scroller to "**Zoom In**" and "**Zoom out**" in order to view the parasites as shown in Figure – 1.137.



further, the values can be checked out as shown in Figure – 1.138.

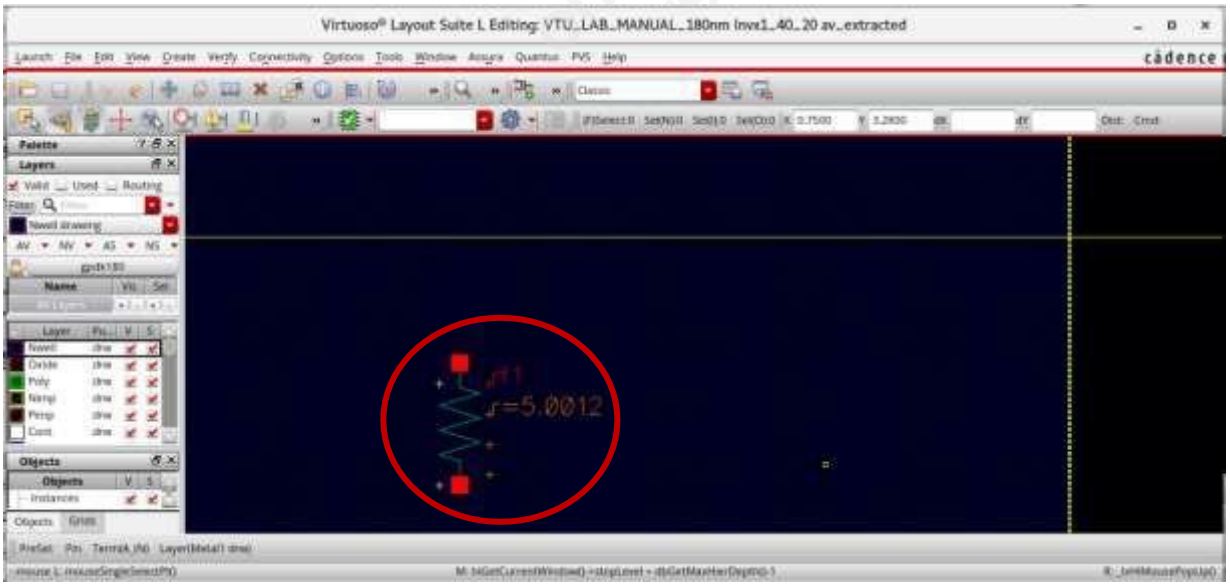


Figure – 1.138: Value of Parasitic Resistance

The impact of these parasitic devices can be checked out through the Backannotation (Post Layout Simulation) process.

BACKANNOTATION (POST LAYOUT SIMULATION):

To run the Post Layout Simulation, the extracted Parasitics have to be imported into the Test Schematic. So, a New Configuration has to be created.

To create a "**New Configuration**" select the Cell which has the Test Schematic and select its "**Schematic**" view as shown in Figure – 1.139.

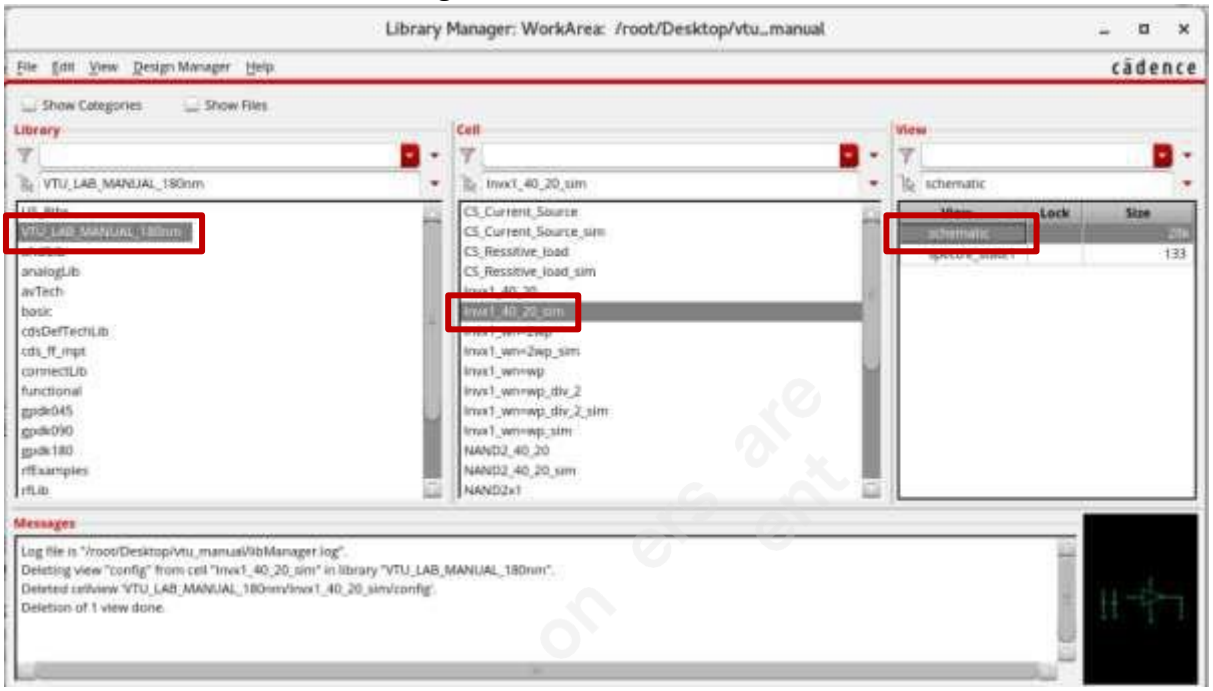


Figure – 1.139: Selecting the "Schematic" view

Click on "**File □ New □ Cell View**" as shown in Figure – 1.140.

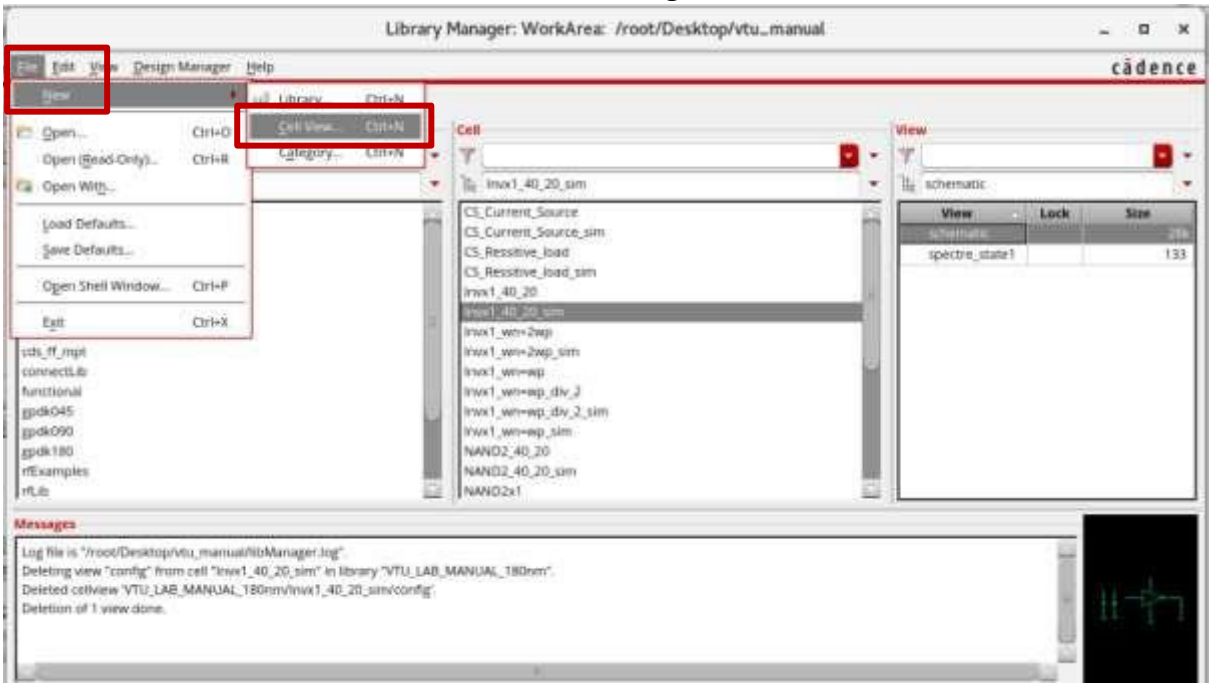


Figure – 1.140: File □ New □ Cell View

The "**New File**" window pops up. Select the "**Type □ config**" from the drop down as shown in Figure – 1.141. Soon as the "**Type □ config**" is selected, "**View □ config**" and in "**Application**", "**Open with □ Hierarchy Editor**" gets updated. Click on "**OK**".

The "**New Configuration**" window pops up as shown in Figure – 1.142. Click on "**Use Template**".

Figure – 1.142: "New Configuration" window

The "**Use Template**" window pops up as shown in Figure – 1.143.

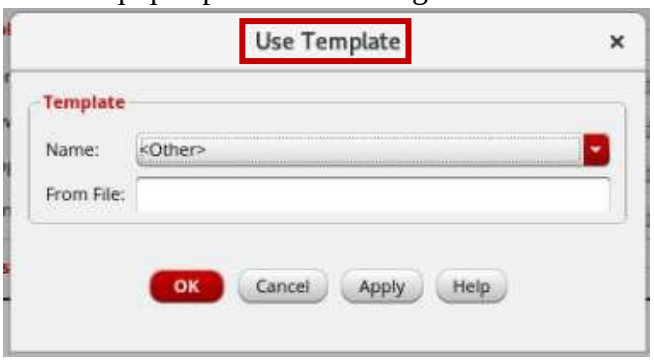


Figure – 1.143: Use Template window

Click on the drop down and select "**Name Spectre**", the name of the Simulator and click on "**OK**" as shown in Figure – 1.144.

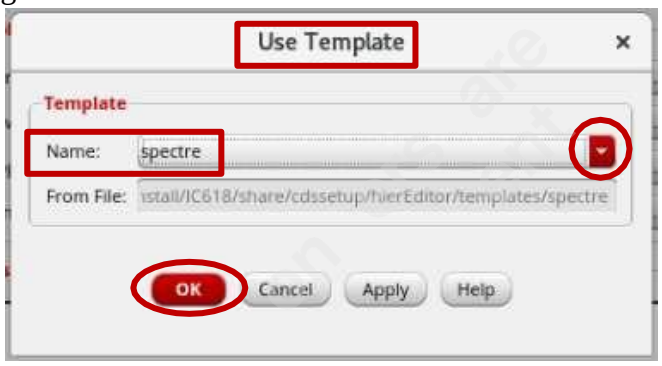


Figure – 1.144: Name Spectre

The updated "**New Configuration**" window pops up as shown in Figure – 1.145.

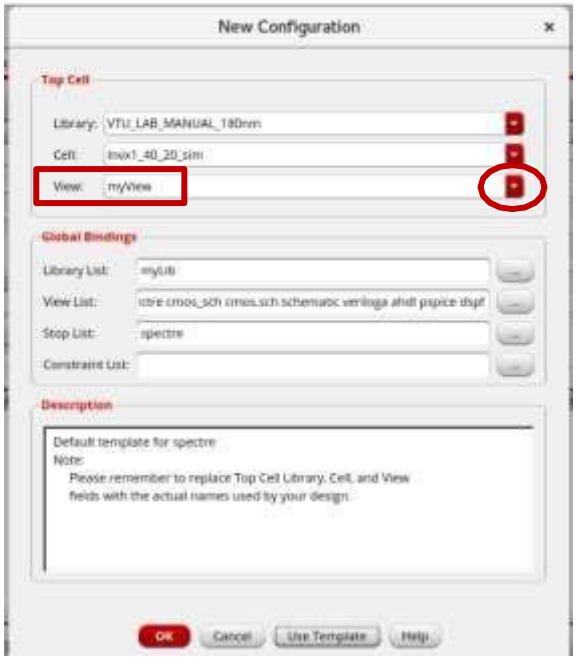


Figure – 1.145: New Configuration window

The "**Top Cell □ View □ Schematic**" has to be selected using the drop down as shown in Figure – 1.145. The "**New Configuration**" window gets updated as shown in Figure – 1.146.

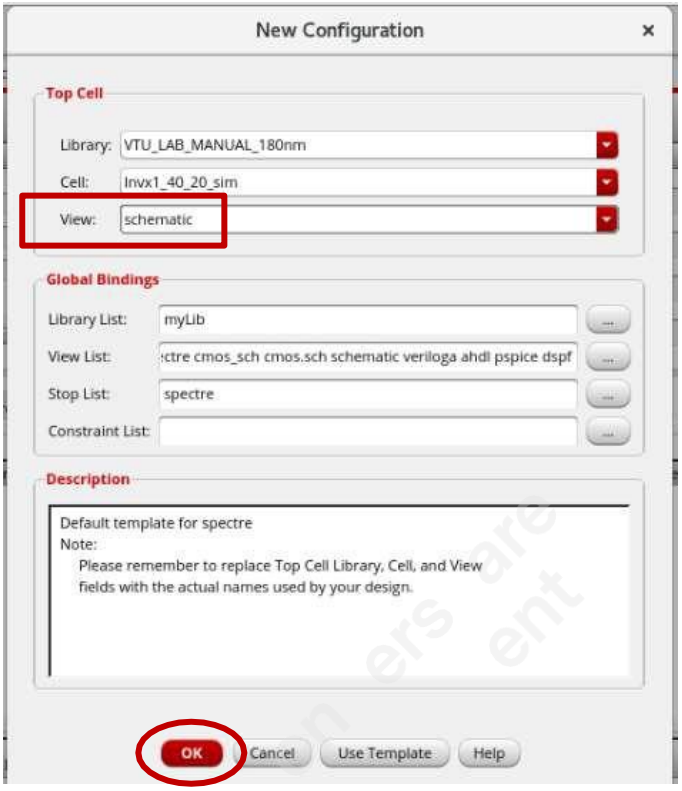


Figure – 1.146: Top Cell □ View □ Schematic

Click on "**OK**" and the "**Virtuoso Hierarchy Editor: New Configuration**" window pops up as shown in Figure – 1.147.

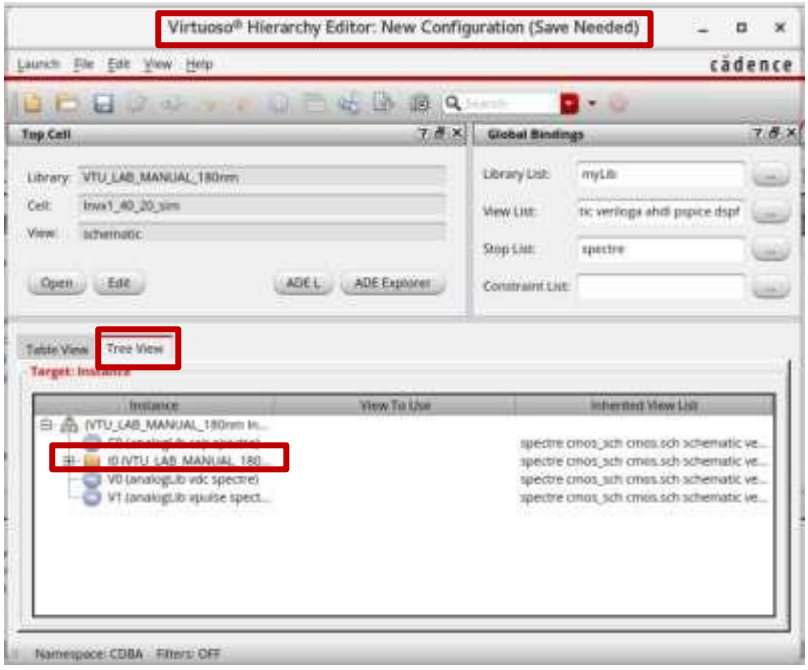


Figure – 1.147: Virtuoso Hierarchy Editor: New Configuration window

Two types of views, "**Table View**" and "**Tree View**" can be seen. Select "**Tree View**" as shown in Figure – 1.147, the instance "**I0**" which is the Instance number of the Symbol with which we had created the Test Schematic can be seen.
Select the Instance "**I0**" as shown in Figure – 1.148, make a Right Click, select "**Set Instance View □ av_extracted**".

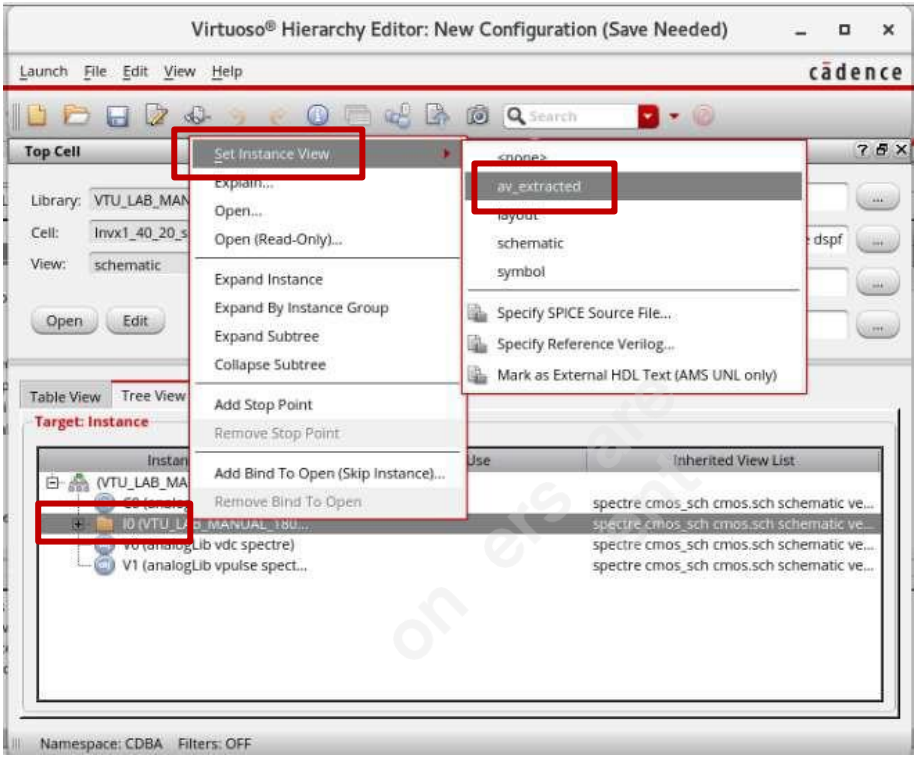


Figure – 1.148: Set Instance View □ av_extracted

Click on the " + " sign before the instance "**I0**" to see the imported parasitics as shown in Figure – 1.149.

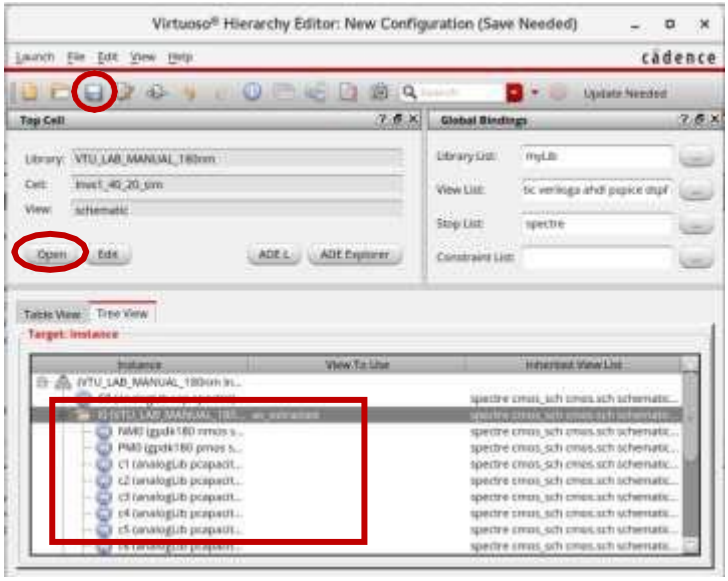


Figure – 1.149: Imported Parasitics

Click on the "**Save**" option, click on "**Open**" to bring back the Test Schematic as shown in Figure – 1.150.

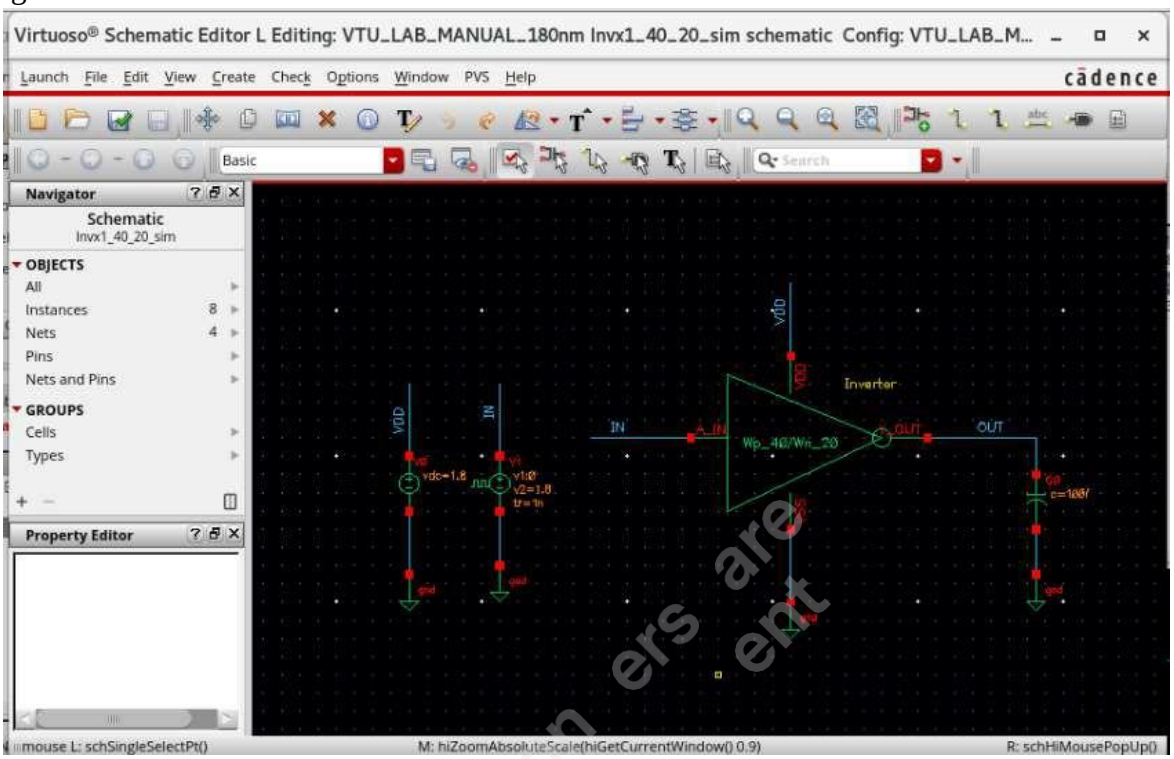


Figure – 1.150: Test

To verify if the parasitics are imported, double click on the Inverter symbol, the "**Descend**" window pops up as shown in Figure – 1.151. Check if "**View** □ **av_extracted**" and select "**Open in** □ **new tab**" and click on "**OK**"

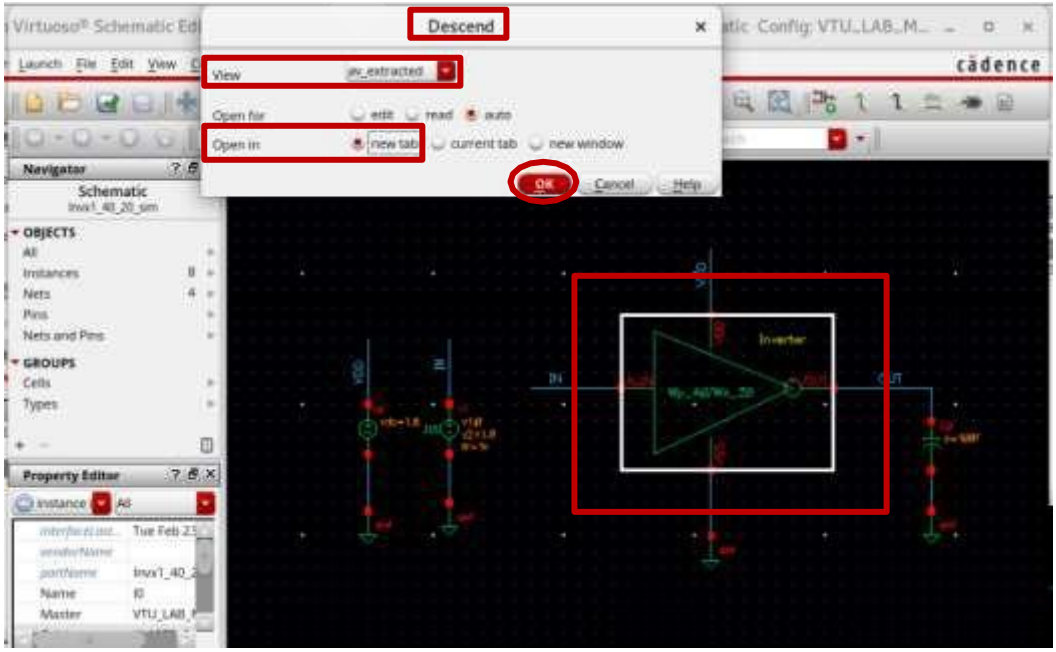


Figure – 1.151: Descend window

This should open the av_extracted view as we had seen in Figure -1.139 in a new tab as shown in Figure – 1.152.

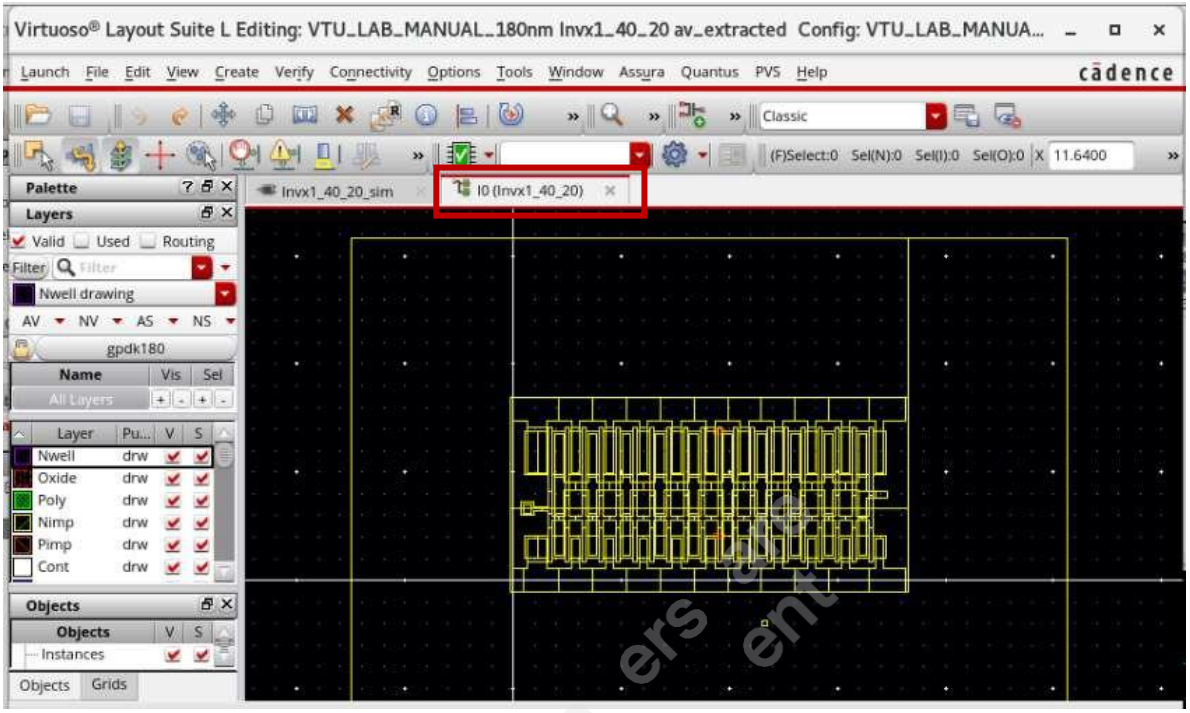


Figure – 1.152: av_extracted view in a new tab

Click on "**Launch □ ADE L**" and select "**Session □ Load State**" to open the Saved State as shown in Figure – 1.153.

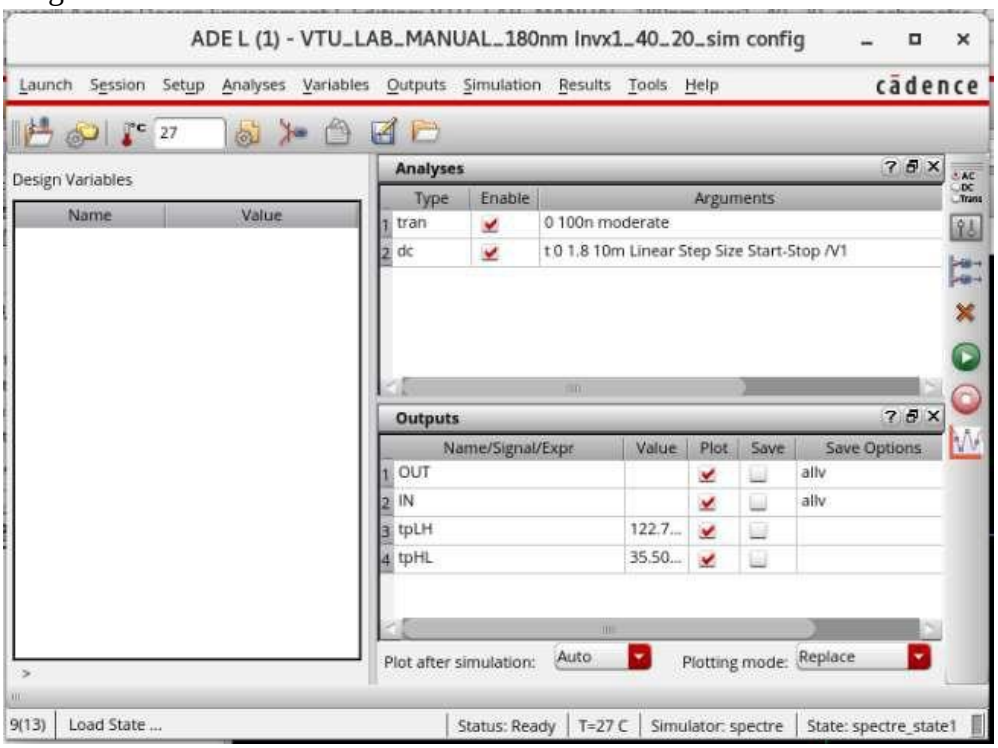


Figure – 1.153: Saved State

Re-run the Simulation and check for the waveforms of Transient Analysis and DC Analysis as shown in Figure – 1.154.

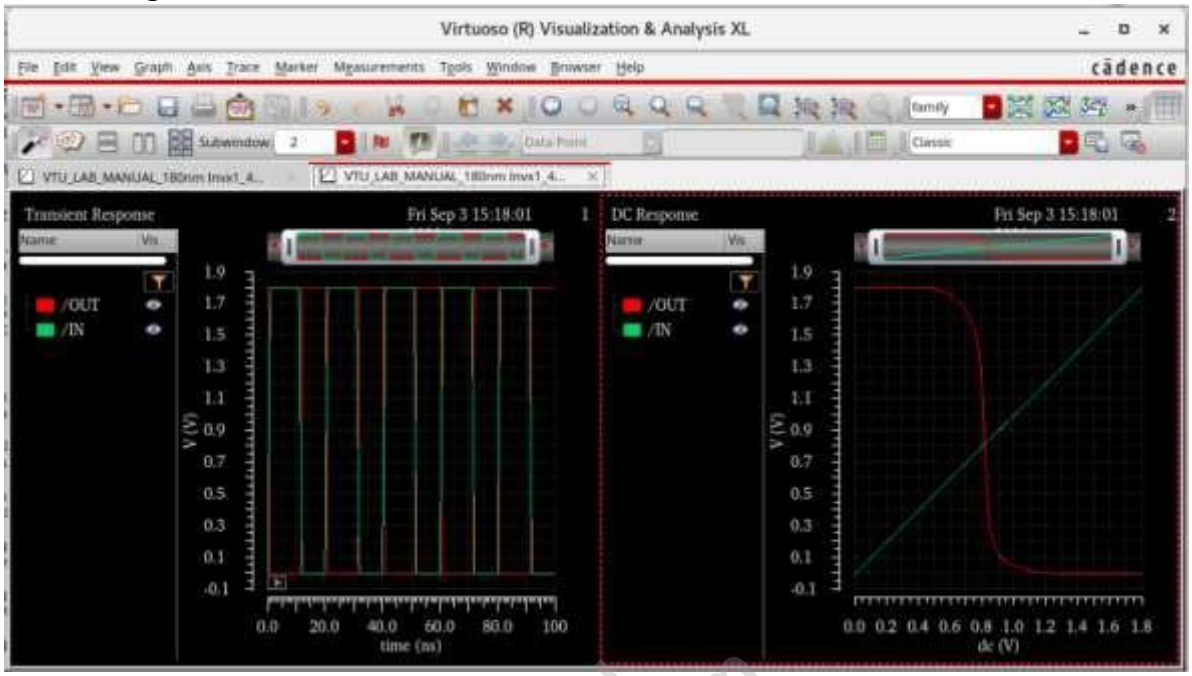


Figure – 1.154: Waveforms of Transient Analysis and DC Analysis

Using the Calculator, obtain the Switching Potential, t , tp_{LH} and t_{PD}. The results aretabulated in Table – 8.

Table – 8: Values of tp_{HL}, tp_{LH} and t_{PD} for CMOS Inverter with $\frac{W_P}{W_N} = \frac{40}{20}$

MOSFET	Length	Width	tpLH	tpHL	tpd
PMOS	180n	40u	1.23E-10	3.55E-11	7.78E-11
NMOS	180n	20u			

LAB – 02: 2 – INPUT CMOS NAND GATE

Objective:

(a) Capture the Schematic of a 2 – input CMOS NAND Gate having similar delay as that of CMOS Inverter computed in Lab – 01. Verify the functionality of the NAND Gate and also find out the delay for all the four possible combinations of input vectors. Tabulate the results. Increase the drive strength to 2X and 4X and tabulate the results.

(b) Draw the layout of NAND with $\frac{WP}{W_N} = \frac{40}{20}$, use optimum layout methods. Verify DRC and LVS, extract the parasitics and perform the post layout simulation, compare the results with pre layout simulations. Record the observations.

Solution – (a):

SCHEMATIC CAPTURE:

Following the techniques demonstrated in Lab – 01, Create a New Library using the option "File New Library", create a New Cell View upon selecting the newly created library using the option "File New Cell View" and instantiate the required devices using the "Create Instance" option.

The device parameters are listed in Table – 9.

Table – 9: Width and Length of NMOS and PMOS Transistors for CMOS NAND Gate

Library Name	Cell Name	Comments / Properties
gpdk180	Nmos	Width, W_N = 1.7 u Length, L = 180 n
gpdk180	Pmos	Width, W_P = 1.275 u Length, L = 180 n

Similarly, the device parameters for the 2 – input CMOS NAND Gate with drive strength 2 and drive strength 4 are listed in Table – 10 and Table – 11.

Table – 10: Width and Length of NMOS and PMOS Transistors for CMOS NAND Gatewith Drive Strength "2"

Library Name	Cell Name	Comments / Properties
gpdk180	Nmos	Width, W_N = 3.4 u Length, L = 180 n
gpdk180	Pmos	Width, W_P = 2.55 u Length, L = 180 n

Table – 11: Width and Length of NMOS and PMOS Transistors for CMOS NAND Gatewith Drive Strength "4"

Library Name	Cell Name	Comments / Properties
gpdk180	Nmos	Width, W_N = 6.8 u Length, L = 180 n
gpdk180	Pmos	Width, W_P = 5.1 u Length, L = 180 n

The completed Schematic for all the three dimensions are shown in Figure – 2.1, Figure – 2.2 and Figure – 2.3.

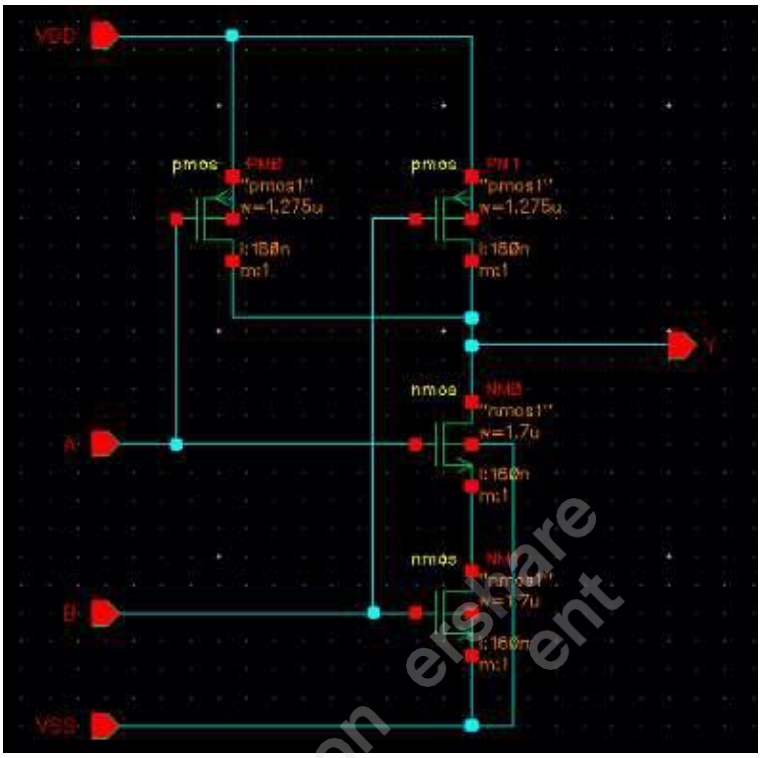


Figure – 2.1: Schematic Capture of 2 – input CMOS NAND Gate (NAND2X1)

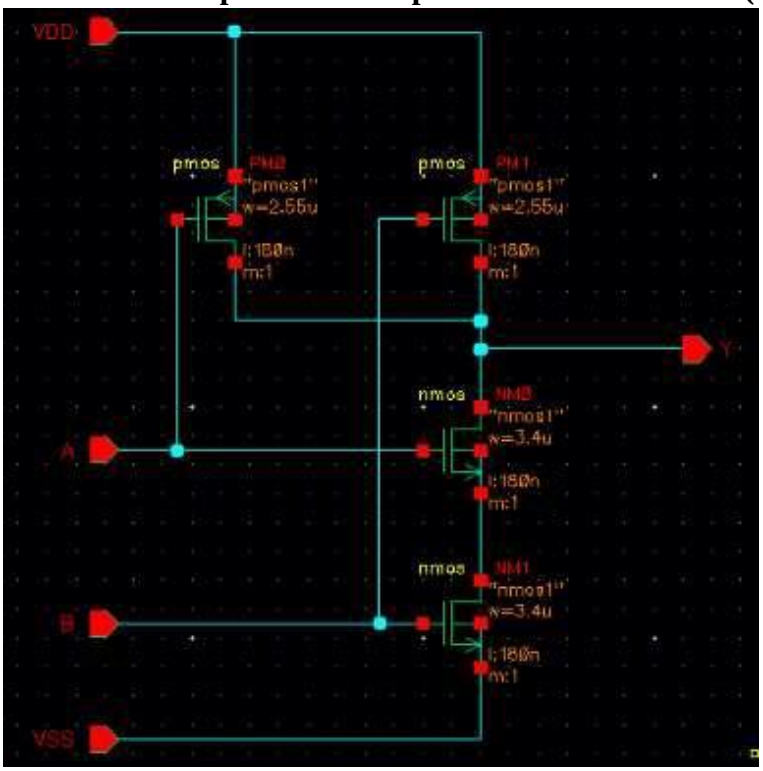


Figure – 2.2: Schematic Capture of 2 – input CMOS NAND Gate with drive strength 2 (NAND2X2)

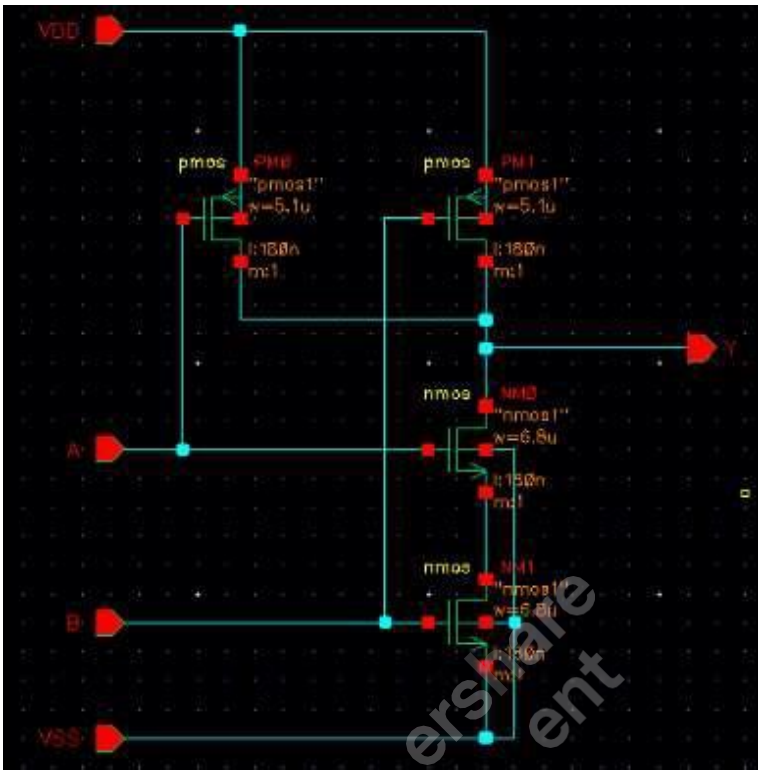


Figure – 2.3: Schematic Capture of 2 – input CMOS NAND Gate with drive strength 2 (NAND2X4)

The symbol for the CMOS NAND Gate is shown in Figure – 2.4.

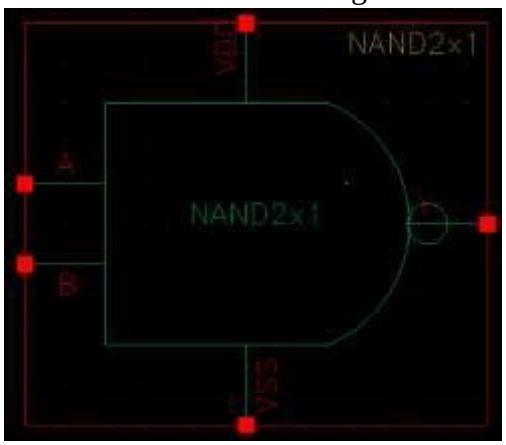
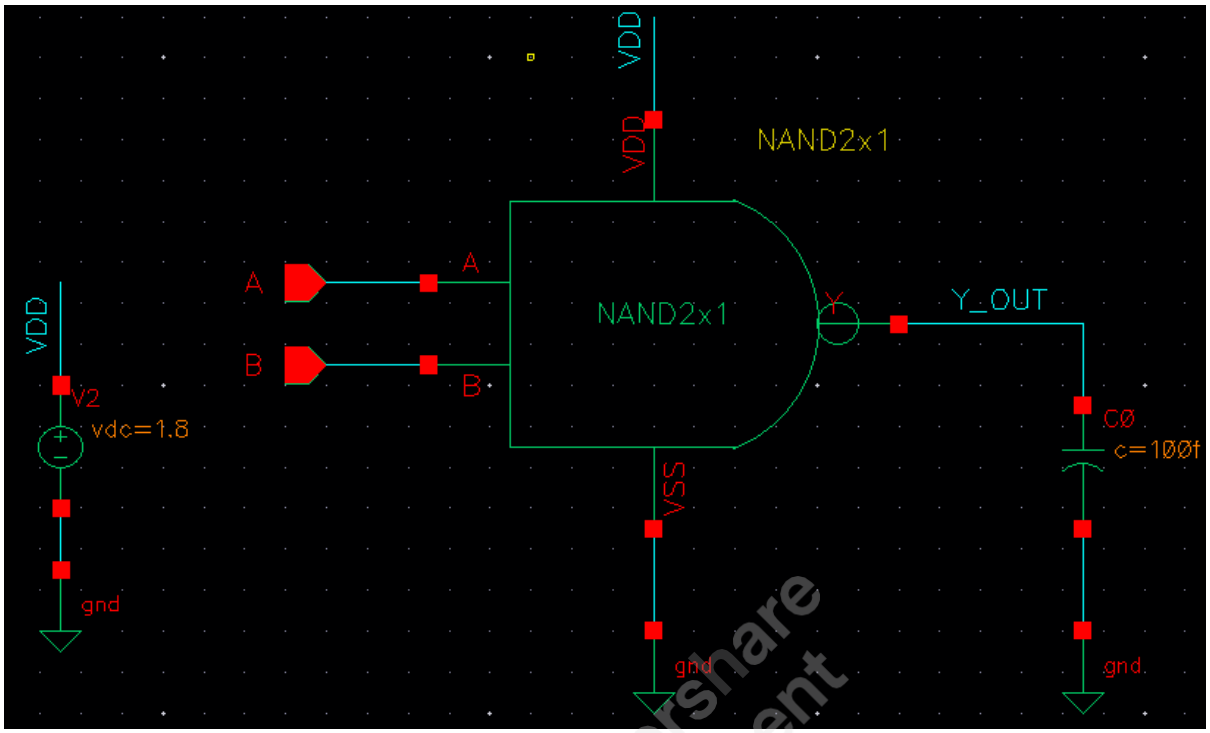


Figure – 2.4: Symbol of 2 – input NAND Gate

FUNCTIONAL SIMULATION:

Using the symbol created, build the Test Schematic. Create a New Cell View, instantiate the symbol of 2 – input NAND Gate, DC Voltage Source, Capacitance and Ground, connect the using wires. Create two input pins for the circuit A and B and connect them to the input of the NAND gate as shown in Figure – 2.5. Repeat the same procedure for creating the Test Schematic for the 2 – input CMOS NAND Gate with drive strength 2 and drive strength 4.



Launch ADE L, select "**Setup** □ **Stimuli**" as shown in Figure – 2.6 to give the required sequence of inputs to pins A and B.

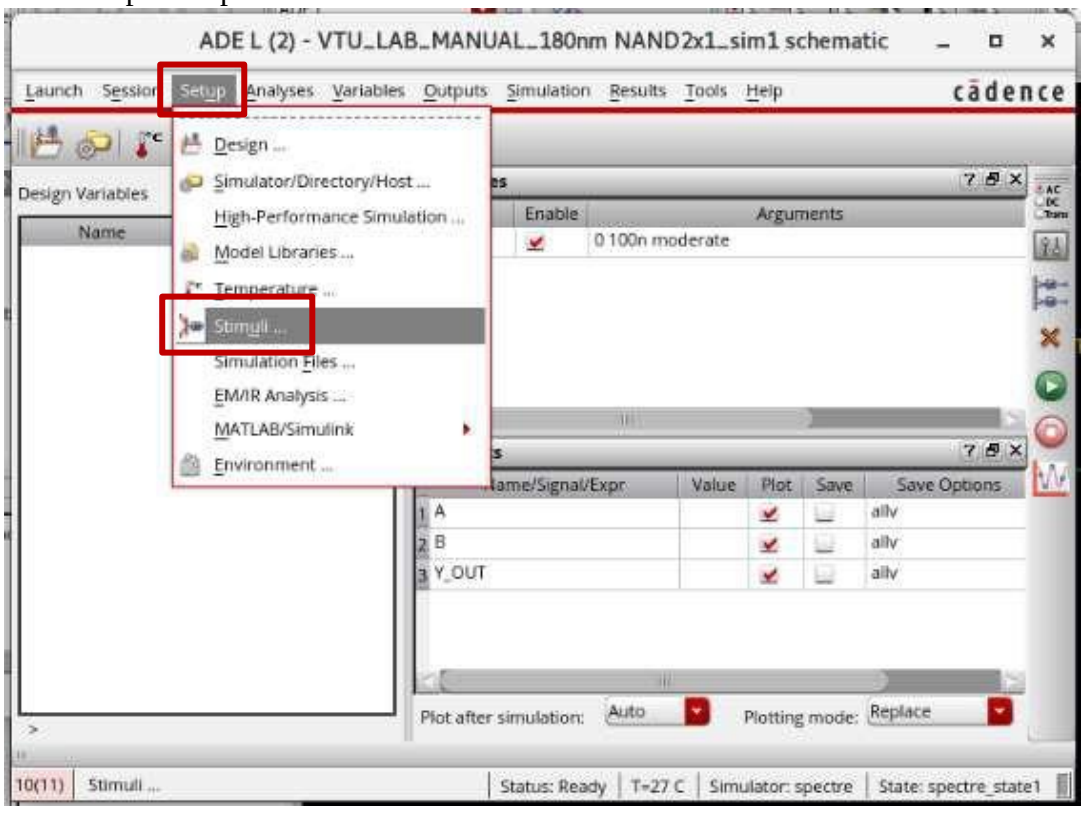


Figure – 2.6: Setup □ Stimuli

The "Setup Analog Stimuli" window pops up as shown in Figure – 2.7.

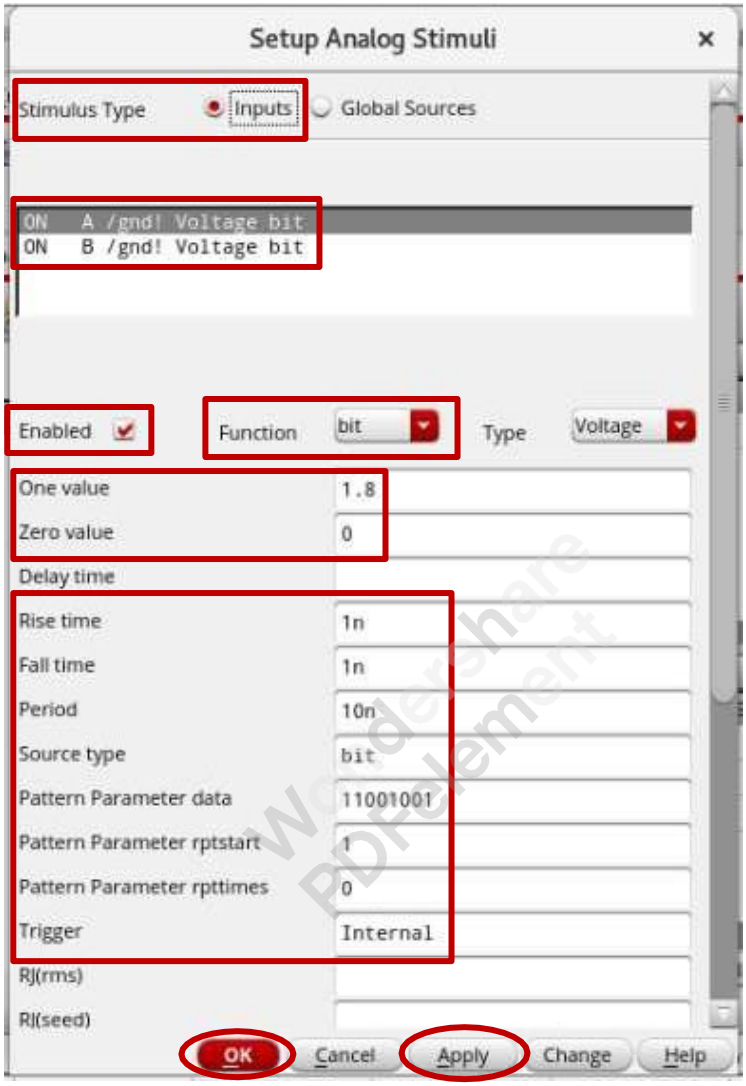


Figure – 2.7: Setup Analog Stimuli window

Select "**Stimulus Type □ Inputs**" and the input pins A and B get listed out as shown in Figure

– 2.7. Select any one of the Inputs, click on "**Enabled**" and select "**Function □ bit**". Mention the value of voltages for "**Logic 0**" and "**Logic 1**" in "**One value □ 1.8**" and "**Zerovalue □ 0**".
Consider the values of Rise time, Fall time and Period similar to that considered in Lab – 01. Select "**Source type □ bit**", "**Pattern Parameter data □ 11001001**", "**Pattern Parameter rptstart □ 1**", "**Pattern Parameter rpttimes □ 0**" and "**Trigger □ Internal**", click on "**Apply**" to "**Turn ON**" the input and click on "**OK**".

Select the type of Analysis to be performed on the 2 – input CMOS NAND Gate.
Select the Input and Output Signals to be plotted.
The ADE L window gets updated as shown in Figure – 2.8.
Run the Simulation to check for the functionality of the NAND Gate.

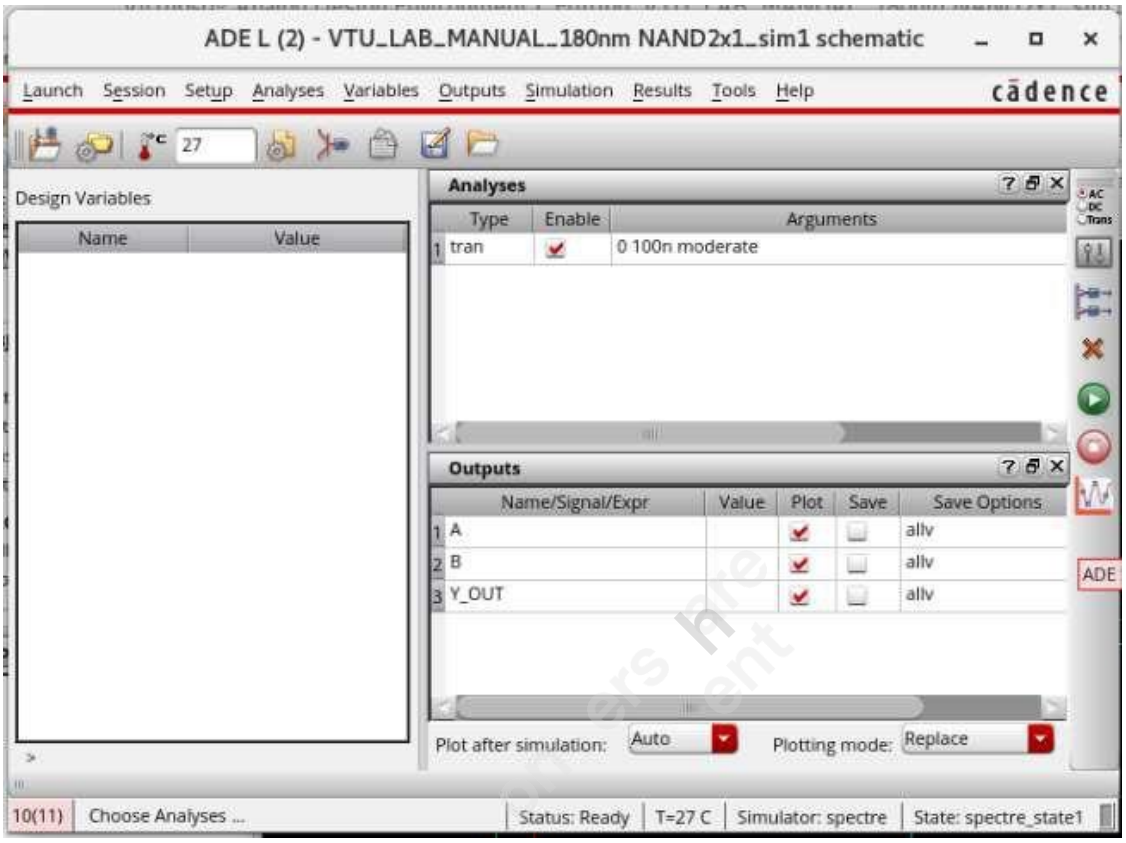


Figure – 2.8: Updated ADE L window

The Simulated waveforms can be seen as shown in Figure – 2.9.

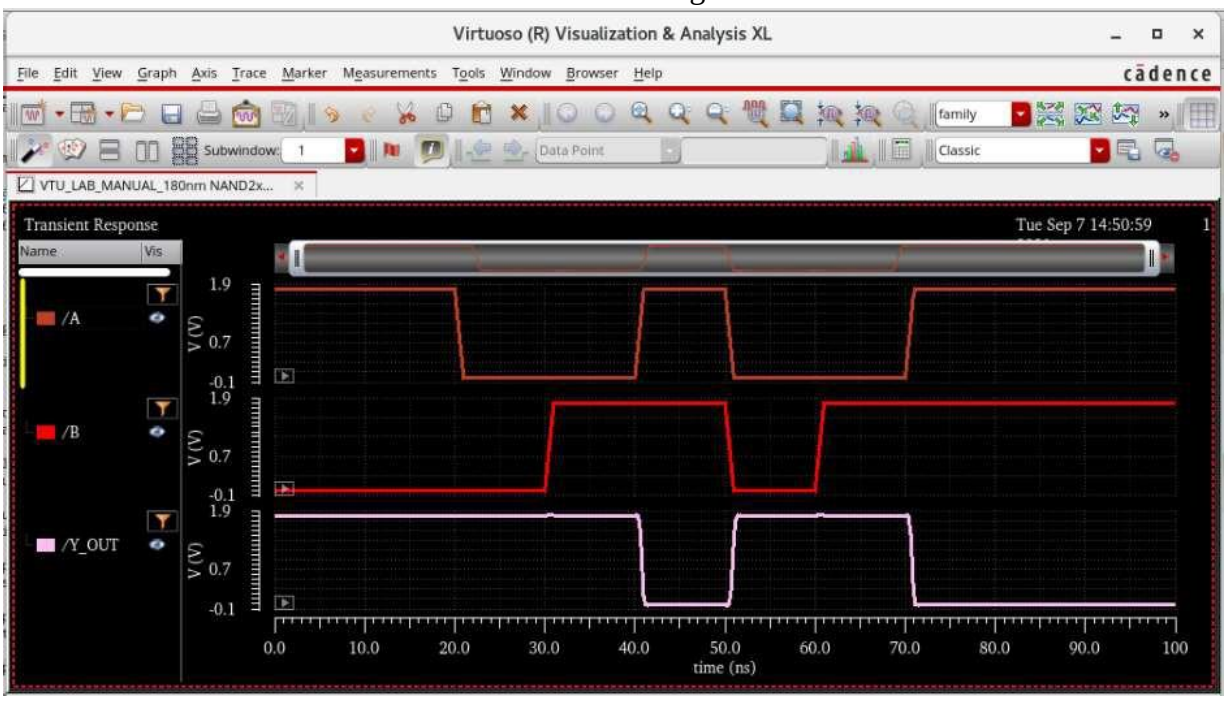


Figure – 2.9: Transient Analysis of 2 – input CMOS NAND Gate

The delay values are obtained using the "**Calculator**" option as demonstrated in Lab – 01. The results are tabulated as shown in Table – 12.

Table – 12: Values of Delay for 2 – input CMOS NAND2X1, NAND2X2 and NAND 2X4

NAND Type	MOSFET	Length	Width	tpLH	tpHL	tpd
NAND2X1	PMOS	180n	1.5 * 850n = 1.275u	3.720E-10	3.340E-10	3.530E-10
	NMOS		2 * 850n = 1.7u			
NAND2X2	PMOS	180n	1.5 * 850n * 2 = 2.55u	2.610E-10	2.200E-10	2.400E-10
	NMOS		2 * 850n * 2 = 3.4u			
NAND2X4	PMOS	180n	1.5 * 850n * 4 = 5.1u	1.900E-10	1.650E-10	1.780E-10
	NMOS		2 * 850n * 4 = 6.8u			

Solution – (b):

SCHEMATIC CAPTURE:

Following the techniques demonstrated in Lab – 01, Create a New Library, a New Cell View and instantiate the devices as per the Schematic of 2 – input CMOS NAND Gate.

The device parameters for the NMOS and PMOS Transistors are listed in Table – 13.

Table – 13: Device parameters for 2 – input CMOS NAND Gate with $\frac{W_P}{W_N} = \frac{40}{20}$

Library Name	Cell Name	Comments / Properties
gpdk180	Nmos	Width, W_N = 20 u Length, L = 180 n
gpdk180	Pmos	Width, W_P = 40 u Length, L = 180 n

The Schematic as per the dimensions of NMOS and PMOS transistors listed above is shown in Figure – 2.10.

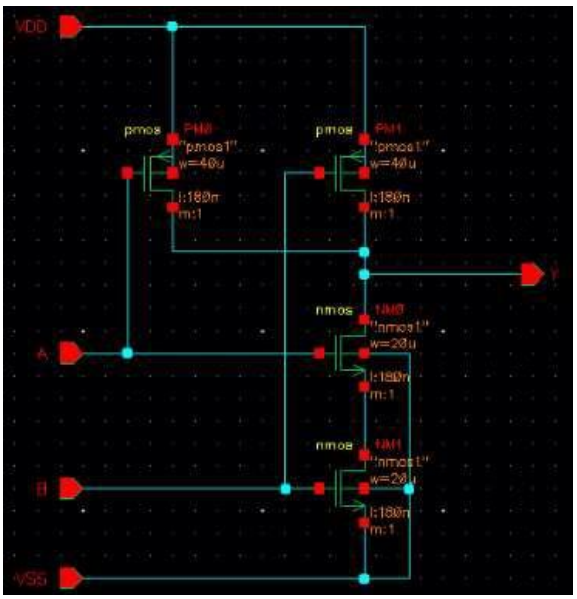


Figure – 2.10: Schematic for 2 – input CMOS NAND with $\frac{W_P}{W_N} = \frac{40}{20}$

Symbol for the Schematic in Figure – 2.10 is shown in Figure – 2.11.

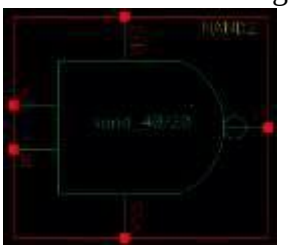


Figure – 2.11: Symbol for 2 – input CMOS NAND with $\frac{WP}{W_N} = \frac{40}{20}$

FUNCTIONAL SIMULATION:

The Test Schematic for the functionality check of the 2 – input CMOS NAND Gate is shown in Figure – 2.12.

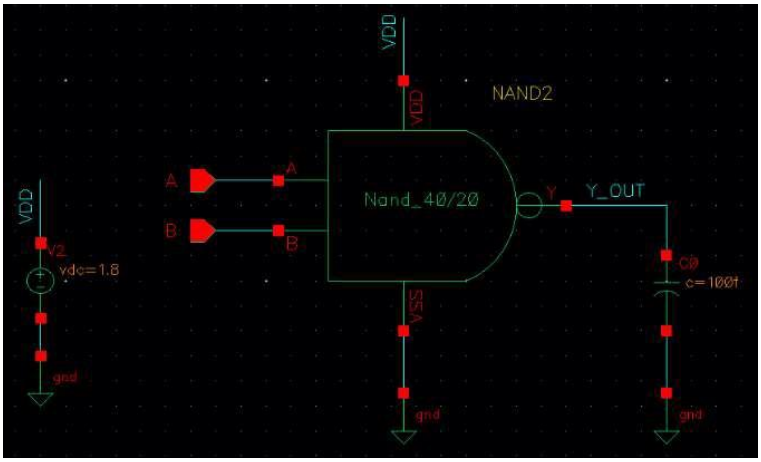


Figure – 2.12: Test Schematic for 2 – input CMOS NAND with $\frac{WP}{W_N} = \frac{40}{20}$

The ADE L window after choosing the Analysis and the Signals to be plotted is shown in Figure – 2.13.

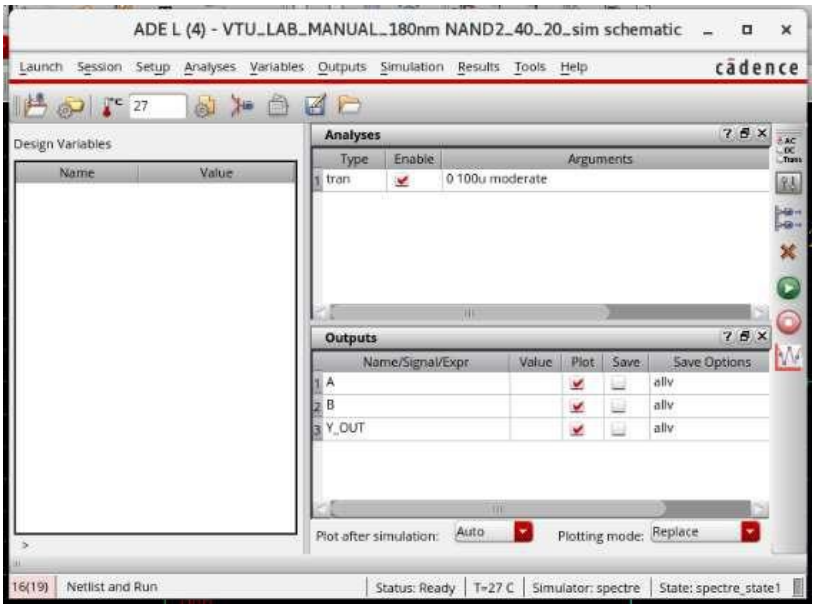


Figure – 2.13: Updated ADE L window

The waveforms after simulation is shown in Figure – 2.14.

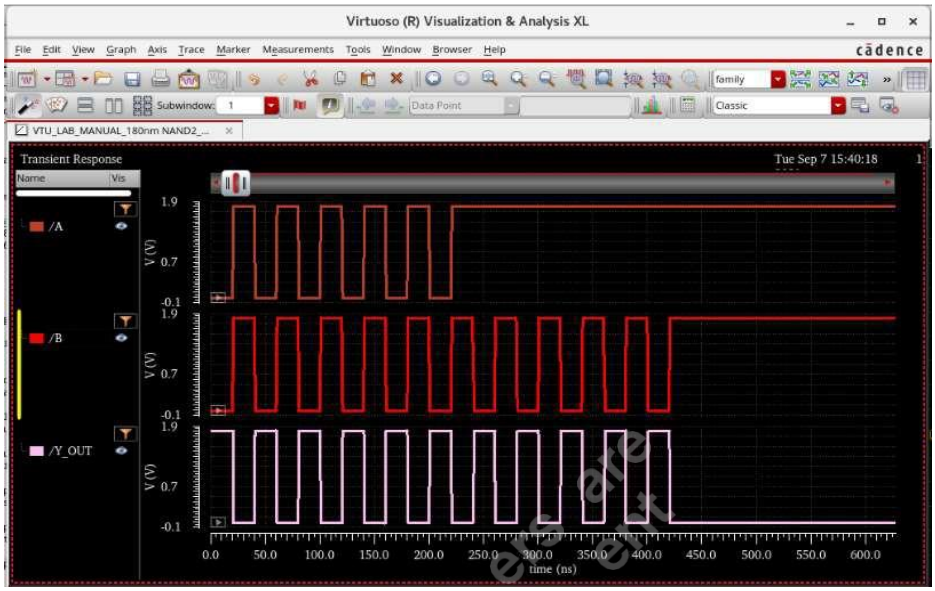


Figure – 2.14: Simulated Waveforms for 2 – input CMOS NAND with $\frac{WP}{W_N} = \frac{40}{20}$

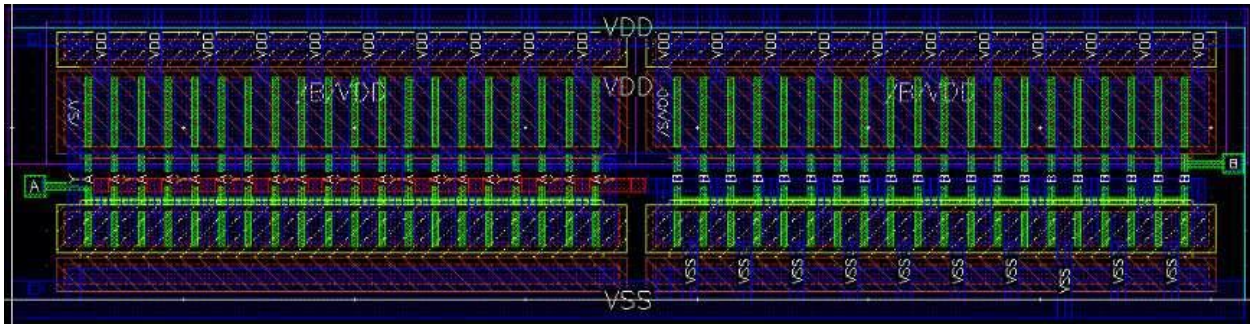
The values of delay elements are tabulated in Table – 14.

Table – 14: Delay Elements for 2 – input CMOS NAND Gate with $\frac{WP}{W_N} = \frac{40}{20}$ (Pre Layout Simulation)

MOSFET	Length	Width	tpLH	tpHL	tpd
PMOS	180n	40u	3.64E-11	1.55E-10	9.57E-11
NMOS	180n	20u			

LAYOUT:

Follow the techniques demonstrated in Lab – 01 to open the Layout Editor, import the devices from the Schematic, place the devices as per the requirement and complete the routing. The completed layout can be seen as shown in Figure – 2.15.



DRC:

To check for the DRC violations, browse the "**assura_tech.lib**" file, select "**Assura □ Run DRC**", verify the Layout Design Source, mention a "**Run Name**", select "**Technology □ gpdk180**" and click on "**OK**" as demonstrated in Lab – 01.

LVS:

To check for the LVS violations, select "**Assura □ Run LVS**", verify the Schematic Design Source and the Layout Design Source, mention a "**Run Name**", select "**Technology □ gpdk180**" and click on "**OK**" as demonstrated in Lab – 01.

QRC:

To extract the Parasitics, select "**Assura □ Quantus**", select "**Technology □ gpdk180**", "**Output □ Extracted View**" from the "**Setup**" option, select "**Extraction Type □ RC**" and "**Ref Node □ VSS**" from the "**Extraction**" and click on "**OK**" as demonstrated in Lab – 01. The result can be checked from the Library Manager.

BACKANNOTATION:

Import the parasitics into the Test Schematic and re-run the simulation to check their impact by calculating the delay elements as demonstrated in Lab – 01.

The values of delay are shown in Table – 15.

Table – 15: Delay Elements for 2 – input CMOS NAND Gate with $\frac{W_P}{W_N} = \frac{40}{20}$ (Post Layout Simulation)

MOSFET	Length	Width	tpLH	tpHL	tpd
PMOS	180n	40u	3.64E-11	1.55E-10	9.57E-11
NMOS	180n	20u			

LAB – 03: COMMON SOURCE AMPLIFIER WITH PMOS CURRENT MIRRORLOAD

Objective:

(a) Capture the Schematic of a Common Source Amplifier with PMOS Current Mirror Load and find its Transient Response and AC Response. Measure the UGB and Amplification Factor by varying transistor geometries, study the impact of variation in width to UGB.

(b) Draw the layout of Common Source Amplifier, use optimum layout methods. Verify DRC and LVS, extract the parasitics and perform the post layout simulation, compare the results with pre layout simulations. Record the observations.

Solution – (a):

SCHEMATIC CAPTURE:

Following the techniques demonstrated in Lab – 01, Create a New Library using the option "File New Library", create a New Cell View upon selecting the newly created library using the option "File New Cell View" and instantiate the required devices using the "Create Instance" option.

The device parameters are listed in Table – 16.

Table – 16: Width and Length of NMOS and PMOS Transistors

Library Name	Cell Name	Comments / Properties
gpdk180	Nmos	Width, W_N = 6 u Length, L = 180 n
gpdk180	Pmos	Width, W_P = 8.85 u Length, L = 180 n

The completed Schematic is shown in Figure – 3.1.

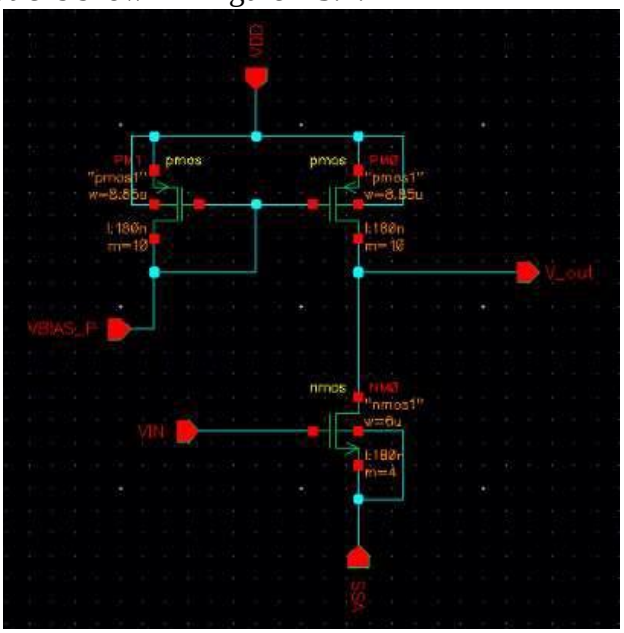


Figure – 3.1: Schematic of Common Source Amplifier with PMOS Current Mirror Load

The symbol for the Common Source Amplifier with PMOS Current Mirror Load is shown in Figure – 3.2.

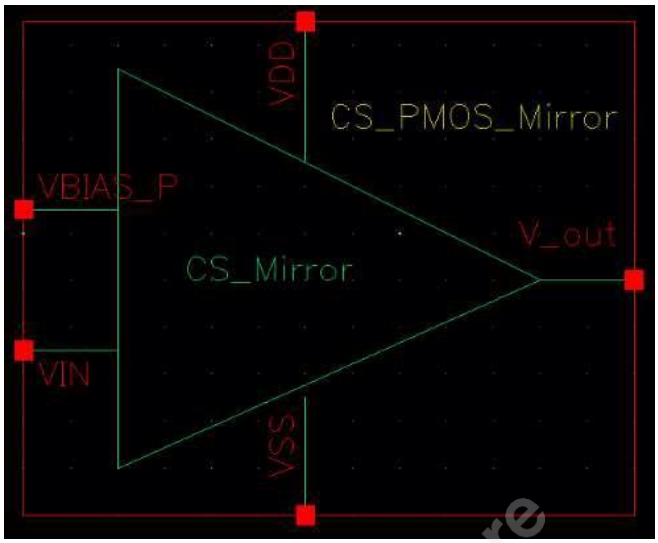


Figure – 3.2: Symbol of Common Source Amplifier with PMOS Current Mirror Load

FUNCTIONAL SIMULATION:

Using the symbol created, build the Test Schematic. Create a New Cell View, instantiate the symbol of Common Source Amplifier with PMOS Current Mirror Load, DC Voltage Source, Current Source, AC Voltage Source, Capacitance, Resistance and Ground, connect the using wires.

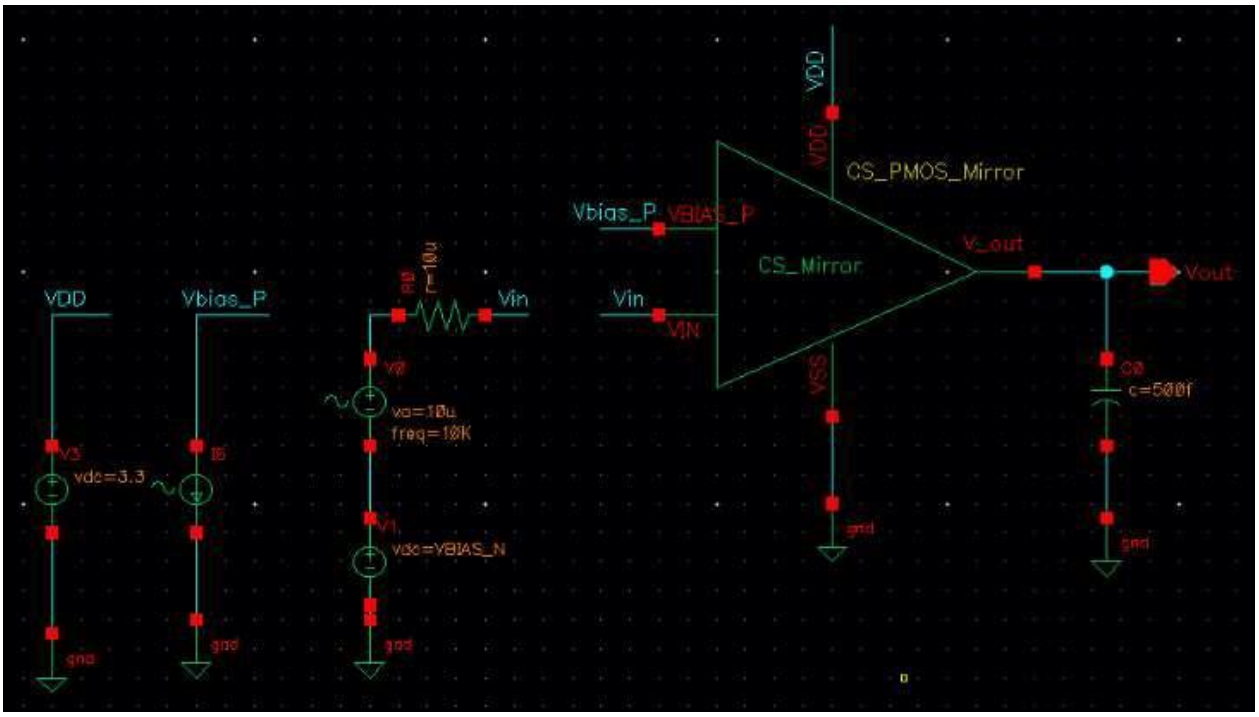


Figure – 3.3: Test Schematic for 2 – input CMOS NAND Gate

The parameters for remaining devices are shown in Table – 17.

Library Name	Cell Name	Comments / Properties
analogLib	vdc	DC voltage = 3.3 V (VDD)
analogLib	vdc	DC voltage = VBIAS_N V (Vin)
analogLib	isin	DC current = 100u A (Vbias_P)
analogLib	vsin	AC Magnitude = 1 V, Amplitude = 10u V, Frequency = 10K Hz (Vin)
analogLib	cap	Capacitance = 500f F
analogLib	res	Resistance = 10u Ohms
analogLib	gnd	

Table – 17: Parameters for the devices used in Test Schematic

Launch ADE L, import the design variables, mention the values and select the Transient Analysis, DC Analysis and AC Analysis, mention the parameters and choose the signals to be plotted as shown in Figure – 3.4.

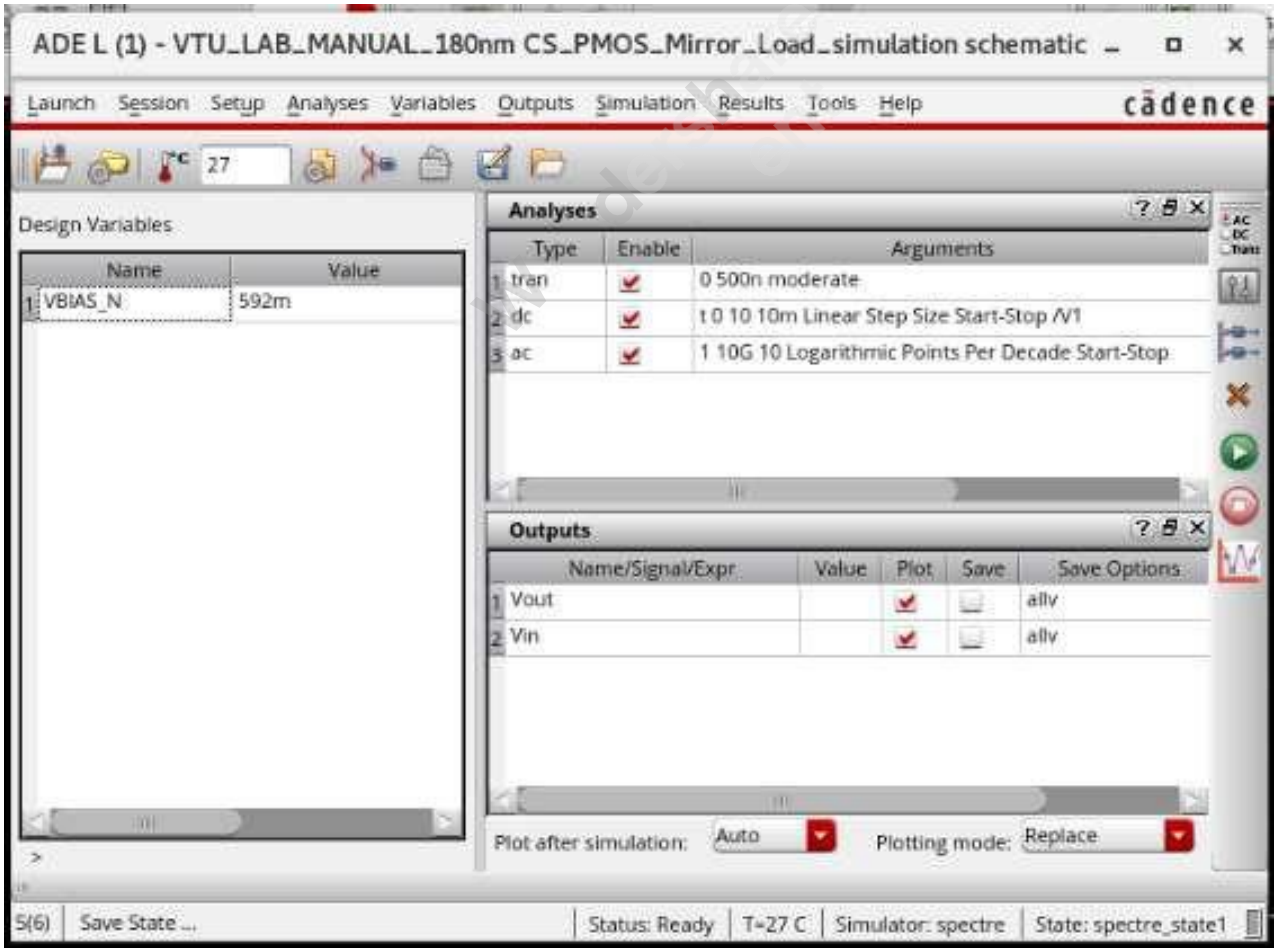


Figure – 3.4: Updated ADE L window

The Simulated waveforms can be seen as shown in Figure – 3.5 and Figure – 3.6.

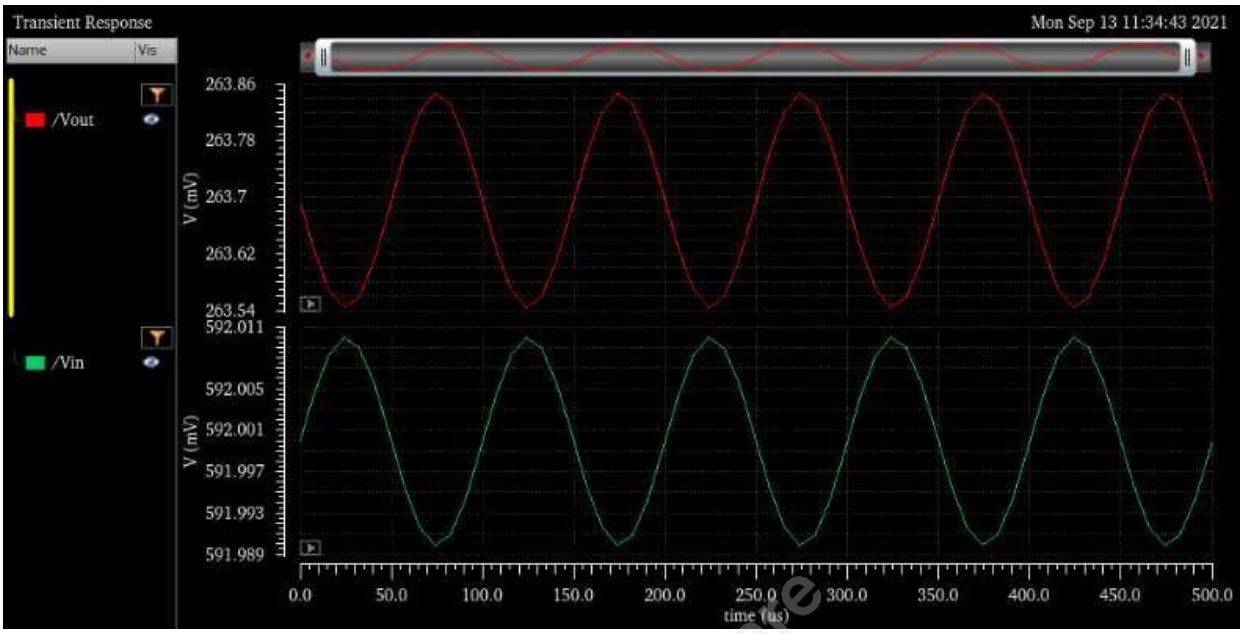


Figure – 3.5: Transient Analysis

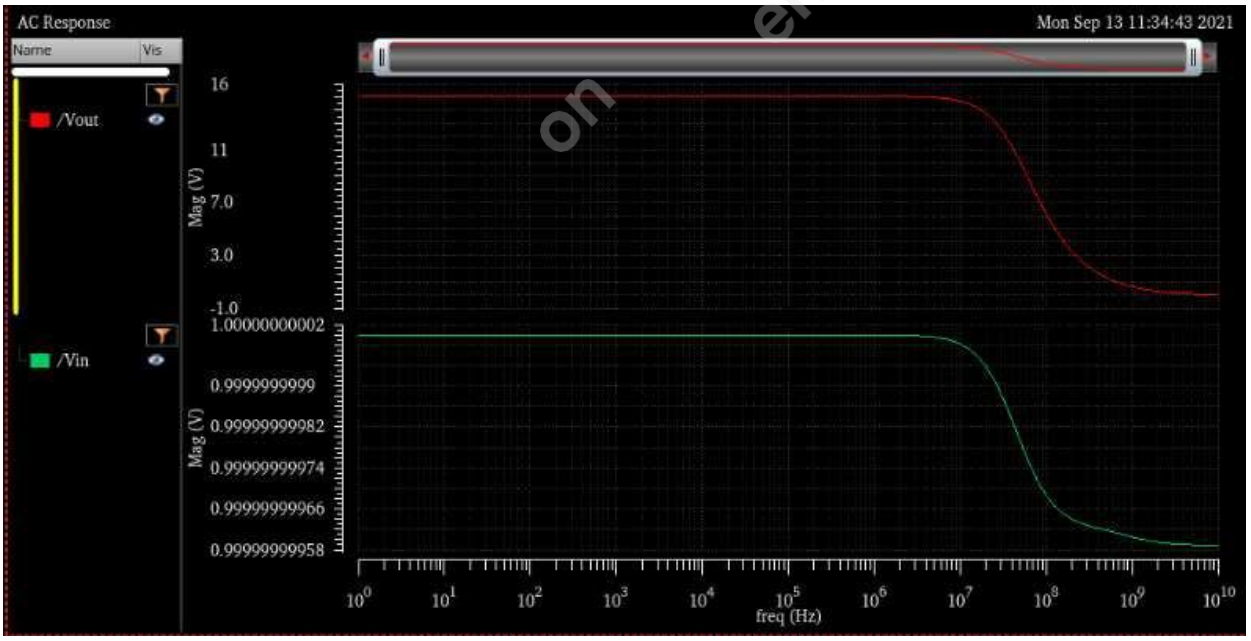
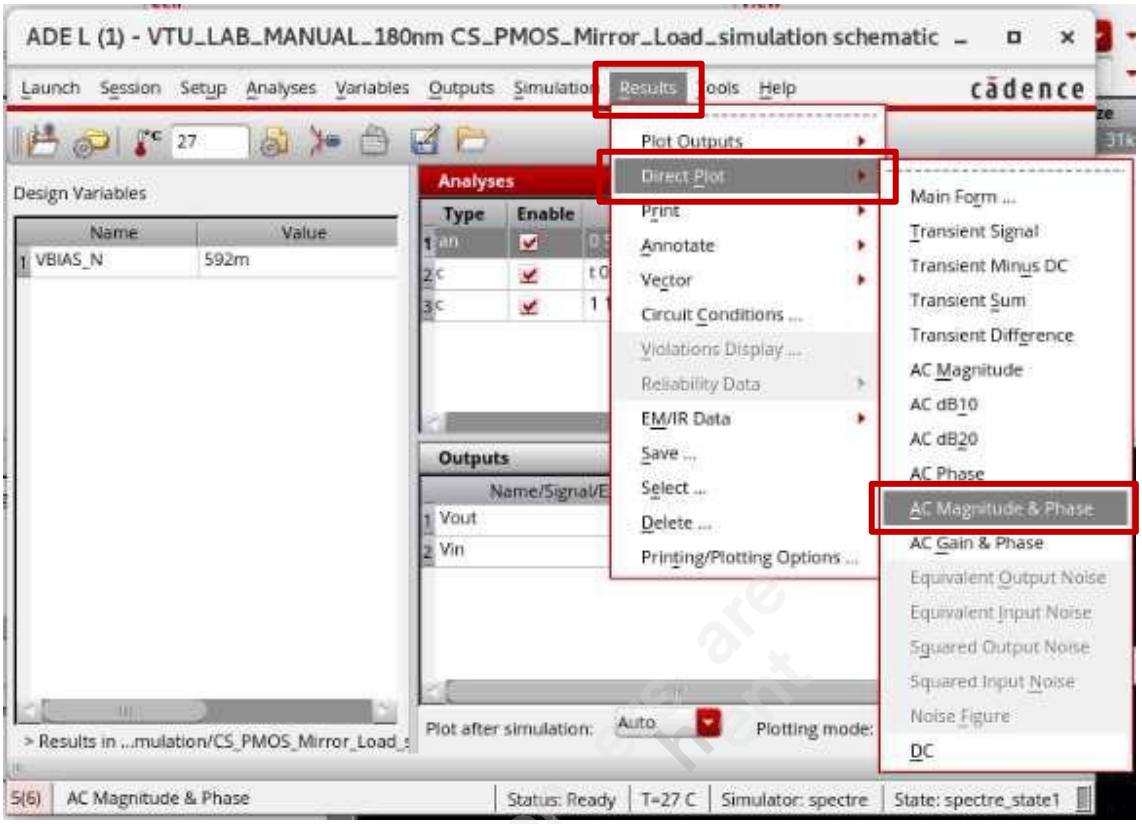


Figure – 3.6: AC Analysis

To measure the Gain and Unity Gain Bandwidth, go back to the ADE L window, select "**Results □ Direct Plot □ AC Magnitude & Phase**" as shown in Figure – 3.7.

The Test Schematic window pops up, select the output net as shown in Figure – 3.8 and click on "**Esc**" key on the keyboard.
The waveform can be seen as shown in Figure – 3.9. The marker placed on the low frequency part of the response gives the DC Gain, use the bind key "**M**" to place the marker.
Place a horizontal cursor at "**0 dB**" and the crossing frequency gives the Unity Gain Bandwidth (UGB) as shown in Figure 3.9.



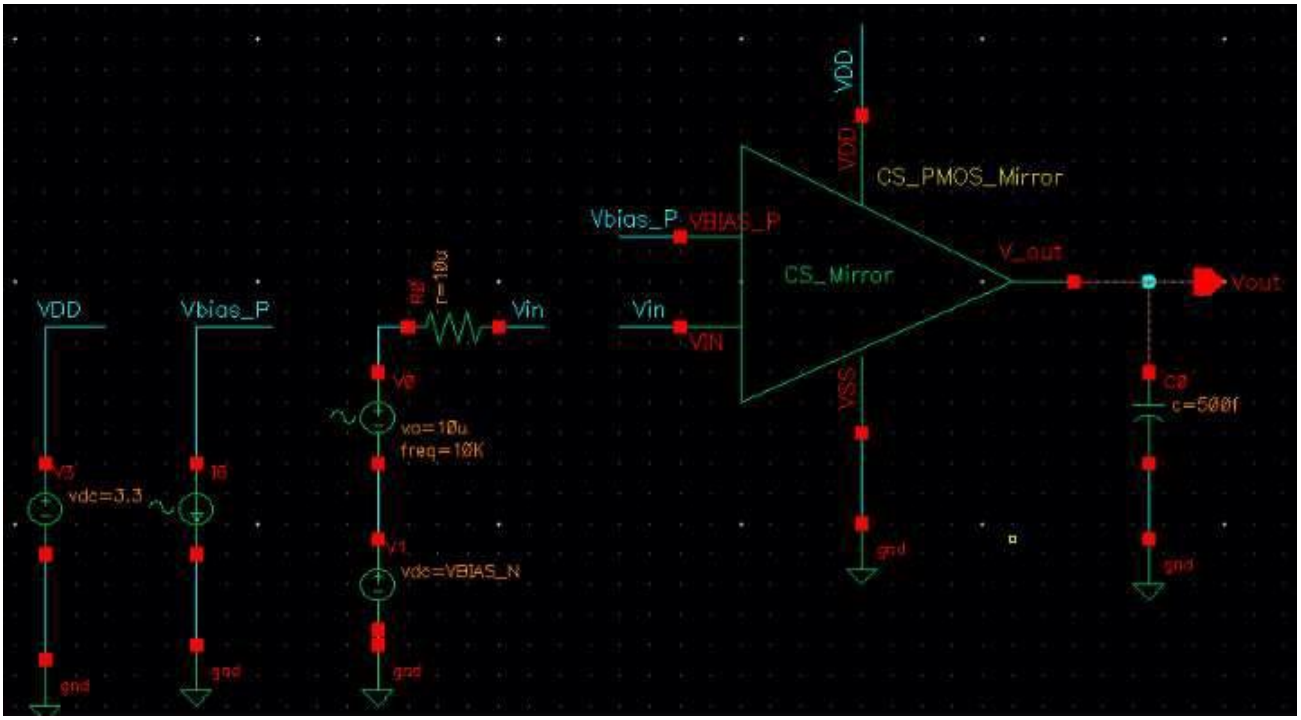


Figure – 3.8: Selecting Output Net from the Test Schematic

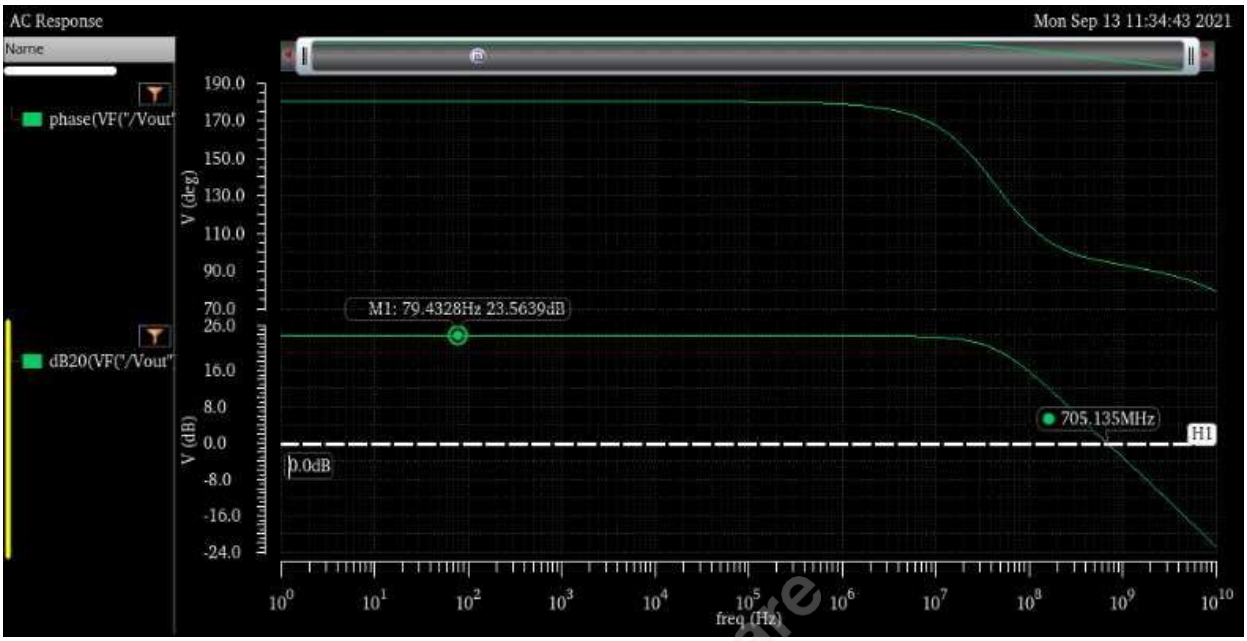


Figure – 3.9: Gain and Phase plot

Solution – (b):

LAYOUT:

Follow the techniques demonstrated in Lab – 01 to open the Layout Editor, import the devices from the Schematic, place the devices as per the requirement and complete the routing. The completed layout can be seen as shown in Figure – 3.10.

DRC:

To check for the DRC violations, browse the **Assura tech lib**. Then select **Assura → Run DRC**, verify the Layout Design Source, mention a **Run Name**, select **Technology gpdk180** and click on **OK** as demonstrated in Lab –01.

LVS:

To check for the LVS violations, select **Assura → Run LVS**, verify the Schematic Design Source, mention a Run Name, select Technology gpdk180 and click on **OK** as demonstrated in Lab –01.

Q

To extract the parasitics, select **Assura → Run Parasitic Extraction**, select **Extracted View**, mention a Run Name, select Technology gpdk180 and click on **OK** as demonstrated in Lab –01.

The result can be checked from the Library Manager.

BACKANNOTATION:

Import the parasitics into the Test Schematic and re-run the simulation to check their impact by calculating the delay elements as demonstrated in Lab – 01.

LAB – 04: 2 – STAGE OPERATIONAL AMPLIFIER

Objective:

(a) Capture the Schematic of a 2 – Stage Operational Amplifier and measure the following:

1. UGB
2. dB Bandwidth
3. Gain Margin and Phase Margin with and without coupling capacitance
4. Use the Op-Amp in the Inverting and Non-Inverting configuration and verify its functionality
5. Study the UGB, 3 dB Bandwidth, Gain and Power Requirement in Op-Amp by varying the stage wise transistor geometries and record the observations

(b) Draw the layout of 2 – stage Operational Amplifier with the maximum transistor width set to 300 (in 180 / 90/ 45n m Technology), choose appropriate transistor geometries as per the results obtained in 4(a). Use optimum layout methods. Verify DRC and LVS, extract the parasitics and perform the post layout simulation, compare the results with pre layout simulations. Record the observations.

Solution – (a):

SCHEMATIC CAPTURE:

Create a New Library, select the Technology Node as "**gpdk045**" (Technology Node used for this demonstration is 45 nm), Create a New Cell View, instantiate the devices as demonstrated in Lab – 01. Use the "**Sideways**" option as shown in Figure – 4.1 to flip the Transistor.

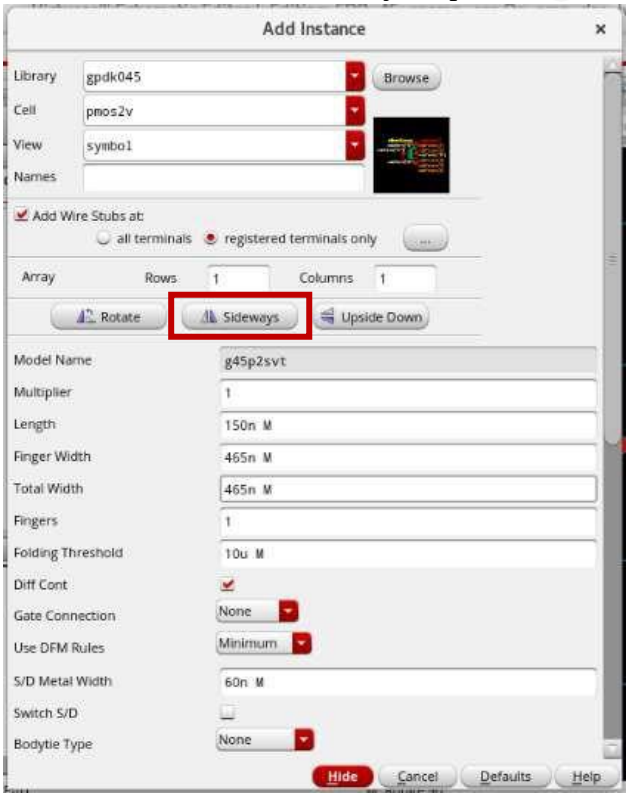


Figure – 4.1: "Sideways" option to flip the Transistors "Sideways"

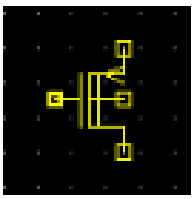


Figure – 4.1(a): Before and After selecting

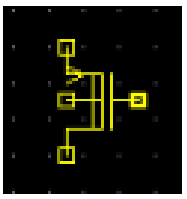


Figure – 4.1(b):

The Transistors before and after flipping are shown in Figure – 4.1(a) and Figure – 4.1(b). The dimensions of all the devices are given in Table – 18 as shown below.

Table – 18: Device Parameters for 2 – Stage Operational Amplifier

Library Name	Transistor	Cell Name	Comments / Properties
gpdk045	M0, M1	pmos2v	Width, W = 465 n Length, L = 150 n
gpdk045	M3, M4	nmos2v	Width, W = 490 n Length, L = 150 n
gpdk045	M5, M7	nmos2v	Width, W = 1.09 u Length, L = 150 n
gpdk045	M2	pmos2v	Width, W = 10 u Length, L = 150 n
gpdk045	M6	nmos2v	Width, W = 6.88 u Length, L = 150 n
gpdk045	M8	pmoscap2v	Calculated Parameter = Capacitance Capacitance = 250.043 f

The completed Schematic as per the dimensions mentioned in Table – 18 is shown in Figure – 4.2.

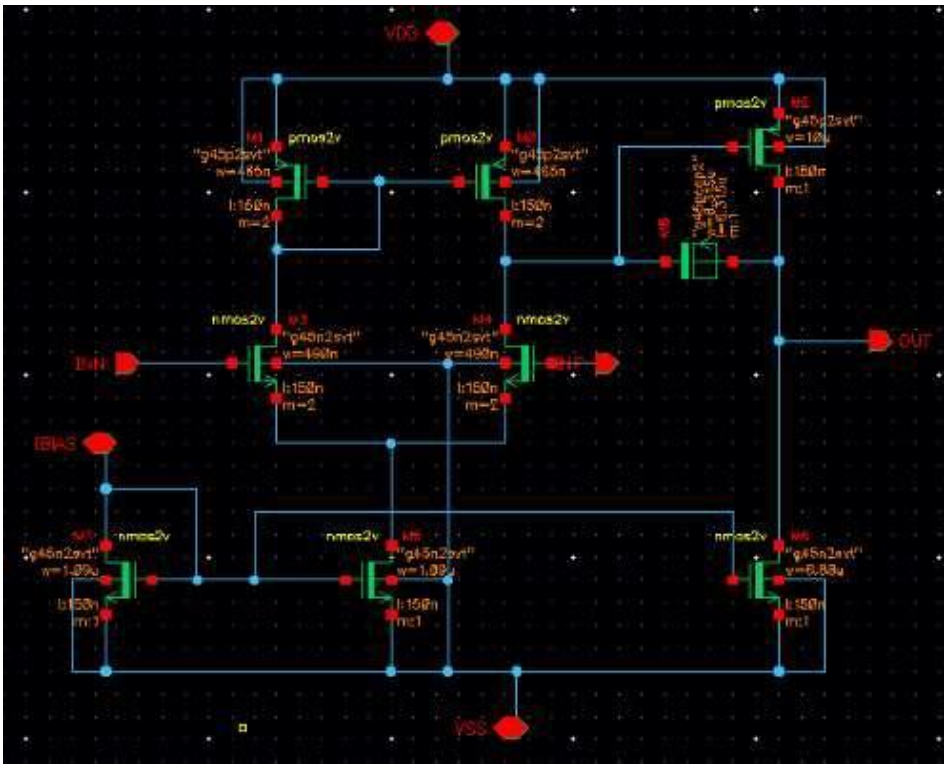


Figure – 4.2: Schematic of 2 – Stage Operational Amplifier

The Symbol created according to the Techniques demonstrated in Lab – 01 is shown in Figure – 4.3.

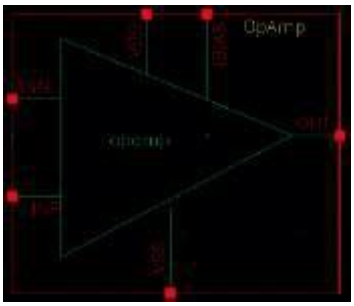


Figure – 4.3: Symbol for 2 – Stage Operational Amplifier

FUNCTIONAL SIMULATION USING ADE EXPLORER AND ASSEMBLER:

To test the functionality of Operational Amplifier, build a Test Schematic using the Symbol that was created as shown in Figure – 4.3. Create a New Cell View, instantiate the symbol. Instantiate the other devices required for testing the circuit, mention the device parameters as shown in Table – 19.

Table – 19: Device Parameters for 2 – Stage Operational Amplifier Test Schematic

Library Name	Cell Name	Comments / Properties
analogLib	vdc	DC voltage = vdd V
analogLib	vdc	DC voltage = vss V
analogLib	vpulse	Voltage 1 = vdc + 0.3 V, Voltage 2 = vdc - 0.3 V, Period = 10u s, Rise time = 10p s, Fall time = 10p s
analogLib	idc	DC current = ibias A
analogLib	cap	Capacitance = CL F
analogLib	gnd	

The Test Schematic after completion of all the interconnections can be seen as shown in Figure – 4.4.

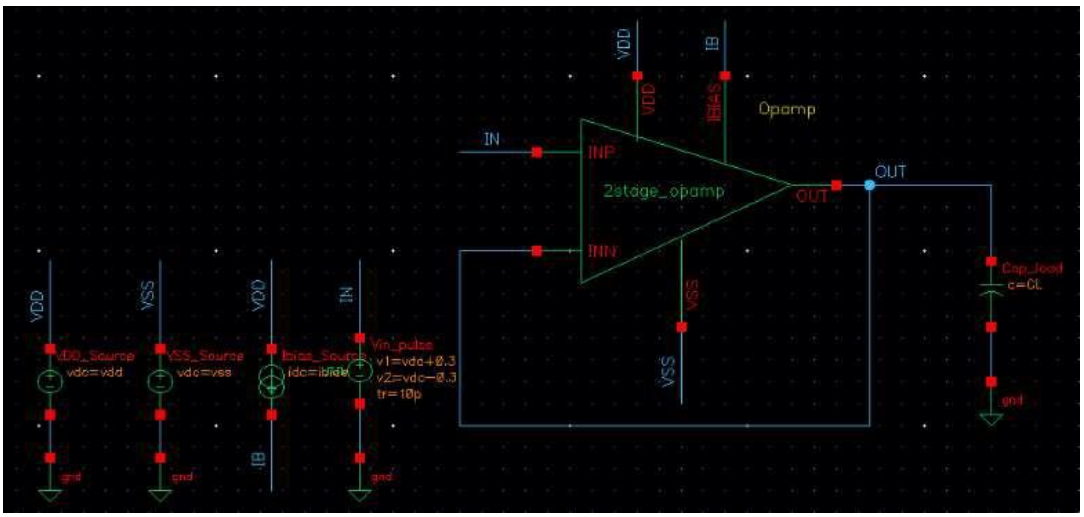


Figure – 4.4: Test Schematic for 2 – Stage Operational Amplifier

The specification that has to be achieved on simulating the design are as follows:

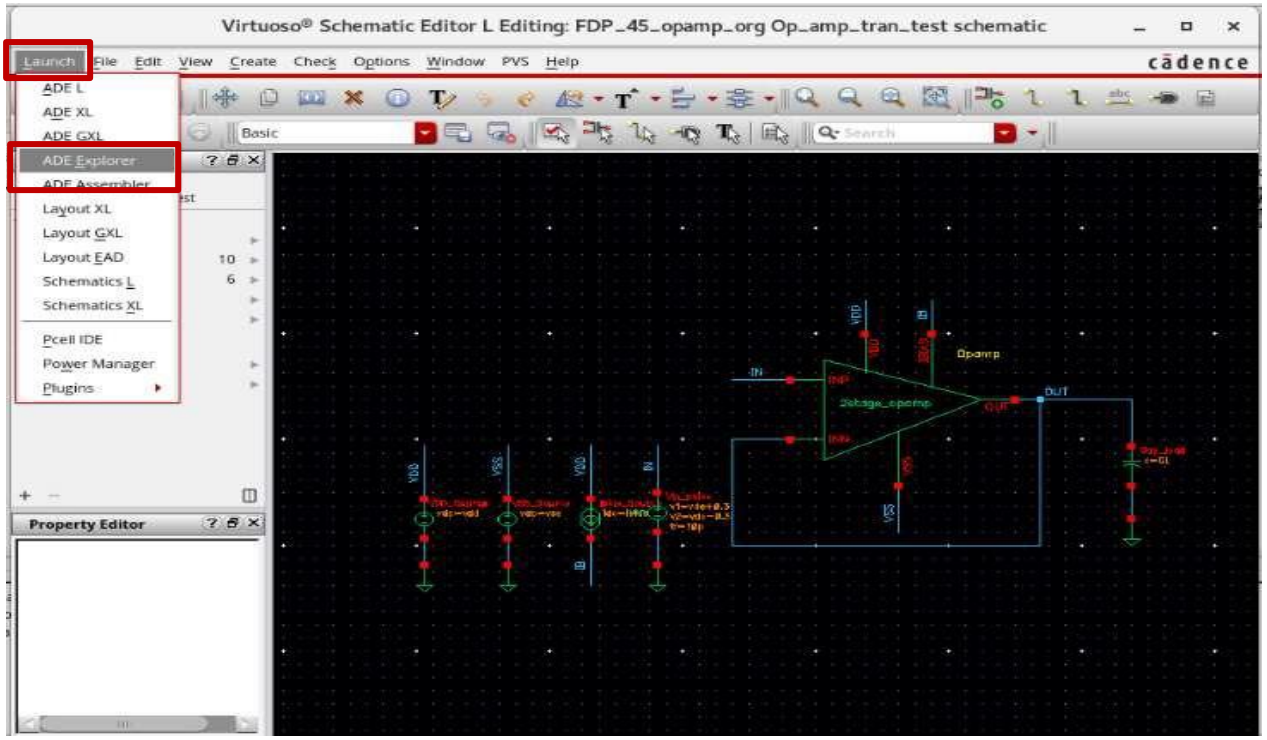
- Slew Rate >= 50 MV/s
- DC Open Loop Gain >= 60 dB (1000 V/V)
- Unity Gain Bandwidth >= 50 MHz
- Output Offset <= ± 10 mV
- Settling Time <= 50 ns

The steps to be carried out are listed below:

Step – 1:

Select "**Launch □ ADE Explorer**" as shown in Figure – 4.5

Figure – 4.5: Launch □ ADE Explorer



The "**Launch ADE Explorer**" window pops up, select "**Create New View**" and click on "**OK**" as shown in Figure – 4.6.

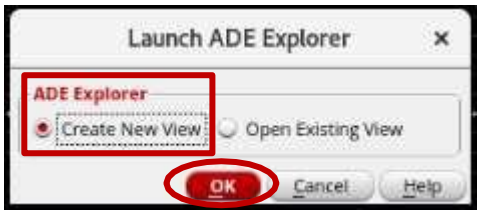


Figure – 4.6: "Launch ADE Explorer" window

The "**Create new ADE Explorer view**" window pops up as shown in Figure – 4.7. Select the

Cell View Name, "**Open in □ new tab**" and click on "**OK**".

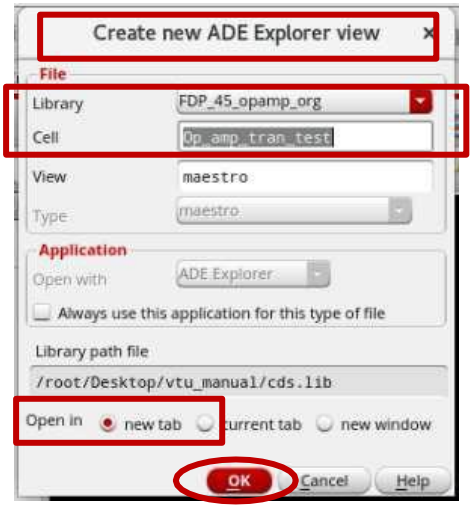


Figure – 4.7: "Create new ADE Explorer view" window

The "**Virtuoso ADE Explorer Editing**" window pops up as shown in Figure – 4.8.

Figure – 4.8: "Virtuoso ADE Explorer Editing" window

Select "**Setup □ Model Libraries**" as shown in Figure – 4.9.

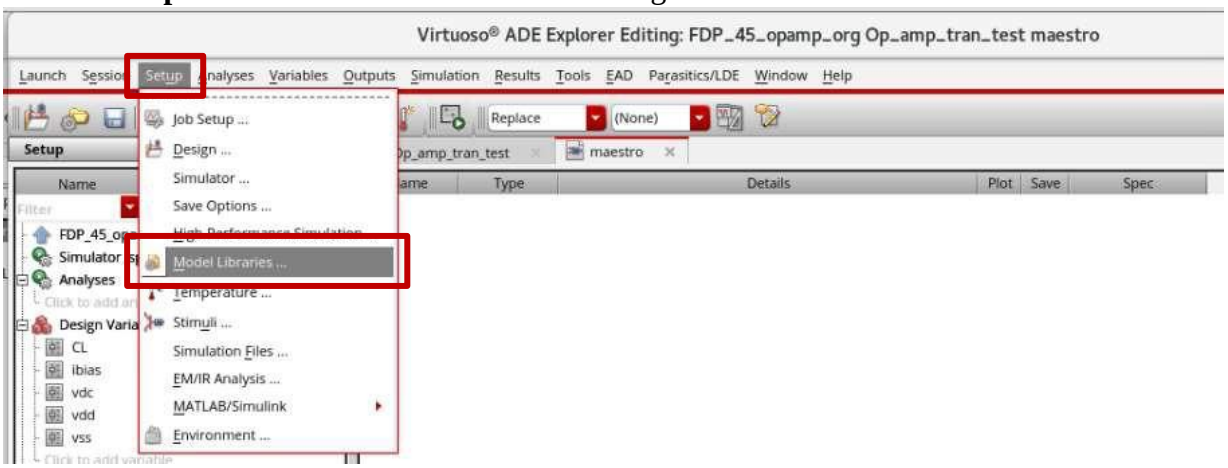


Figure – 4.9: Setup □ Model Libraries

The "**spectre1: Model Library Setup**" window pops up as shown in Figure – 4.10. Select the

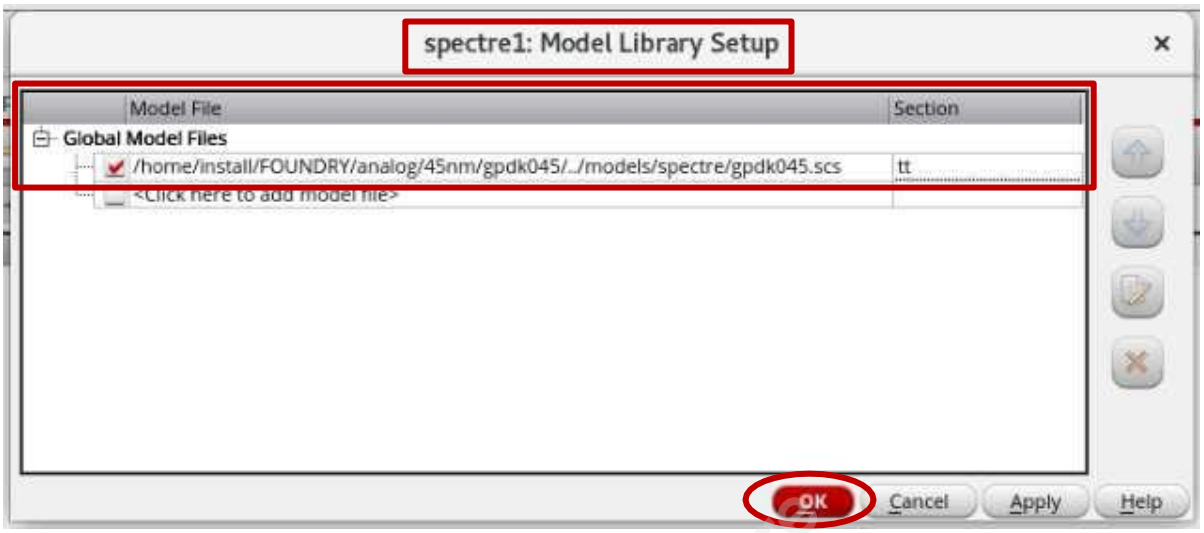


Figure – 4.10: "spectre1: Model Library Setup" window

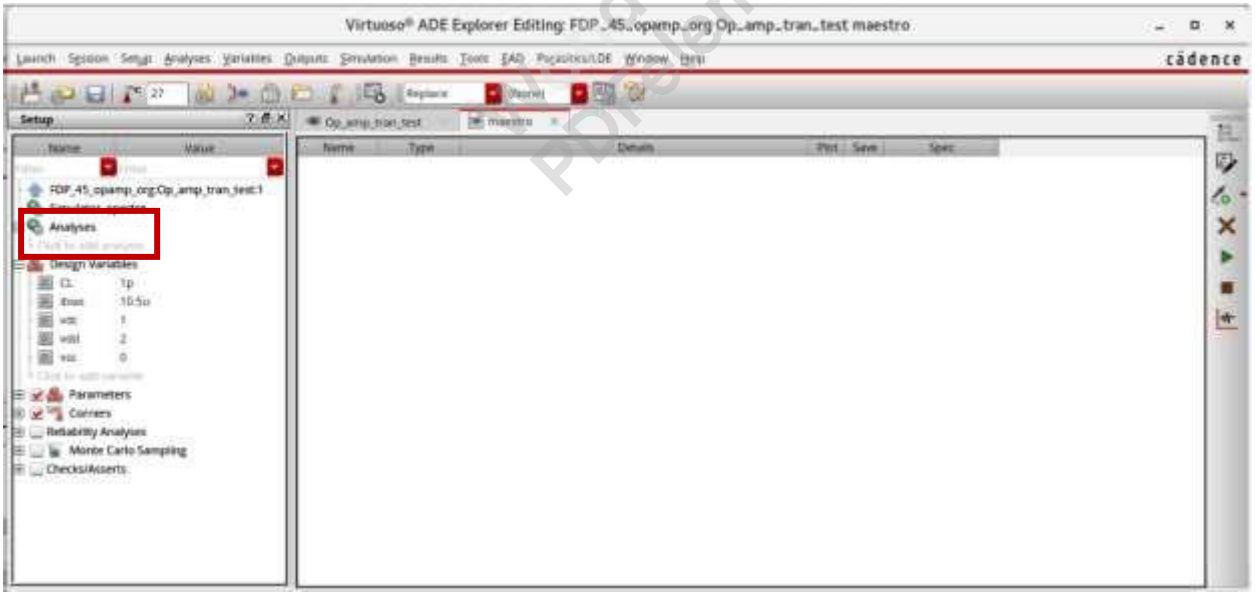


Figure – 4.11: Analyses □ Click to add analysis

To analyze the circuit through Transient Analysis and AC Analysis, select "**Click to addanalysis**" just below the "**Analyses**" option in the "**Setup**" window respective "**.scs**" file and the process corner as "**tt**". Click on "**OK**" as shown in Figure – 4.11.

The "**Choosing Analyses – ADE Explorer**" window pops up as shown in Figure – 4.12. Select the "**tran**" for the "**Transient Analysis**" and "**dc**" for the "**DC Analysis**".

The ADE Explorer window gets updated as shown in Figure – 4.13.
Mention the values for the Design Variables defined in the Schematic of the 2 – Stage

Operational Amplifier.

The defined values for the respective Design Variables are given in Table – 20. The Design Variables along with the values in ADE Explorer window is shown in Figure – 4.13.

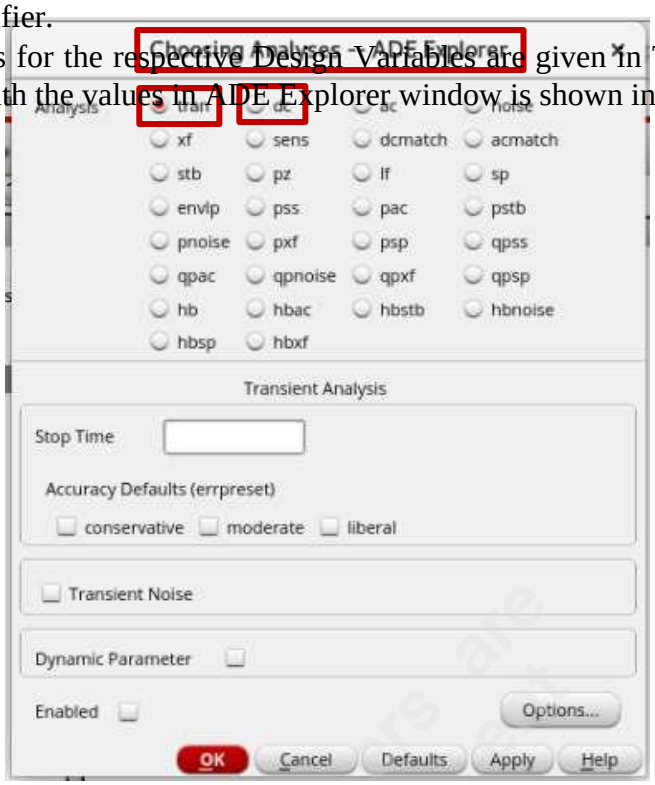


Figure – 4.12: "Choosing Analyses – ADE Explorer" window

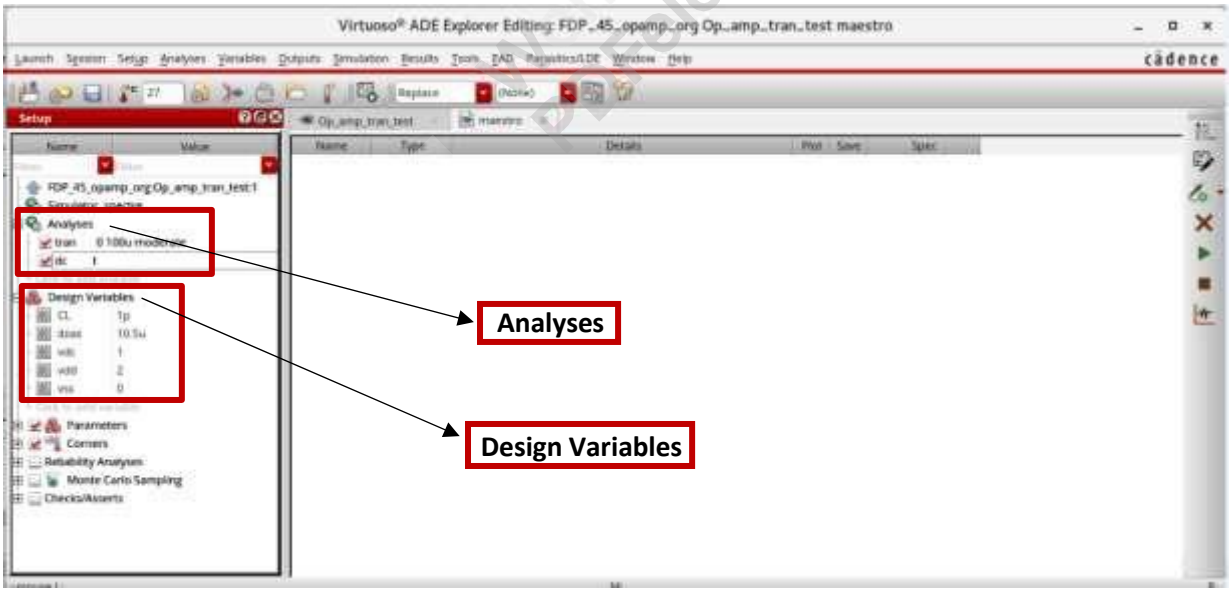


Figure – 4.13: Updated ADE Explorer

Name of the Variable	Value
CL	1p
ibias	10u
vdc	1
vdd	2
vss	0

Table – 20: Design Variables and its Values

To specify the outputs for the simulation, select "**Tools □ Calculator**" as shown in Figure – 4.14.

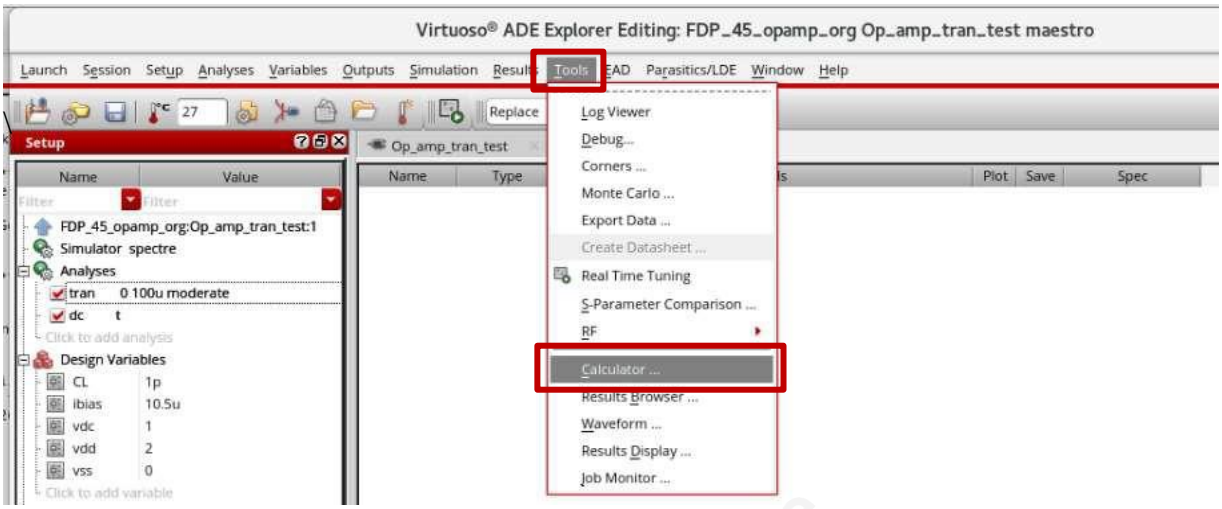


Figure – 4.14: Tools □ Calculator

The "**Virtuoso Visualization & Analysis XL calculator**" window pops up as shown in Figure – 4.15.

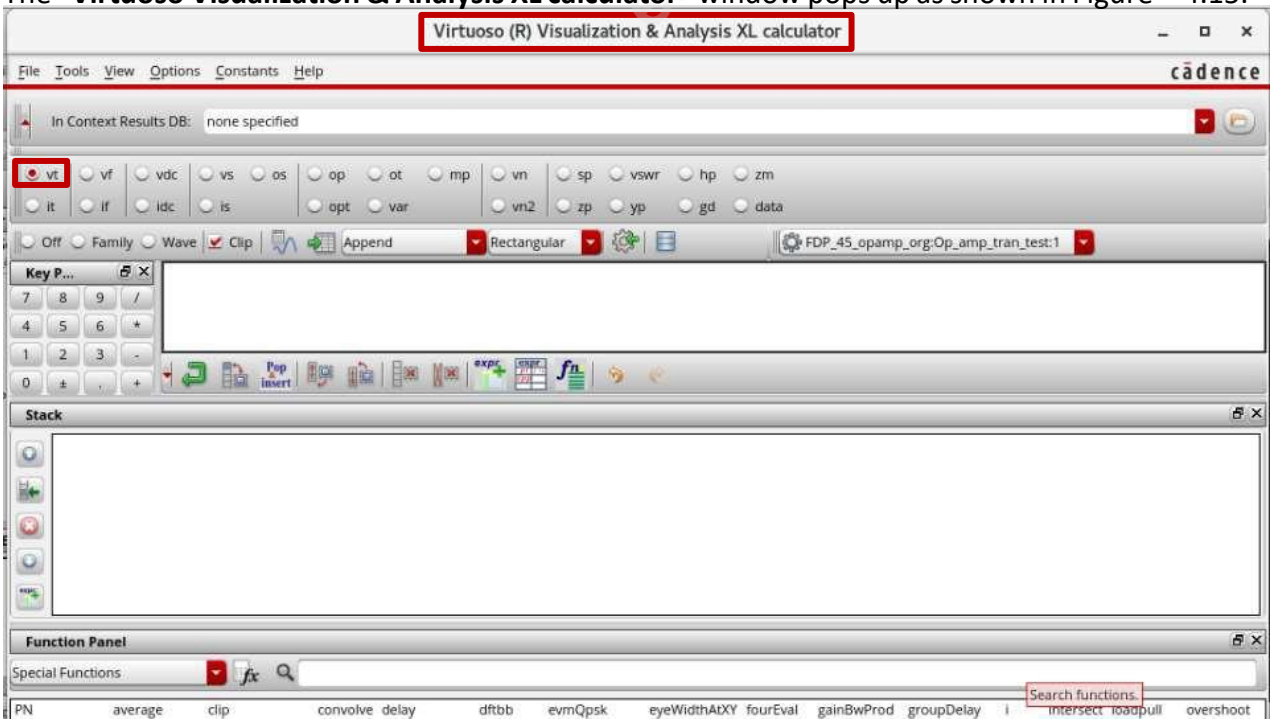


Figure – 4.15: "Virtuoso Visualization & Analysis XL calculator" window

Select "**vt**" as shown in Figure – 4.15. The Test Schematic pops up as shown in Figure – 4.16. Select the output net "**OUT**" from the Schematic and the Buffer window in the Calculator gets updated as shown in Figure – 4.17.

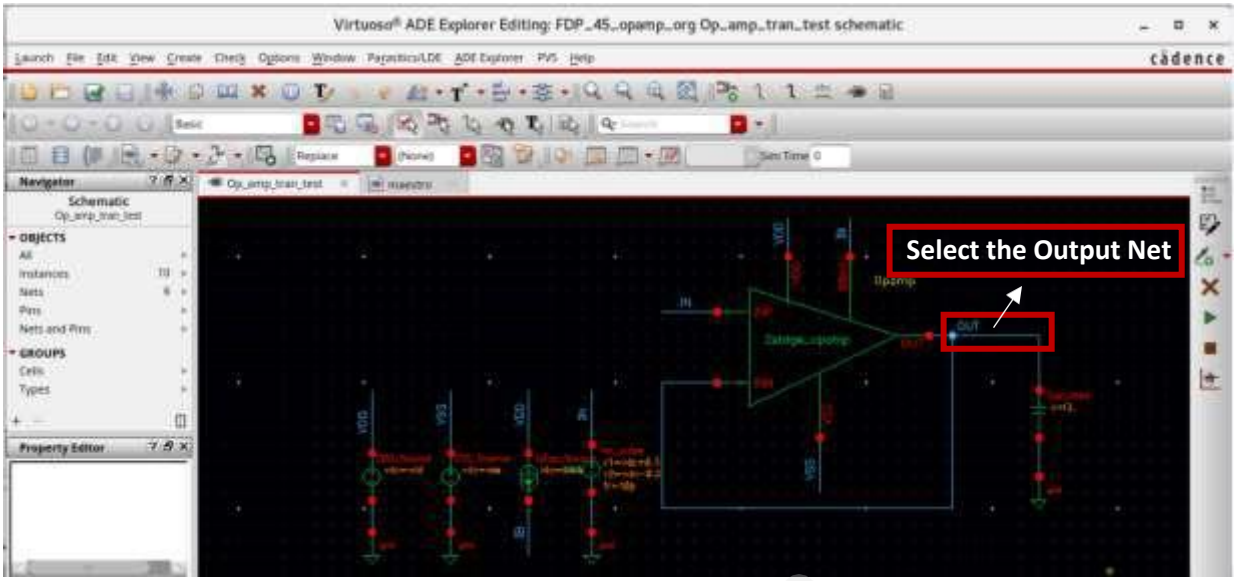


Figure – 4.16: Test Schematic

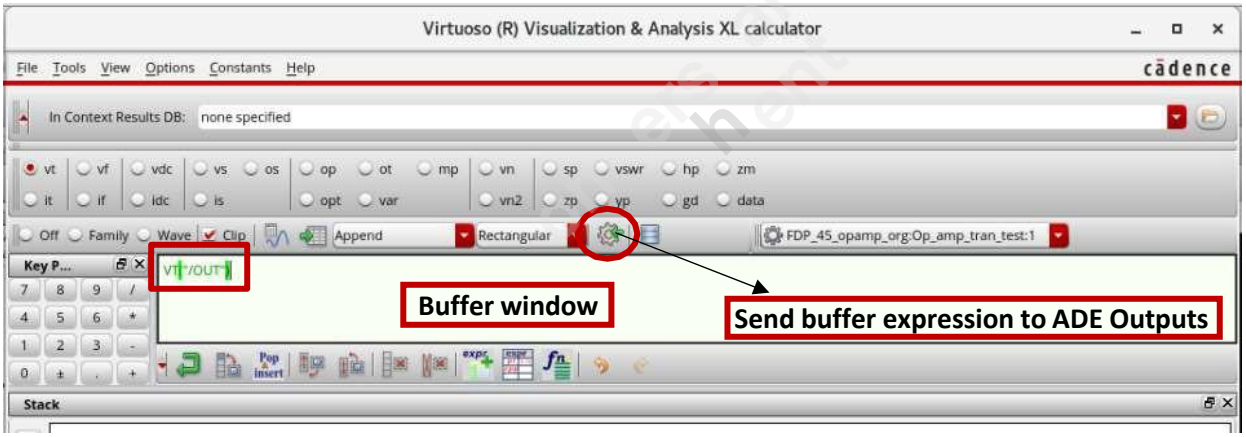


Figure – 4.17: Updated Buffer window

Click on "**Send buffer expression to ADE Outputs**" option to get the expression from the Buffer window into ADE Explorer as shown in Figure – 4.18.

Figure – 4.18: Updated ADE Explorer

Initially, the "**Name**" column would be blank, use the left mouse click to rename (for eg: vout_tran). Similarly, select "**vt**" again, to select the input net "**IN**" and then select "**vdc**" from the calculator, select the input net and the output net from the Test Schematic, rename it for easier identification. The updated ADE Explorer can be seen as shown in Figure – 4.19.

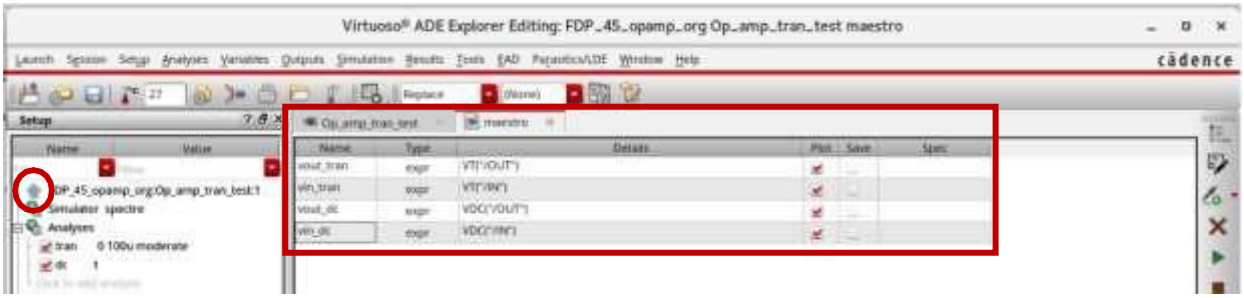


Figure – 4.19: Updated ADE Explorer

Click on the "**Upward Arrow**" just before the Test Circuit name in the **Setup** tab to invoke the **ADE Assembler** as shown in Figure – 4.20. The ADE Assembler allows multiple tests to be simulated on the same environment.

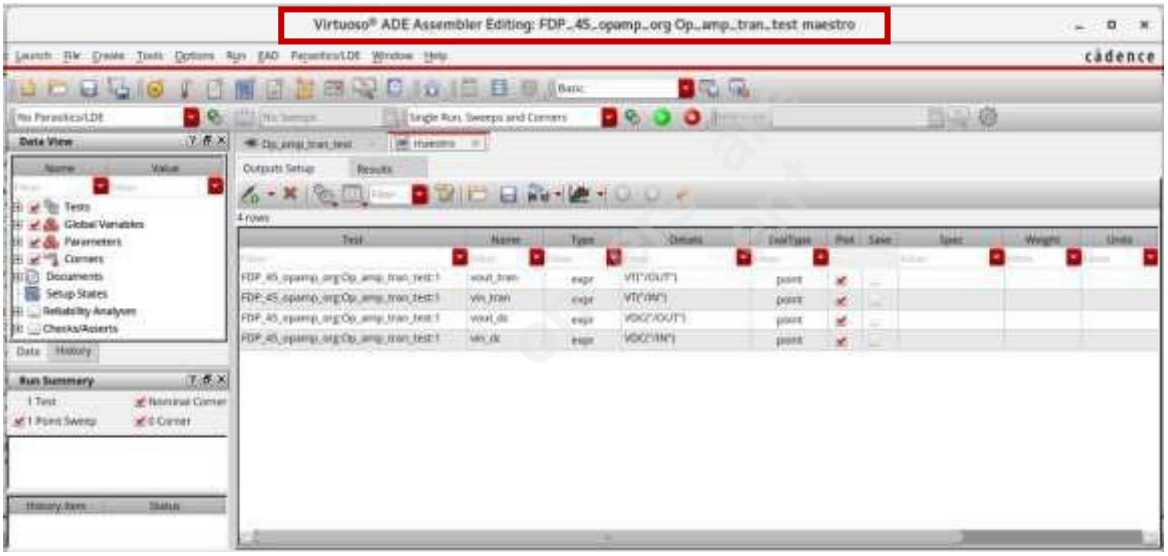


Figure – 4.20: ADE Assembler invoked

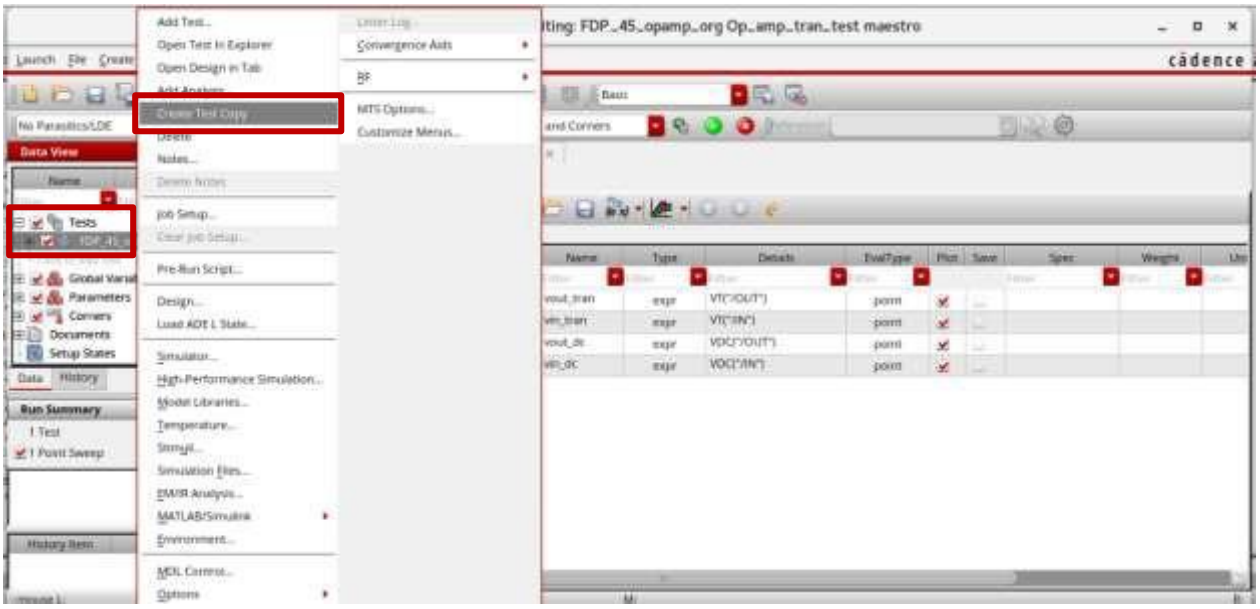


Figure – 4.21: Create Test Copy

Expand "**Tests**" and use the left mouse click to select the Test Circuit and use the right mouse click to select "**Create Test Copy**" as shown in Figure – 4.21.
Use a left mouse click to select the "**Copied Test**" (for eg: **FDP_45_opamp_org:Op_amp_ tran_test:1:1**) as shown in Figure – 4.22. Left mouse click again to rename it to **FDP_45_opamp_org:Op_amp_ ac_test:1** as shown in Figure – 4.22.

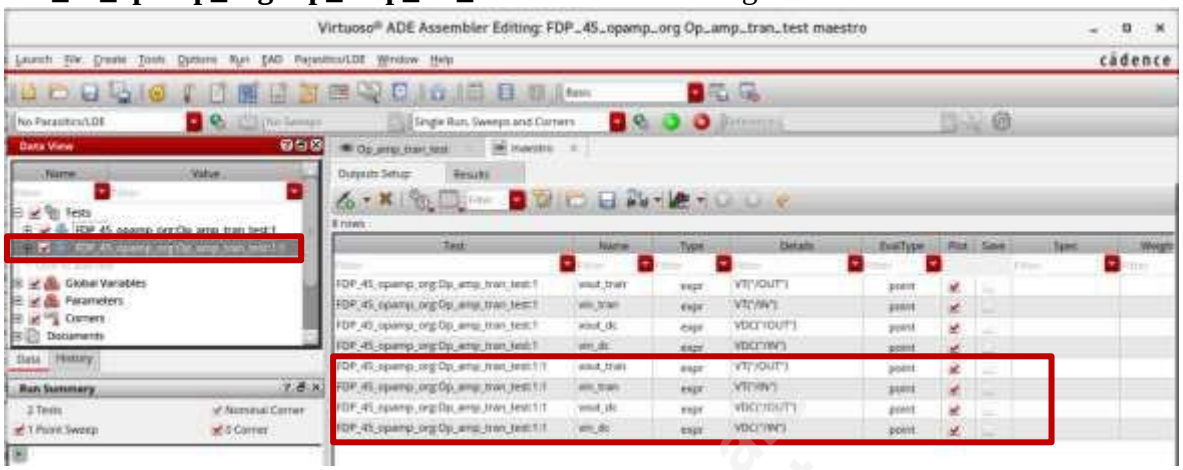


Figure – 4.22: Copied Test

Left mouse click again to rename it to **FDP_45_opamp_org:Op_amp_ ac_test:1** as shown in Figure – 4.23.

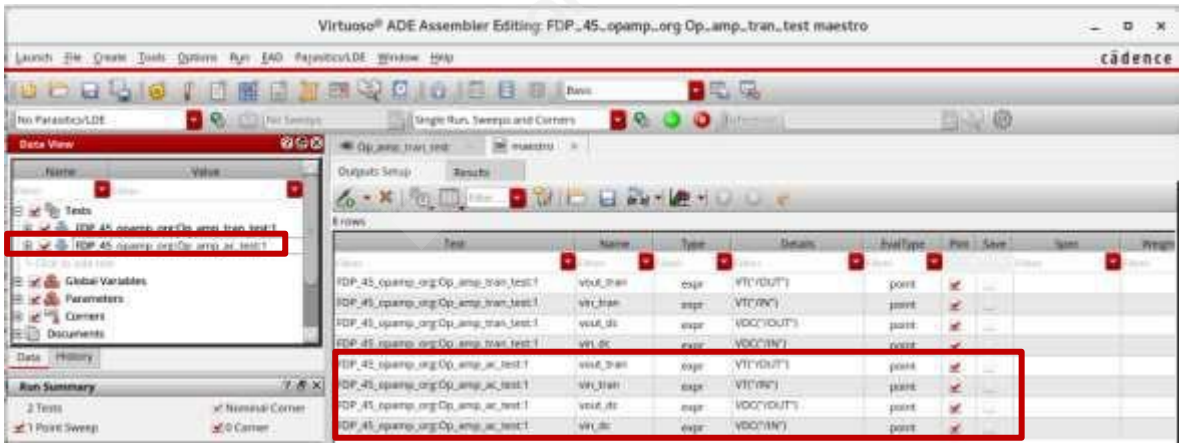


Figure – 4.23: Renamed Test

To verify the selected design for "**ac**" test, right mouse click on the test and select "**Design**" as shown in Figure – 4.24.
The "**Choose Design – ADE Assembler**" window pops up as shown in Figure – 4.25.
Select the **Library, Cell Name, "View Name Schematic"** and click on "**OK**".

Select all the tests related to "**ac**" from the "**Outputs Setup**" and delete them.
Select the "**ac**" test from the "**Data View**" window, expand, select "**Analyses**" and remove the "**tran**" and "**dc**" analysis that were copied.
The updated ADE Assembler window is shown in Figure – 4.26.

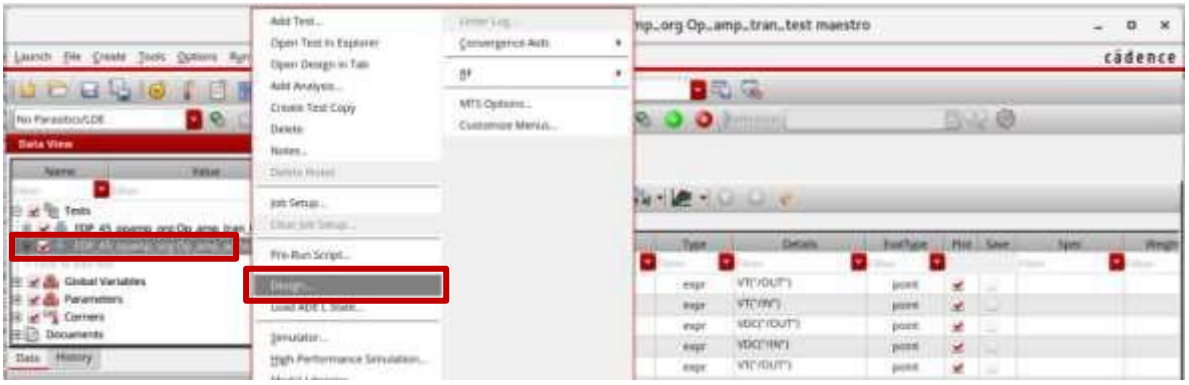


Figure – 4.24: Select "Design" option

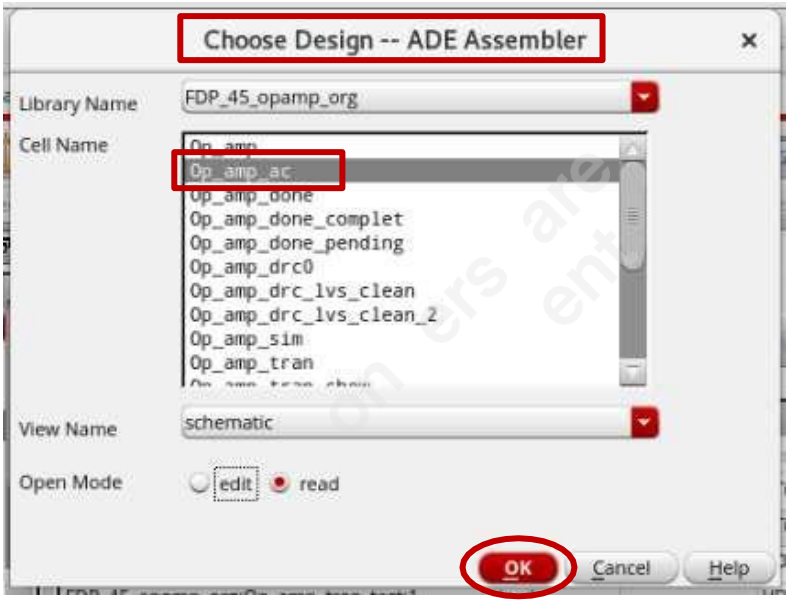


Figure – 4.25: Select the "Op_amp_ac" Cell Name

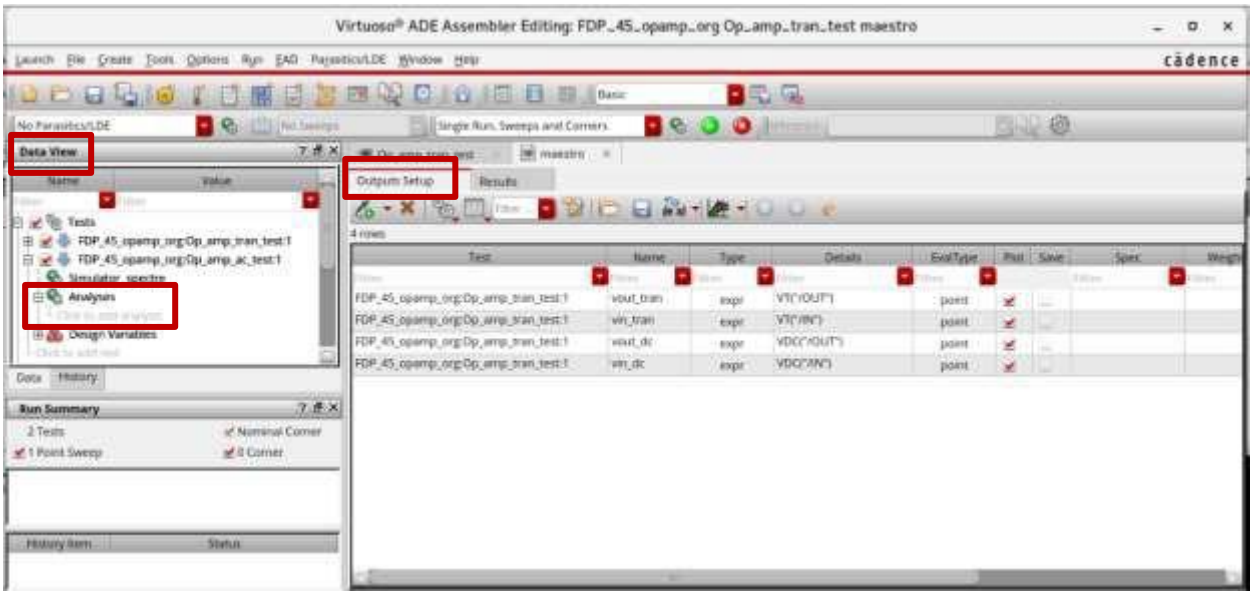


Figure – 4.26: Updated ADE Assembler

Select "**Click to add analysis**" option from the **Analyses** option to select the "**ac**" analysis for the Test Schematic. The parameters are shown in Figure – 4.27.

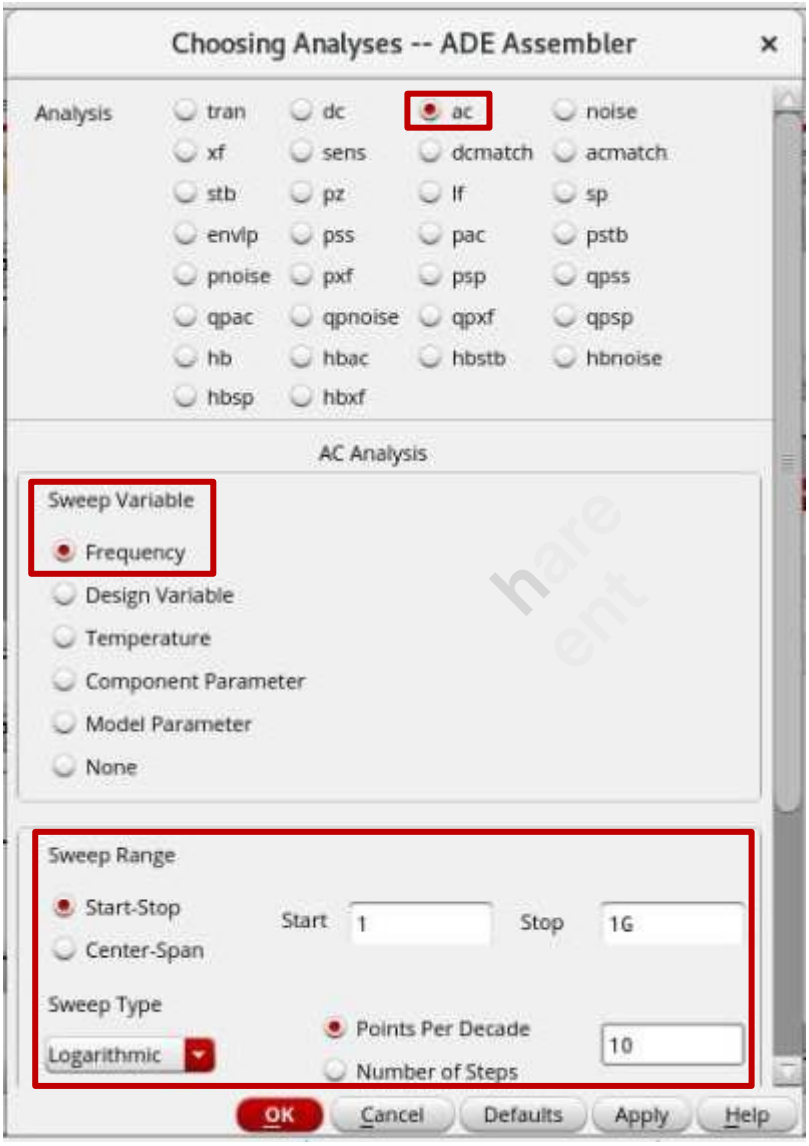


Figure – 4.27: "ac" analysis

Click on "**Apply**", click on "**OK**" to see the ADE Assembler updated as shown in Figure – 4.28.

Figure – 4.28: Updated ADE Assembler with "ac" analysis

Expand the "**Design Variables**", add "**vac**" as the variable and "100m" as its value by selecting the "**Click to add variable**" option. The updated Design Variables are shown in Figure – 4.29.

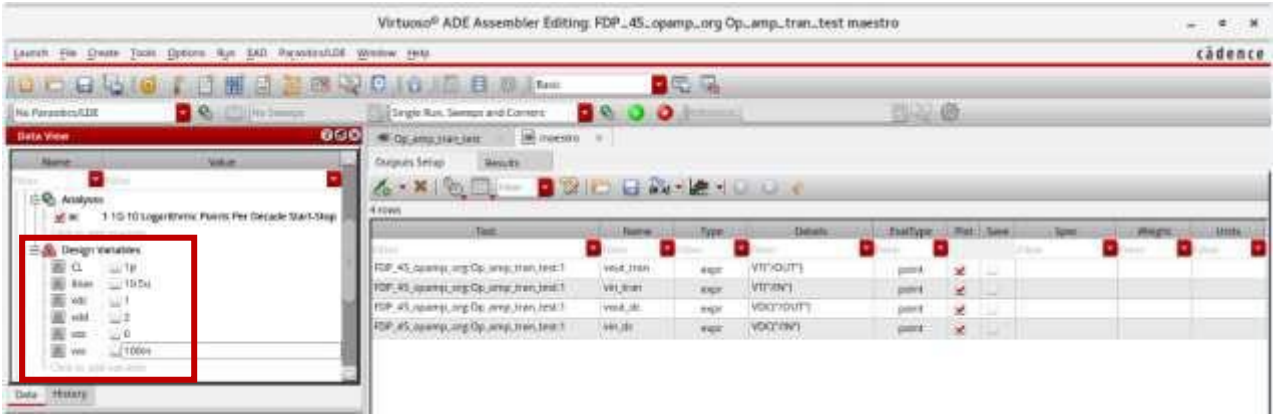


Figure – 4.29: Updated Design Variables

Select "**Tools ➔ Calculator**" and select the "**ac**" analysis test circuit as shown in Figure –4.30.

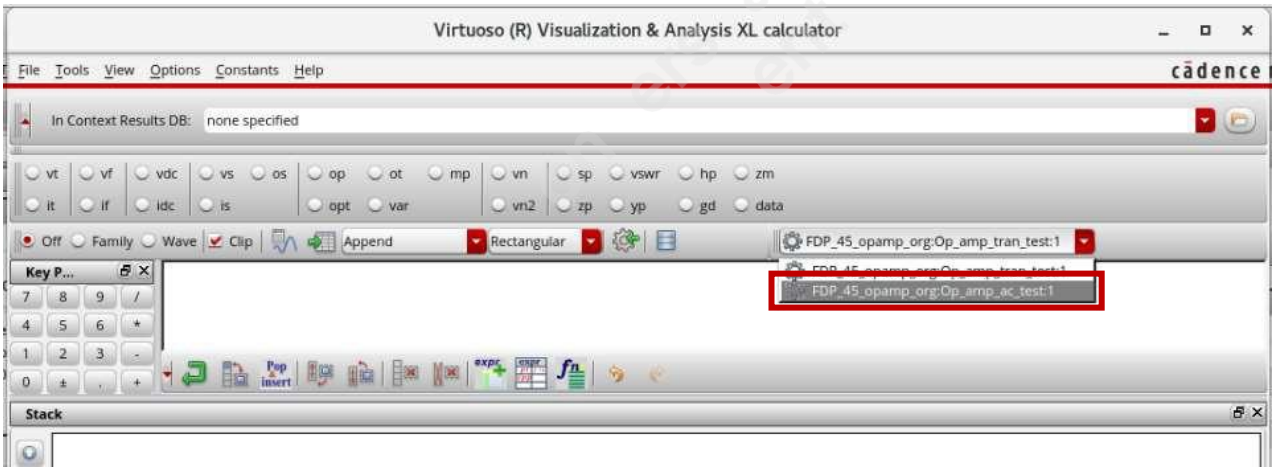


Figure – 4.30: Selecting "ac" test from calculator

Select "**vf**" which accesses voltage over frequency and select the output net from the Test Schematic. The updated Buffer can be seen in Figure – 4.31.

Figure – 4.31: Updated Buffer after "vf" and output net selection

Similarly, select the input net from the Test Schematic. The buffer and stack gets updated as shown in Figure – 4.32.

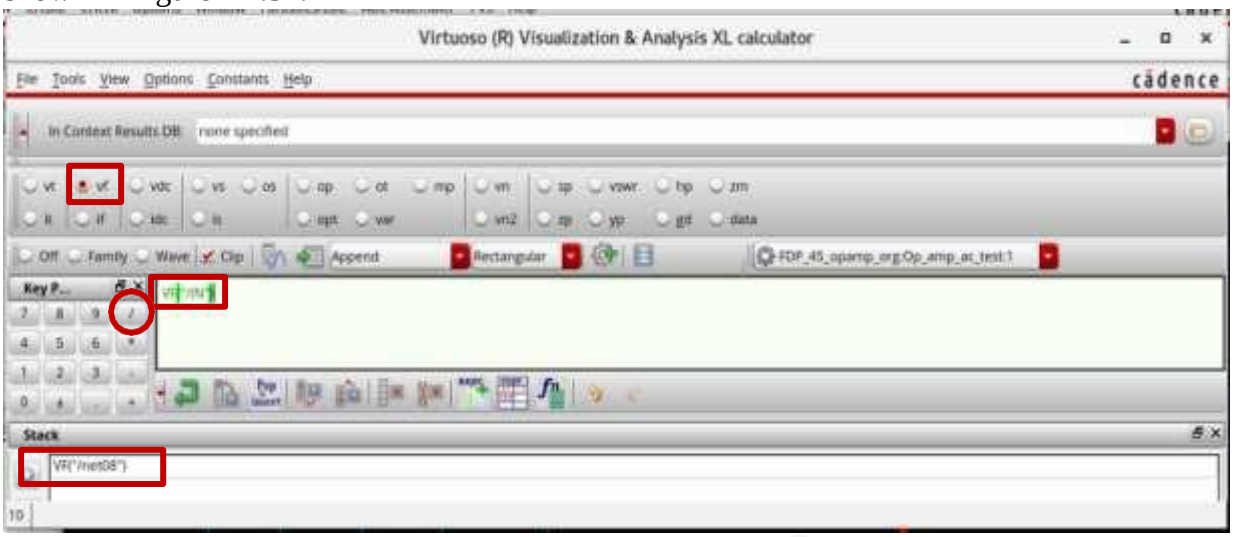


Figure – 4.32: Updated Buffer and Stack

Click on " **/** " from the keypad as shown in Figure – 4.32. The expression in the Buffer gets updated as shown in Figure – 4.33.

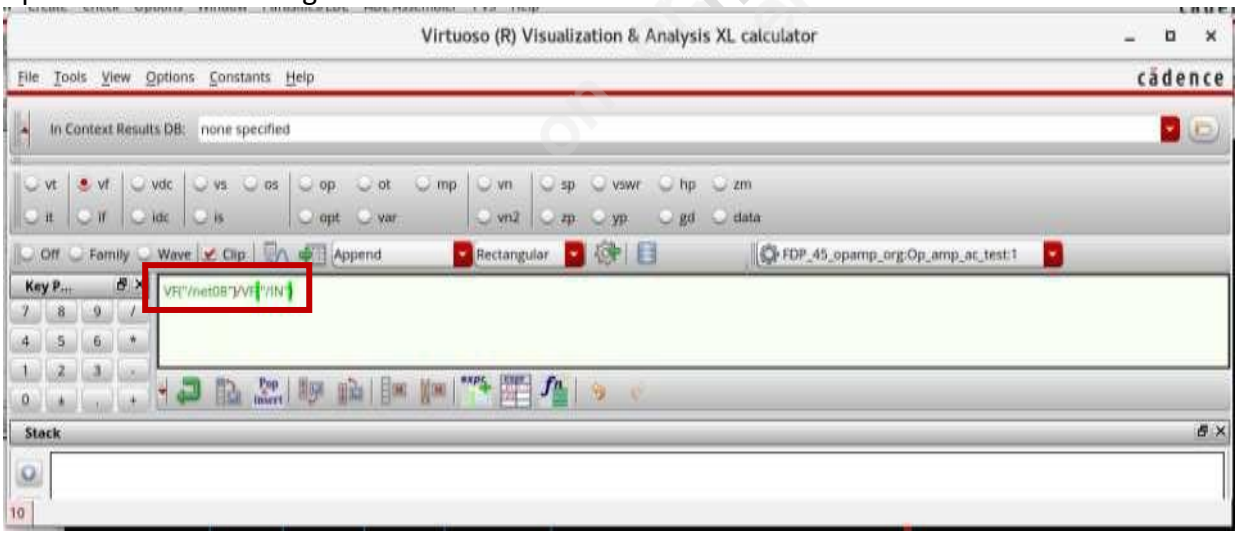


Figure – 4.33: Updated Buffer and Stack

Select "**dB20**" from the Function Panel, the expression in buffer gets updated as shown in Figure – 4.34.

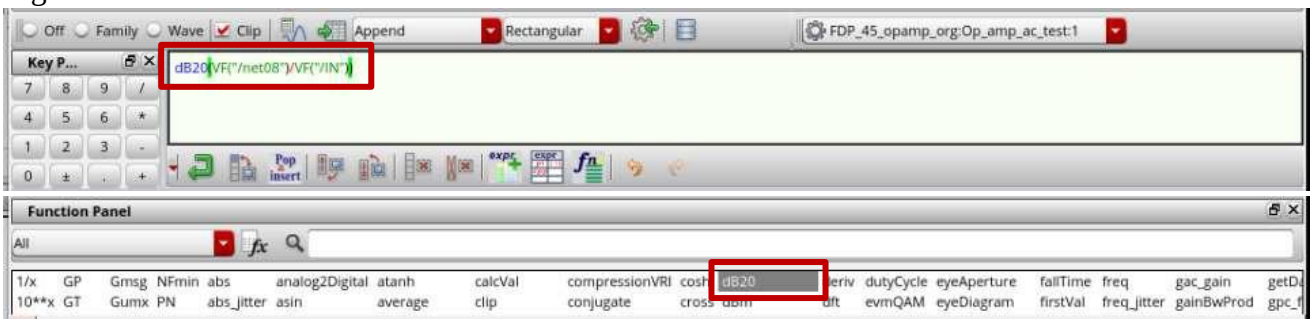


Figure – 4.34: Updated Buffer after selecting "dB20"

This expression calculates the Gain in dB for the Amplifier. Click on "**Send buffer expression to ADE Outputs**". Rename the expression and the updated ADE Assembler can be seen as shown in Figure – 4.35.

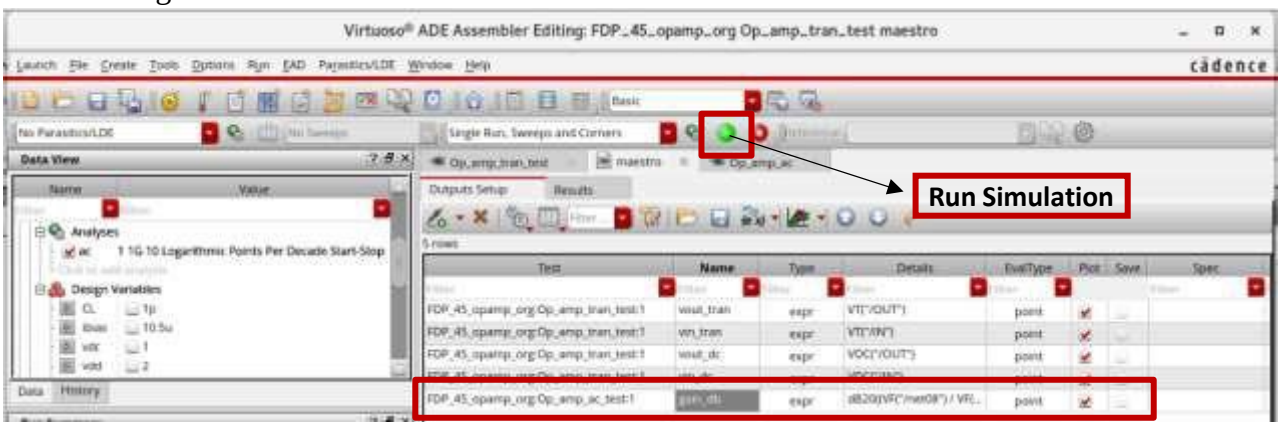


Figure – 4.35: Expression from the Calculator

Click on "**Run Simulation**" option as shown in Figure – 4.35 to simulate the design. The ADEAssembler after simulation is shown in Figure – 4.36.

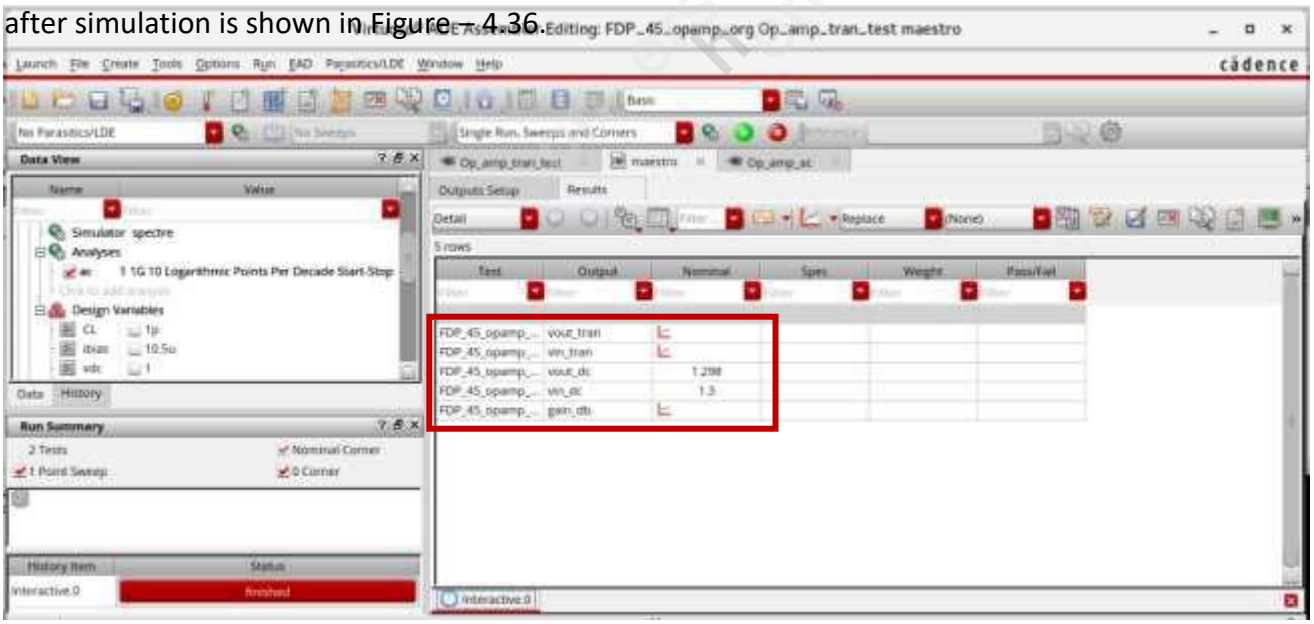


Figure – 4.36: ADE Assembler after simulation

Select "**Options □ Plotting/Printing**" as shown in Figure – 4.37.

Figure – 4.37: Options □ Plotting/Printing

The "**ADE Assembler Plotting/Printing Options**" window pops up as shown in Figure – 4.38. Select "**Plotting Option** □ **Auto**" and **uncheck** "**Plot Scalar Expressions**", click on "**OK**" as shown in Figure – 4.38.

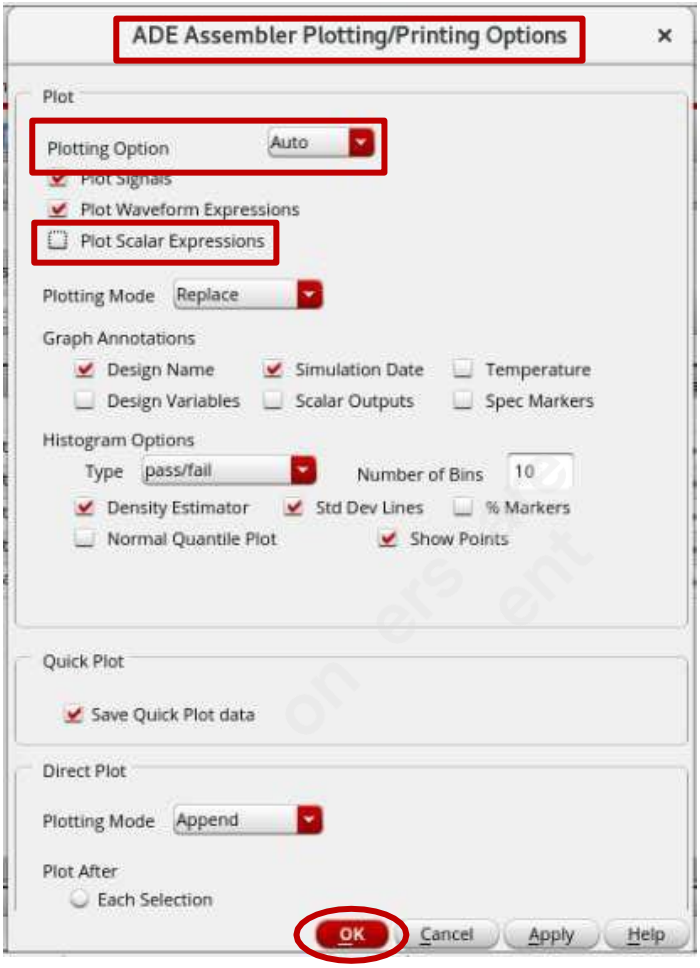


Figure – 4.38: ADE Assembler Plotting/Printing Options" window

Click on "**Plot All**" to see the waveforms as shown in Figure – 4.39.

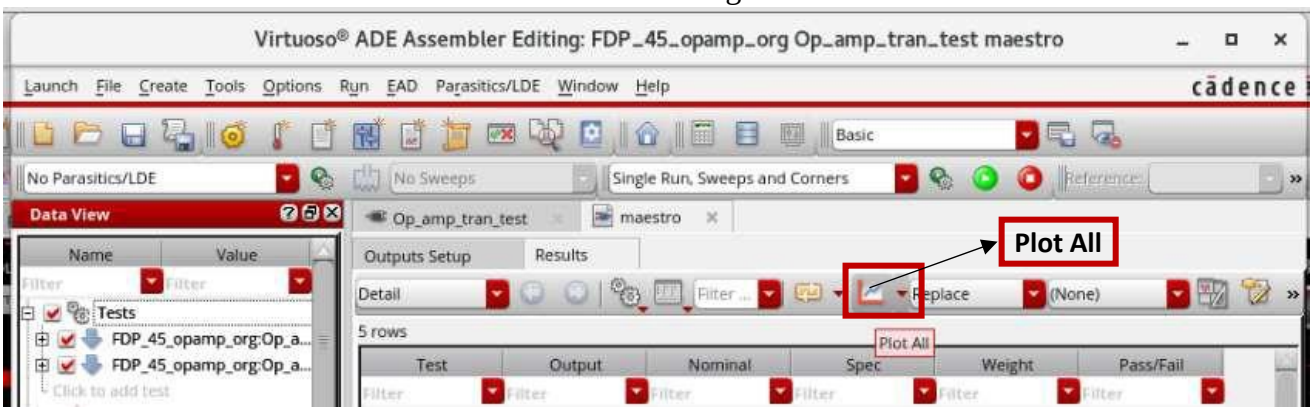


Figure – 4.39: "Plot All" option

The plotted waveforms can be visualized in the "**Virtuoso Visualization & Analysis XL**" window as shown in Figure – 4.40.

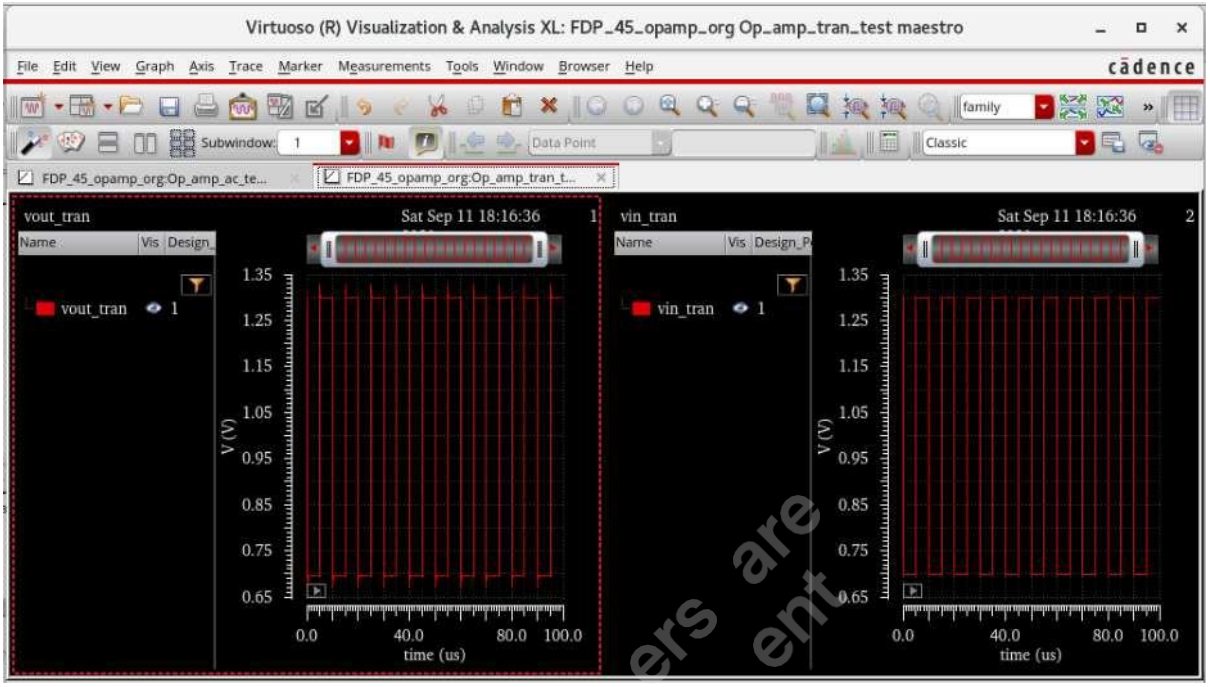


Figure – 4.40: Plotted Waveforms

Select "**Measurements □ Transient Measurement**" as shown in Figure – 4.41.

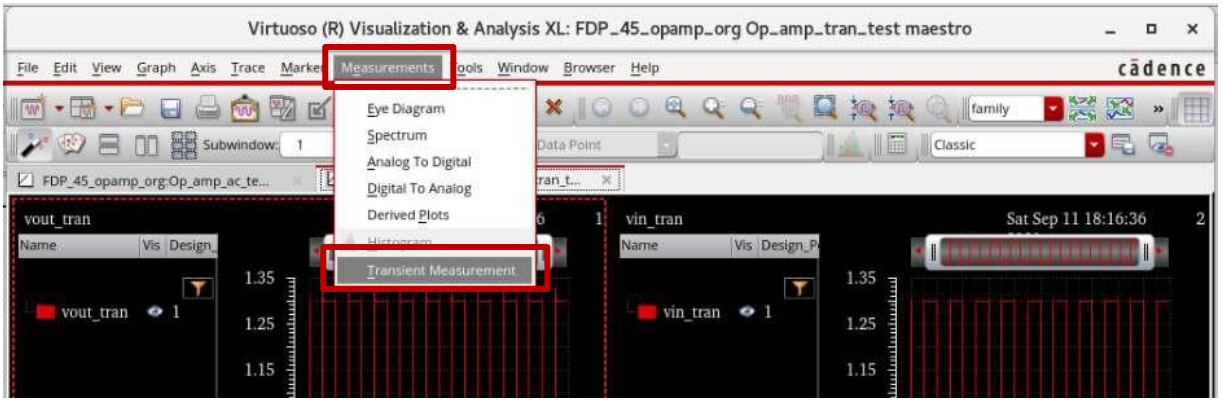


Figure – 4.41: Measurements □ Transient Measurement

The **Slew Rate** can be checked out as shown in Figure – 4.42.

To calculate the **Settling Time**, consider the time the wave takes to reach and stay between (+/- 1%) of its final value in comparison with the initial value. This is calculated as follows:

Lower Bound = 99 % * (1.3 V – 0.7 V) + 0.7 V = 1.294 V

Upper Bound = 101 % * (1.3 V – 0.7 V) + 0.7 V = 1.306 V

Right Mouse click on X-axis properties, "**Independent Axis Properties for time**" window pops up as shown in Figure – 4.43.

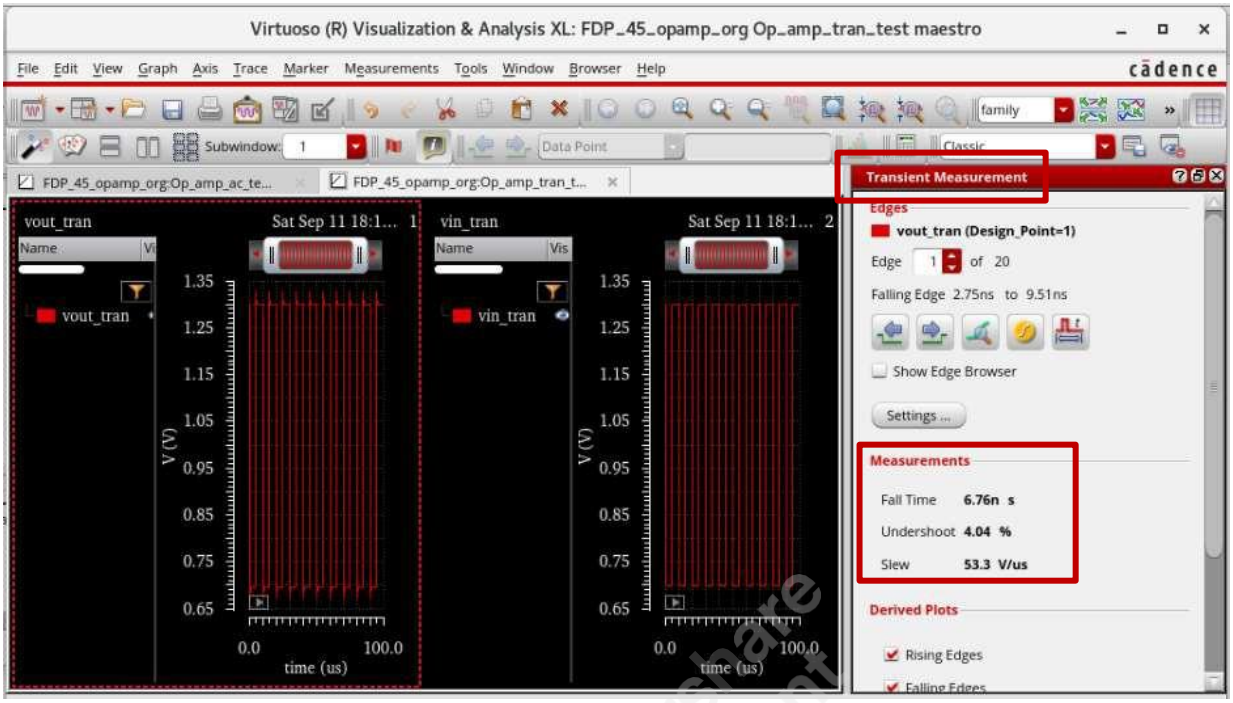


Figure – 4.42: Transient Measurement tab

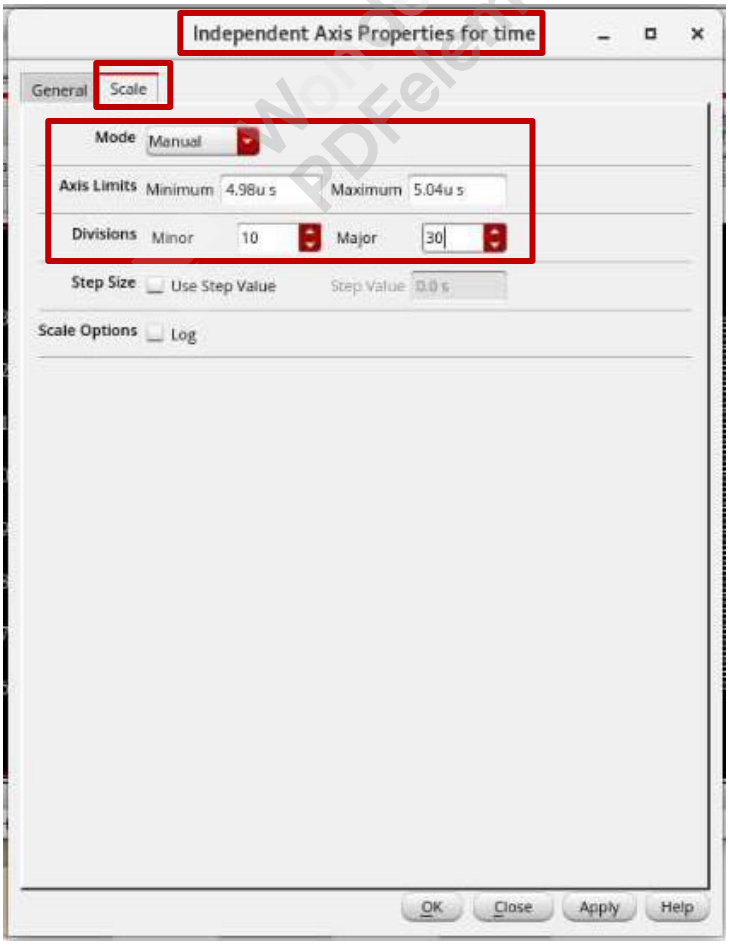


Figure – 4.43: "Independent Axis Properties for time" window

Select the "**Scale**" tab, select "**Mode □ Manual**", mention **Axis Limits** "**Minimum □ 4.98u s**", "**Maximum □ 5.04u s**" and **Divisions** "**Minor □ 10**", "**Major □ 30**", click on "**OK**" as shown in Figure – 4.43. This will isolate the edges that are to be analyzed as shown in Figure – 4.44.

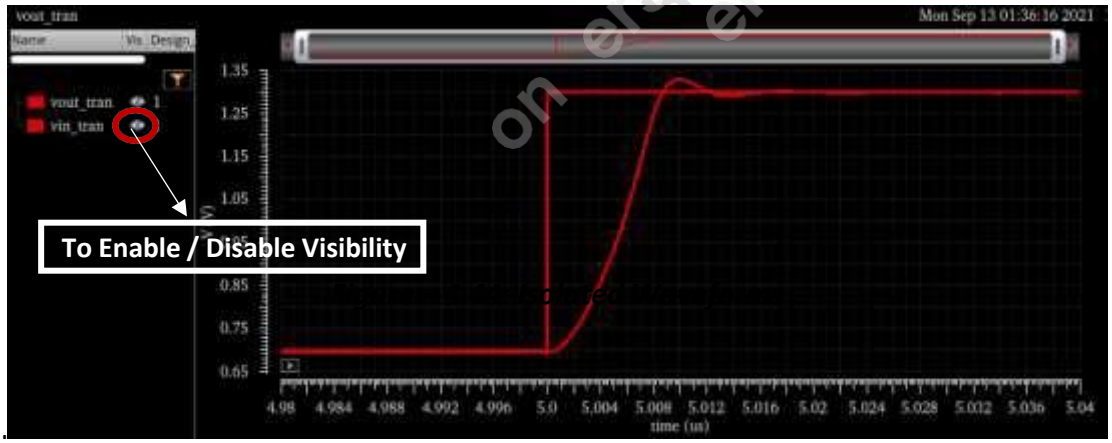


Figure – 4.45: Combined Waveforms

Use the bind key "**M**" to setup a Marker at the required time instance as shown in Figure – 4.46.

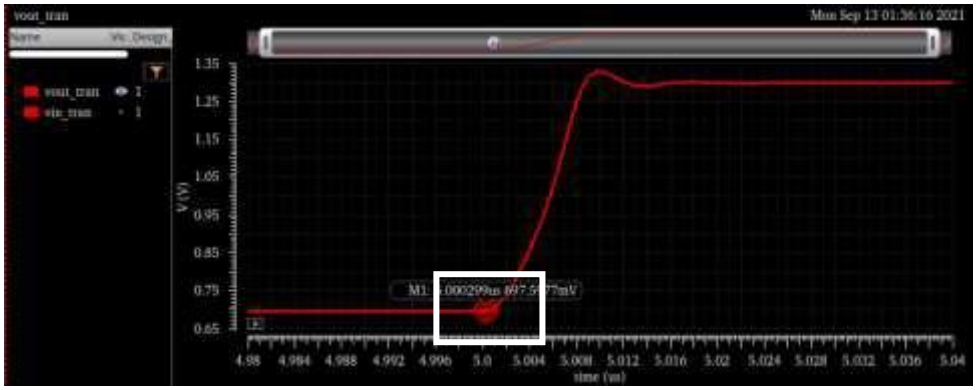


Figure – 4.46: Waveform with Marker

Use the bind key "**H**" to setup horizontal cursors at **1.294 V** and **1.306 V** as shown in Figure – 4.47.

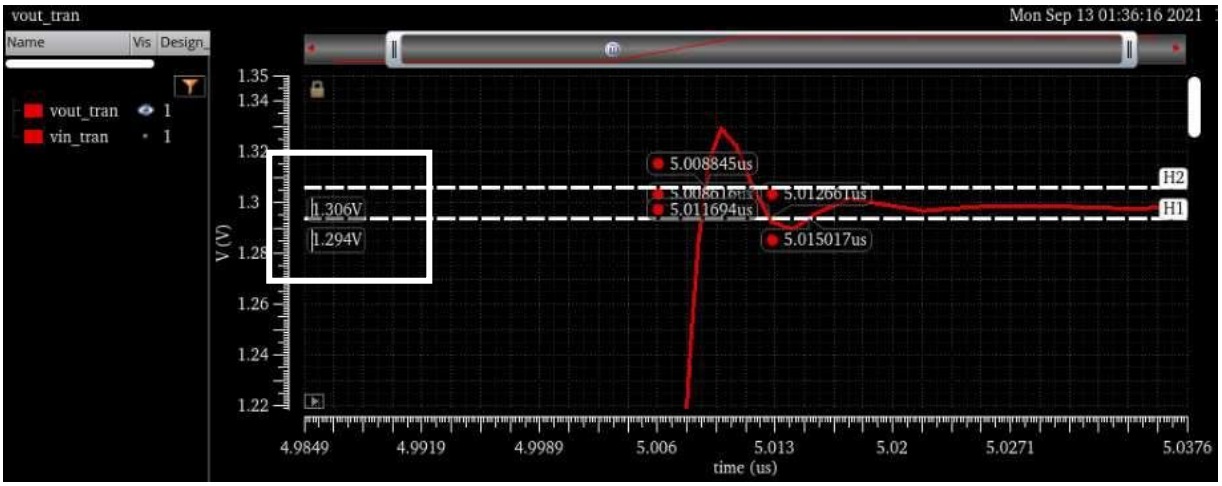


Figure – 4.47: Horizontal cursors at 1.294V and 1.306 V

Figure – 4.48: Marker on the 2nd horizontal cursor

The difference between the timing instances gives the **Settling Time** as **12.5n s**. Without closing the waveform window, open the "**maestro**" in the ADE Assembler.For this simulation, the output dc value is 1.298 V and the input dc value is 1.3 V. The difference gives the **DC Offset (1.298 V – 1.3 V = 2m V)**.

From the AC Analysis curve, set the marker on the low frequency portion of the signal asshown in Figure – 4.49.
The marker reading gives the **DC Open Loop Gain** which is **50.98 dB**.

Setup a horizontal cursor at **0 dB** as shown in Figure – 4.50. The point of intersection of thecursor with the AC Analysis curve gives the Unity Gain Bandwidth.
The **Unity Gain Bandwidth** is measured as **84.51M Hz**.

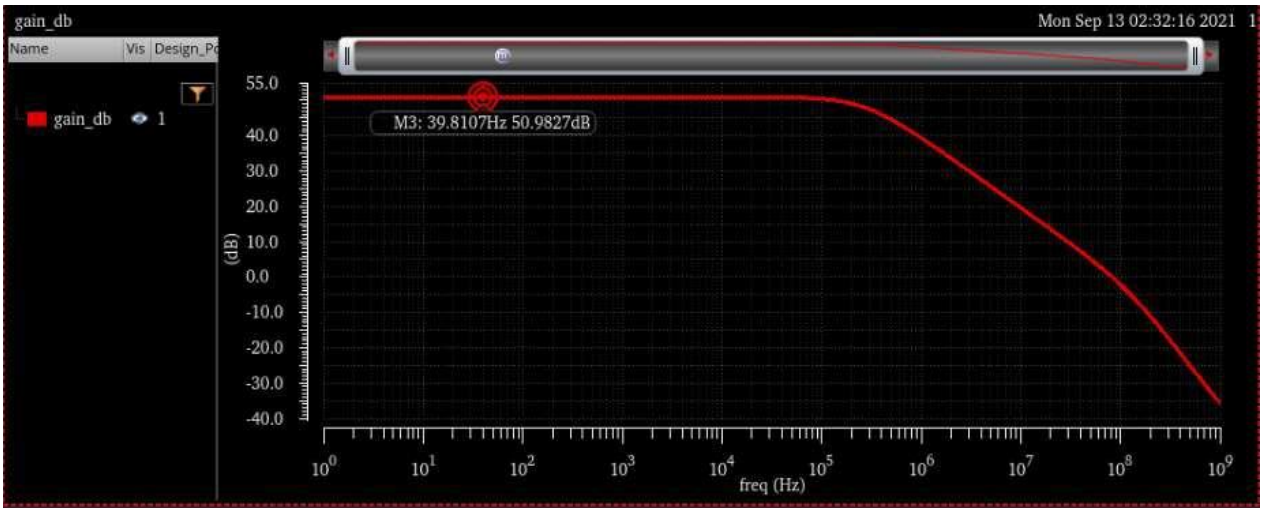
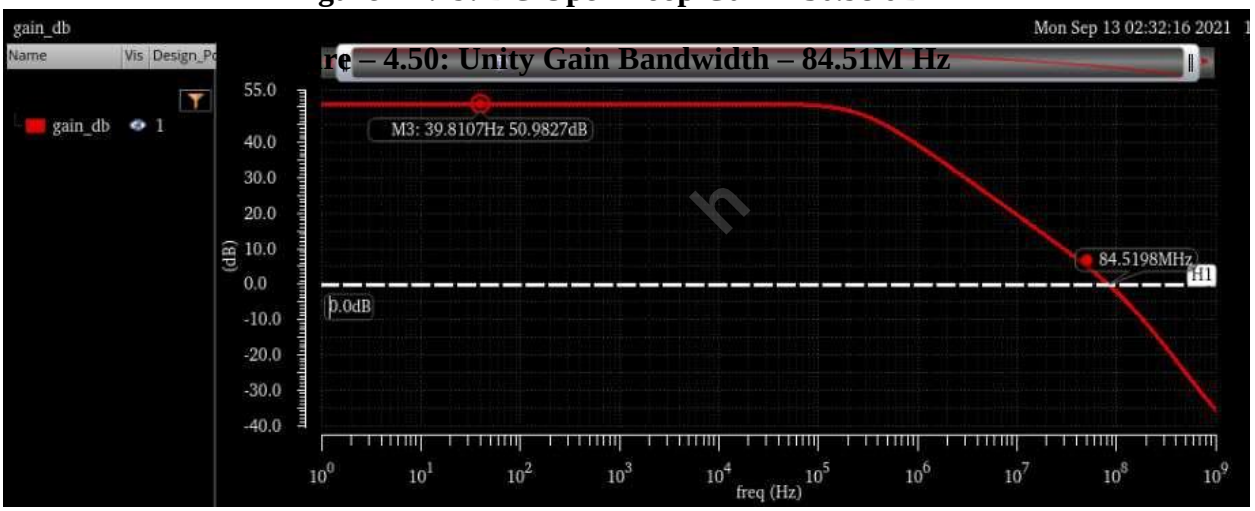


Figure – 4.49: DC Open Loop Gain – 50.98 dB



re – 4.50: Unity Gain Bandwidth – 84.51M Hz

GENERATING THE EXPRESSIONS:

To improve productivity, create reusable expressions rather than using the measurements from waveforms.

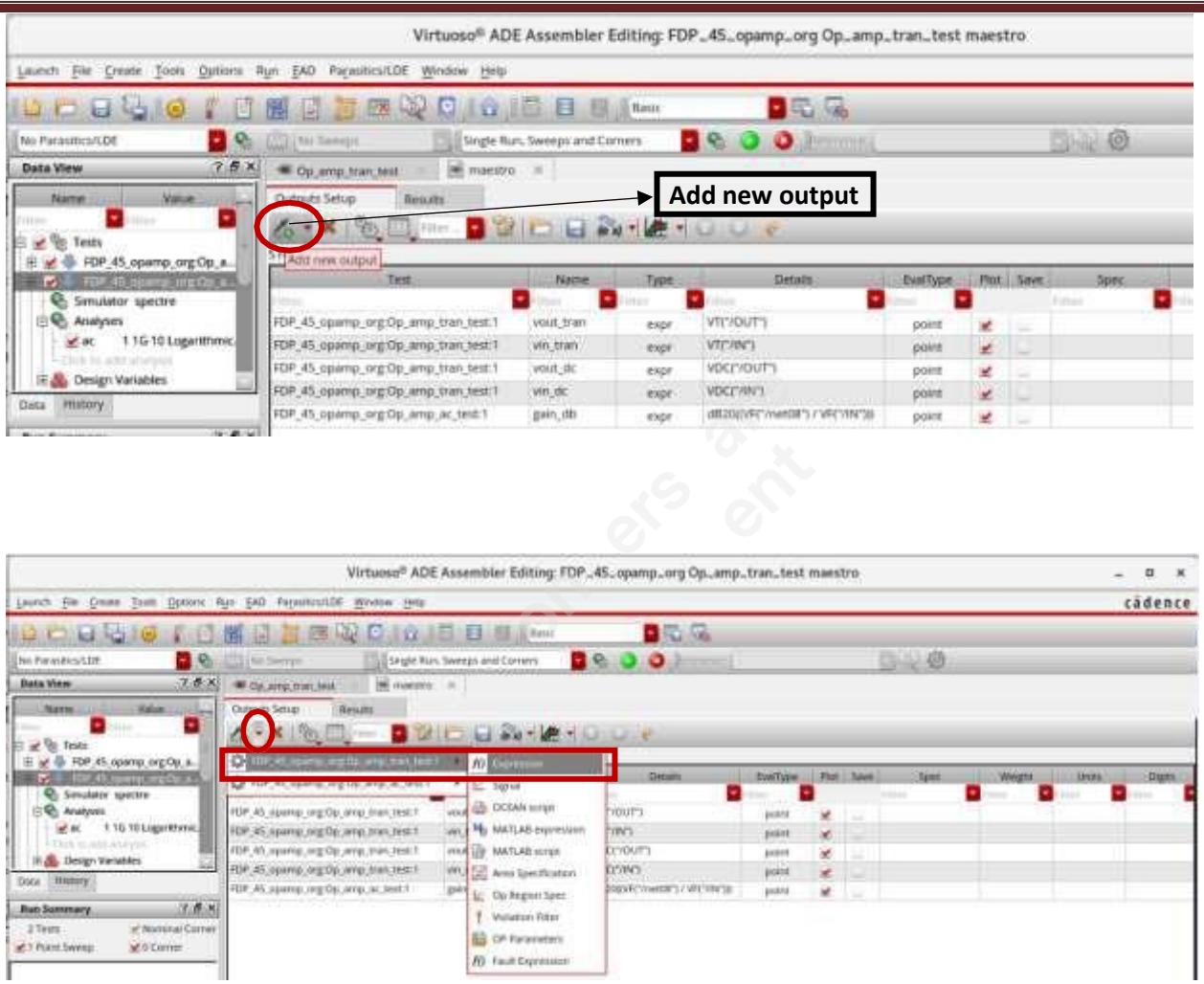


Figure – 4.51: "Add new output" option

To generate the expressions, open the "**Outputs Setup**" tab from ADE Assembler, click on "**Add new output**" as shown in Figure – 4.51.

Figure – 4.52: FDP_45_opamp_org:Op_amp_tran_test □ Expression

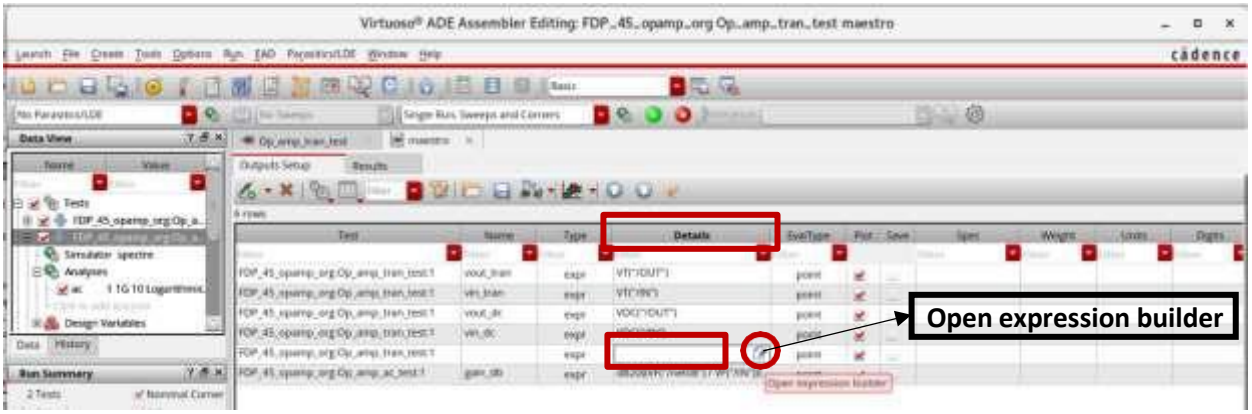
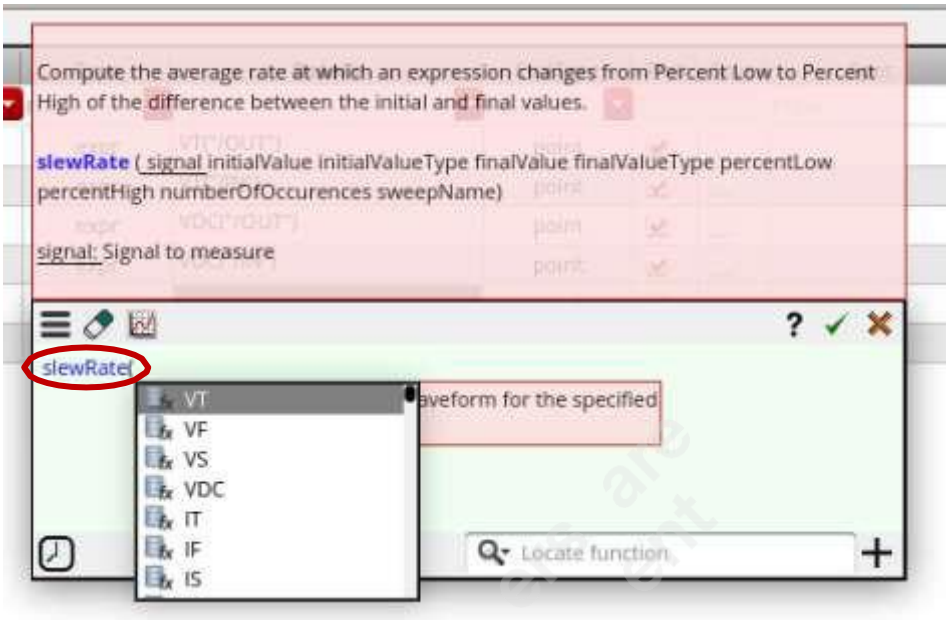


Figure – 4.53: Open expression builder

For the new expression, click on the "**Details**" column and click on "**Open expression builder**" as shown in Figure – 4.53.

Type "**Slew**" and the auto-completion can be seen. Select "**slewRate**" as shown in Figure – 4.54.



Scroll down and select "**vout_tran**", it points to the next parameter. Mention the values and the completed expression can be seen as shown in Figure – 4.55.

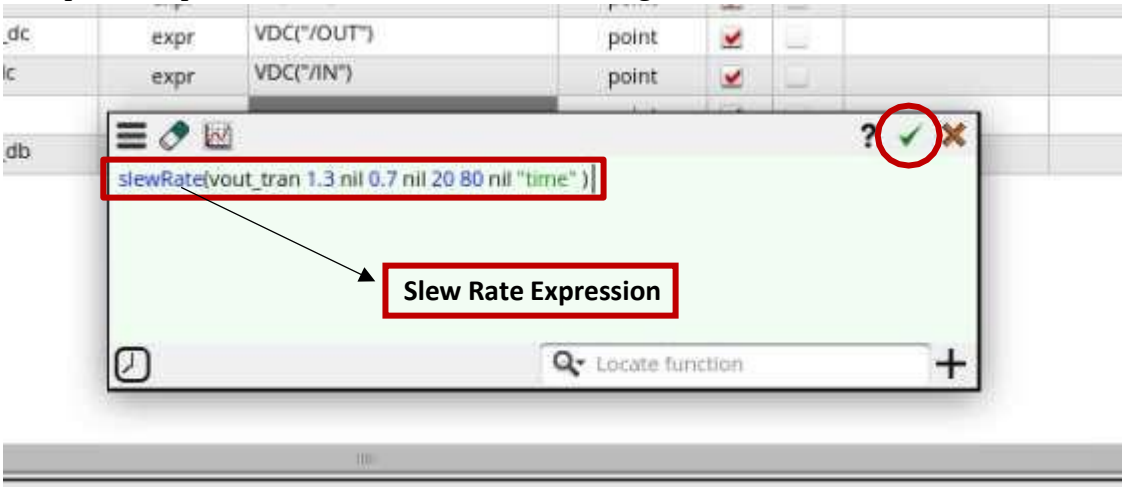


Figure – 4.55: Completed expression

The values for the remaining parameters are given below:

initialValue	- 1.3
initalValueType	- nil
finalValue	- 0.7
finalValueType	- nil

percentLow	- 20
percentHigh	- 80
numberOfOccurences	- nil
sweepName	- time

Click on the "**closing parenthesis**" to complete the expression. Click on the "**Green**" colored tick mark to update the expression in the "**Details**" tab as shown in Figure – 4.56.

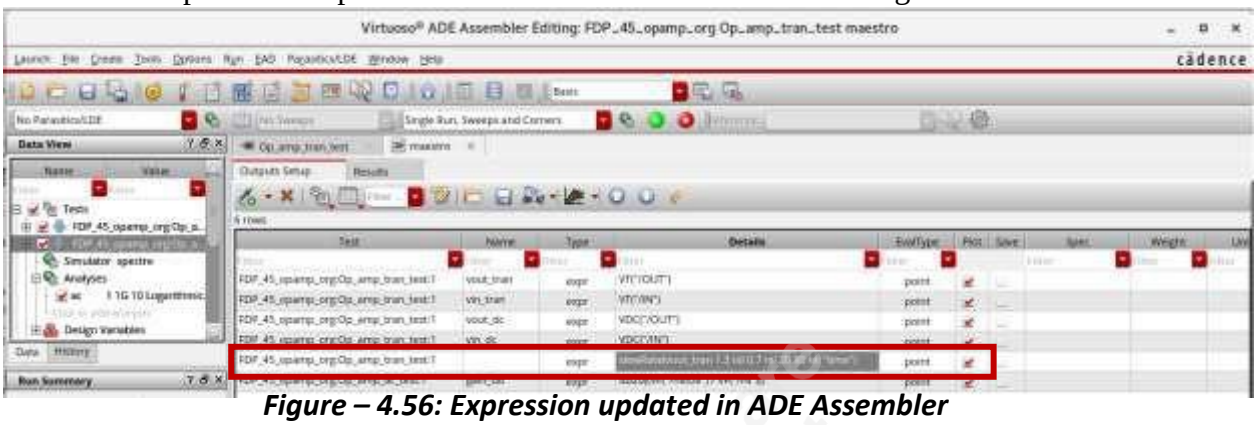


Figure – 4.56: Expression updated in ADE Assembler

Mention a "**Name**" for the created expression as shown in Figure – 4.57.

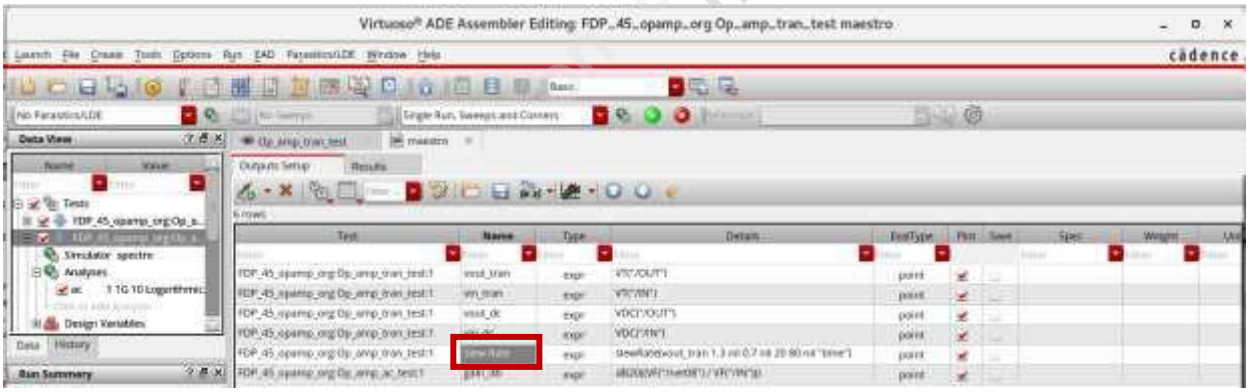
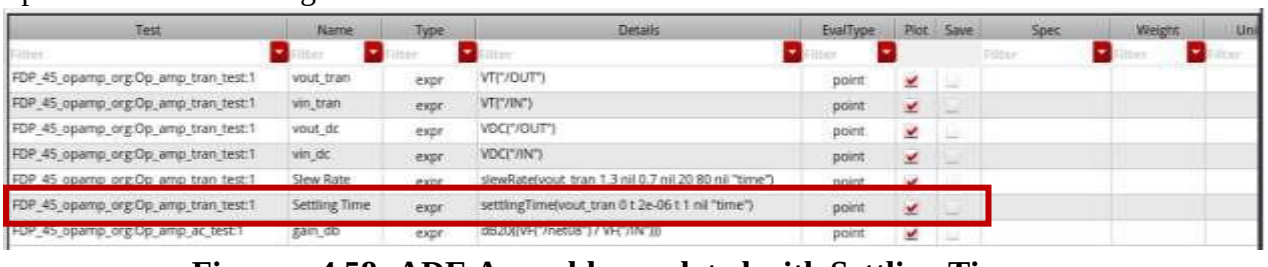


Figure – 4.57: Naming the expression

Similarly, include the expression for Settling Time. After completion, ADE Assembler is updated as shown in Figure – 4.58.



Test	Name	Type	Details	EvalType	Plot	Save	Spec	Weight
FDP_45_opamp_org:Op_amp_tran_test:1	vout_tran	expr	VT("/OUT")	point	✓			
FDP_45_opamp_org:Op_amp_tran_test:1	vin_tran	expr	VT("/IN")	point	✓			
FDP_45_opamp_org:Op_amp_tran_test:1	vout_dc	expr	VDC("/OUT")	point	✓			
FDP_45_opamp_org:Op_amp_tran_test:1	vin_dc	expr	VDC("/IN")	point	✓			
FDP_45_opamp_org:Op_amp_tran_test:1	Slew Rate	expr	slewRate(vout_tran 1.3 nil 0.7 nil 20 80 nil "time")	point	✓			
FDP_45_opamp_org:Op_amp_tran_test:1	Settling Time	expr	settlingTime(vout_tran 0 t 2e-06 t 1 nil "time")	point	✓			
FDP_45_opamp_org:Op_amp_ac_test:1	gain_db	expr	db20(VF("/net08") / VF("/IN" 1))	point	✓			

Figure – 4.58: ADE Assembler updated with Settling Time

The expression for DC Offset is shown in Figure – 4.59.

Figure – 4.59: DC Offset

The expression for dissipated power is shown in Figure – 4.60. After typing " **2 *** ", select "**IDC**" from the list ("2" □ Total Supply voltage range applied on the op-amp). Click on "**Select from design**" and select the top pin of the "**DC Voltage Source**" instantiated for "**VDD**".

To include the expression for DC Gain and Bandwidth, select the AC Analysis and mention the expressions. The expression for Bandwidth is shown in Figure – 4.61.

Figure – 4.61: Bandwidth

The expression for DC Gain is shown in Figure – 4.62.

Figure – 4.62: DC Gain

After defining all the expressions, the ADE Assembler gets updated as shown in Figure – 4.63.

Test	Name	Type	Details	EvalType	Plot
FDP_45_opamp_org:Op_amp_tran_test:1	vout_tran	expr	VT("/OUT")	point	✓
FDP_45_opamp_org:Op_amp_tran_test:1	vin_tran	expr	VT("/IN")	point	✓
FDP_45_opamp_org:Op_amp_tran_test:1	vout_dc	expr	VDC("/OUT")	point	✓
FDP_45_opamp_org:Op_amp_tran_test:1	vin_dc	expr	VDC("/IN")	point	✓
FDP_45_opamp_org:Op_amp_tran_test:1	Slew Rate	expr	slewRate(vout_tran 1.3 nil 0.7 nil 20 80 nil "time")	point	✓
FDP_45_opamp_org:Op_amp_tran_test:1	Settling Time	expr	settlingTime(vout_tran 0 t 2e-06 t 1 nil "time")	point	✓
FDP_45_opamp_org:Op_amp_tran_test:1	DC Offset	expr	(vout_dc - vin_dc)	point	✓
FDP_45_opamp_org:Op_amp_tran_test:1	Power Dissipation	expr	(2 * IDC("/VDD_Source/PLUS"))	point	✓
FDP_45_opamp_org:Op_amp_ac_test:1	gain_db	expr	dB20((VF("/net08") / VF("/IN")))	point	✓
FDP_45_opamp_org:Op_amp_ac_test:1	Bandwidth	expr	cross(gain_db 0 1 "falling" nil nil nil)	point	✓
FDP_45_opamp_org:Op_amp_ac_test:1	DC Gain	expr	value(gain_db 0 ?scale '(roundDown))	point	✓

Figure – 4.63: Updated ADE Assembler with expressions

Go back to the "**Results**" tab in the "**maestro**" and click on "**Re-evaluates results using current settings from the outputs setup table or with partial simulation data**" option as shown in Figure – 4.64 to re-simulate the expressions evaluate the data.

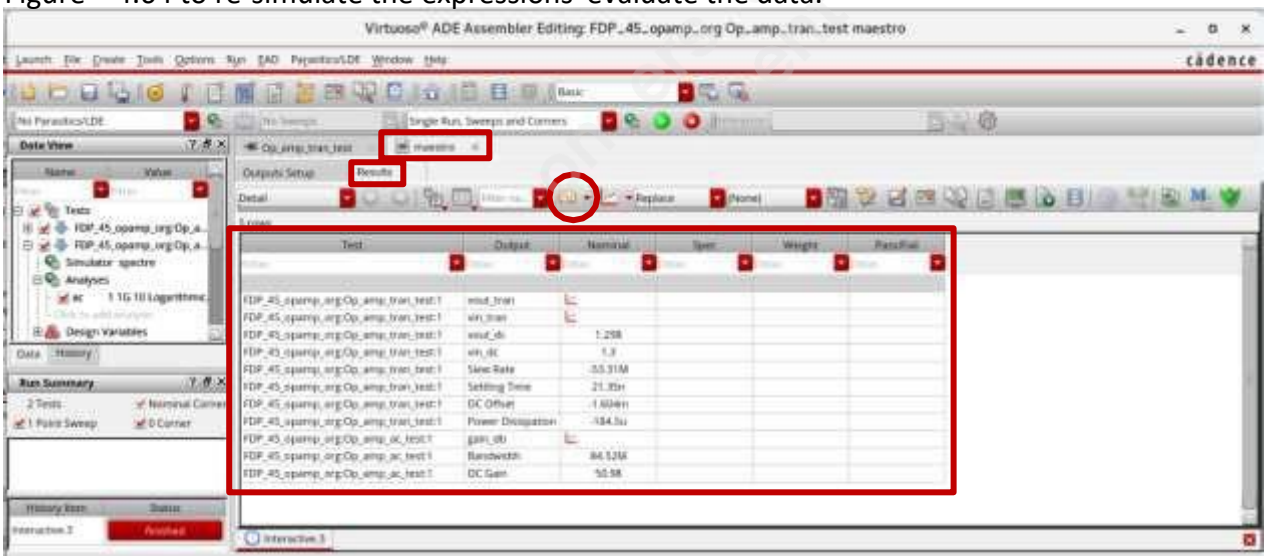


Figure – 4.64: Evaluated Results after re-simulation

Slew Rate and Power Dissipation are seen as negative values after re-simulation. To get the positive values, change the expression on the Outputs Setup as shown in Figure – 4.65.

FDP_45_opamp_org:Op_amp_tran_test:1	vin_dc	expr	VDC("/IN")
FDP_45_opamp_org:Op_amp_tran_test:1	Slew Rate	expr	abs(slewRate(vout_tran 1.3 nil 0.7 nil 20 80 nil "time"))
FDP_45_opamp_org:Op_amp_tran_test:1	Settling Time	expr	settlingTime(vout_tran 0 t 2e-06 t 1 nil "time")
FDP_45_opamp_org:Op_amp_tran_test:1	DC Offset	expr	(vout_dc - vin_dc)
FDP_45_opamp_org:Op_amp_tran_test:1	Power Dissipation	expr	abs((2 * IDC("/VDD_Source/PLUS")))
FDP_45_opamp_org:Op_amp_ac_test:1	gain_db	expr	dB20((VF("/net08") / VF("/IN")))

Figure – 4.65: Modified expressions to get positive values

Specifications can be mentioned as shown in Figure – 4.66.

Test	Name	Type	Details	EvalType	Plot	Save	Spec
FDP_45_opamp_org:Op_amp_tran_test:1	vout_tran	expr	VT("/OUT")	point	✓		
FDP_45_opamp_org:Op_amp_tran_test:1	vin_tran	expr	VT("/IN")	point	✓		
FDP_45_opamp_org:Op_amp_tran_test:1	vout_dc	expr	VDC("/OUT")	point	✓		
FDP_45_opamp_org:Op_amp_tran_test:1	vin_dc	expr	VDC("/IN")	point	✓		
FDP_45_opamp_org:Op_amp_tran_test:1	Slew Rate	expr	abs(slewRate(vout_tran 1.3 nil 0.7 nil 20 80 nil "time"))	point	✓		> 50M
FDP_45_opamp_org:Op_amp_tran_test:1	Settling Time	expr	settlingTime(vout_tran 0 t 2e-06 t 1 nil "time")	point	✓		< 50n
FDP_45_opamp_org:Op_amp_tran_test:1	DC Offset	expr	(vout_dc - vin_dc)	point	✓		
FDP_45_opamp_org:Op_amp_tran_test:1	Power Dissipation	expr	abs((2 * IDC("/VDD_Source/PLUS")))	point	✓		< 200u
FDP_45_opamp_org:Op_amp_ac_test:1	gain_db	expr	dB20((VF("/net08") / VF("/IN")))	point	✓		
FDP_45_opamp_org:Op_amp_ac_test:1	Bandwidth	expr	cross(gain_db 0 1 "falling" nil nil nil)	point	✓		> 50M
FDP_45_opamp_org:Op_amp_ac_test:1	DC Gain	expr	value(gain_db 0 ?scale '(roundDown))	point	✓		> 60

Figure – 4.66: Specifications

After re-evaluation, the results can be seen as shown in Figure – 4.67.

Test	Output	Nominal	Spec	Weight	Pass/Fail
FDP_45_opamp_org:Op_amp_tran_test:1	vout_tran				
FDP_45_opamp_org:Op_amp_tran_test:1	vin_tran				
FDP_45_opamp_org:Op_amp_tran_test:1	vout_dc	1.298			
FDP_45_opamp_org:Op_amp_tran_test:1	vin_dc	1.3			
FDP_45_opamp_org:Op_amp_tran_test:1	Slew Rate	53.31M	> 50M		pass
FDP_45_opamp_org:Op_amp_tran_test:1	Settling Time	21.35n	< 50n		pass
FDP_45_opamp_org:Op_amp_tran_test:1	DC Offset	-1.604m			
FDP_45_opamp_org:Op_amp_tran_test:1	Power Dissipation	184.5u	< 200u		pass
FDP_45_opamp_org:Op_amp_ac_test:1	gain_db				
FDP_45_opamp_org:Op_amp_ac_test:1	Bandwidth	84.52M	> 50M		pass
FDP_45_opamp_org:Op_amp_ac_test:1	DC Gain	50.98	> 60		fail

Figure – 4.67: Results after re-evaluation

GAIN MARGIN AND PHASE MARGIN:

The Schematic for measuring the Gain Margin and Phase Margin is shown in Figure – 4.68.

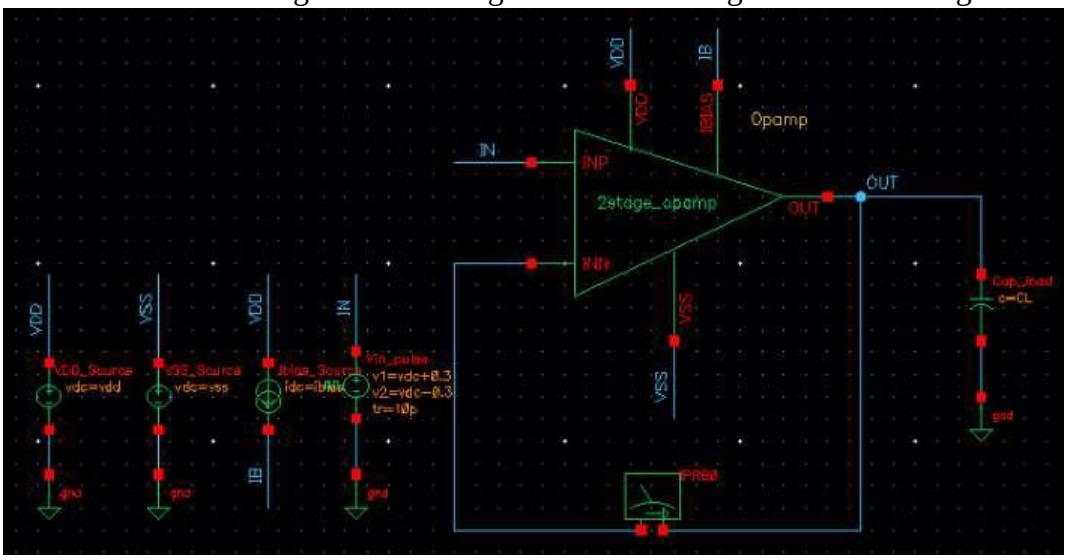
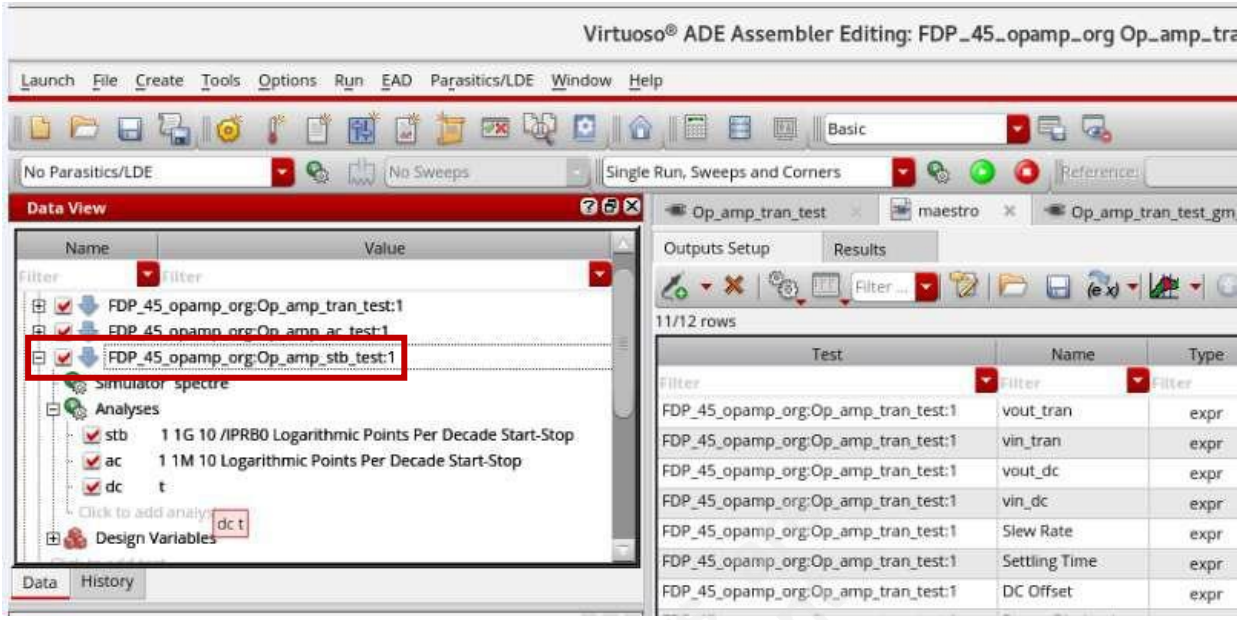


Figure – 4.68: Schematic for Gain Margin and Phase Margin measurement

The "**iprobe**" (available in "**analogLib**") acts as a signal source for the stability analysis. Create a Test Copy for the Stability Analysis as shown in Figure – 4.69.



Browse "**Design □ Op_amp_tran_test_gm_pm**" as shown in Figure – 4.70.

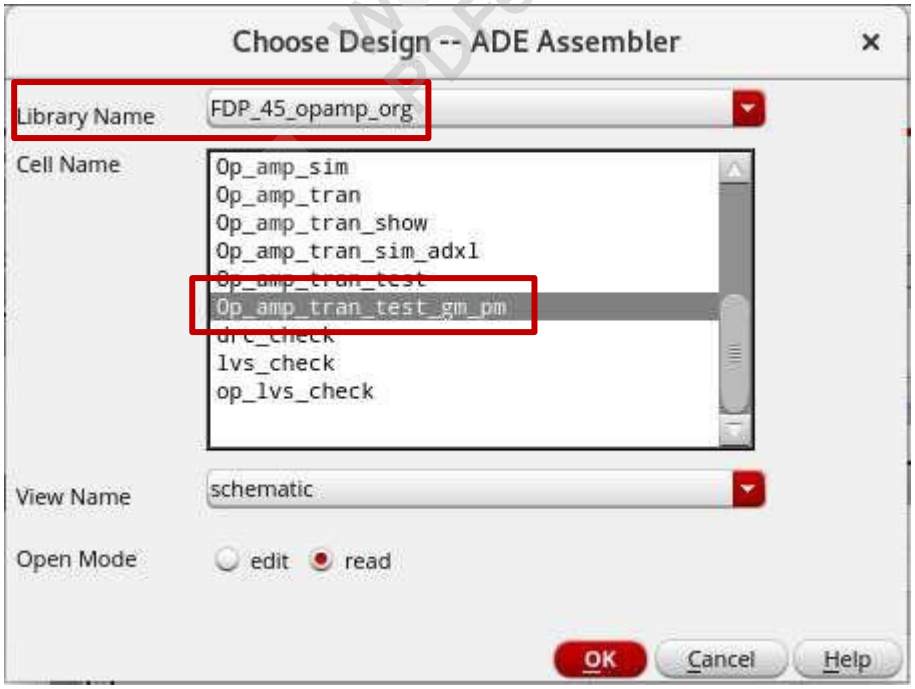


Figure – 4.70: Design Selection

Choose "**stb**" through "**Analyses □ Click to add analysis □ Choosing Analyses – ADE Assembler**". The parameters are shown in Figure – 4.71.

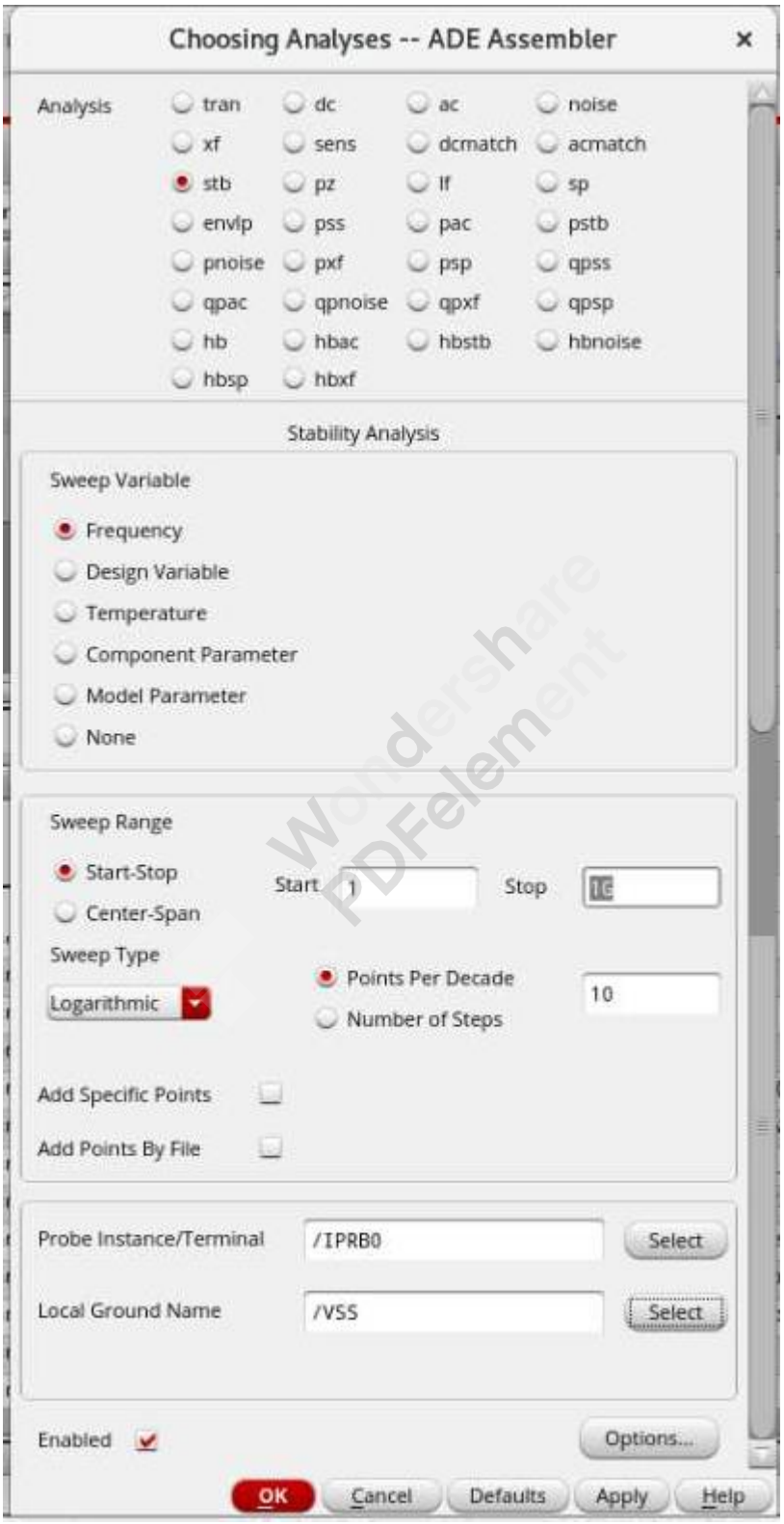


Figure – 4.71: "stb" selection and its parameters

Click on the "**downward arrow**" just before the test name as shown in Figure – 4.72 to go back to the ADE Explorer.

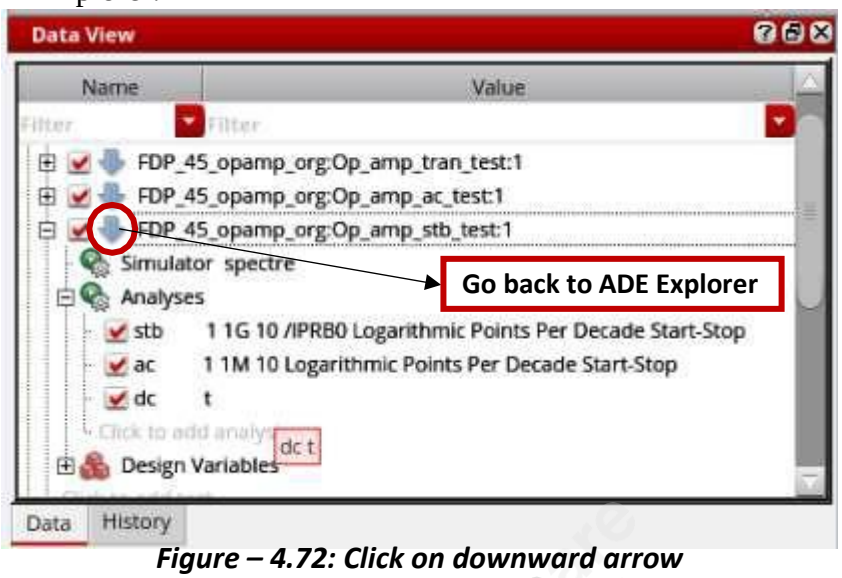


Figure – 4.72: Click on downward arrow

The ADE Explorer window pops up as shown in Figure – 4.73.

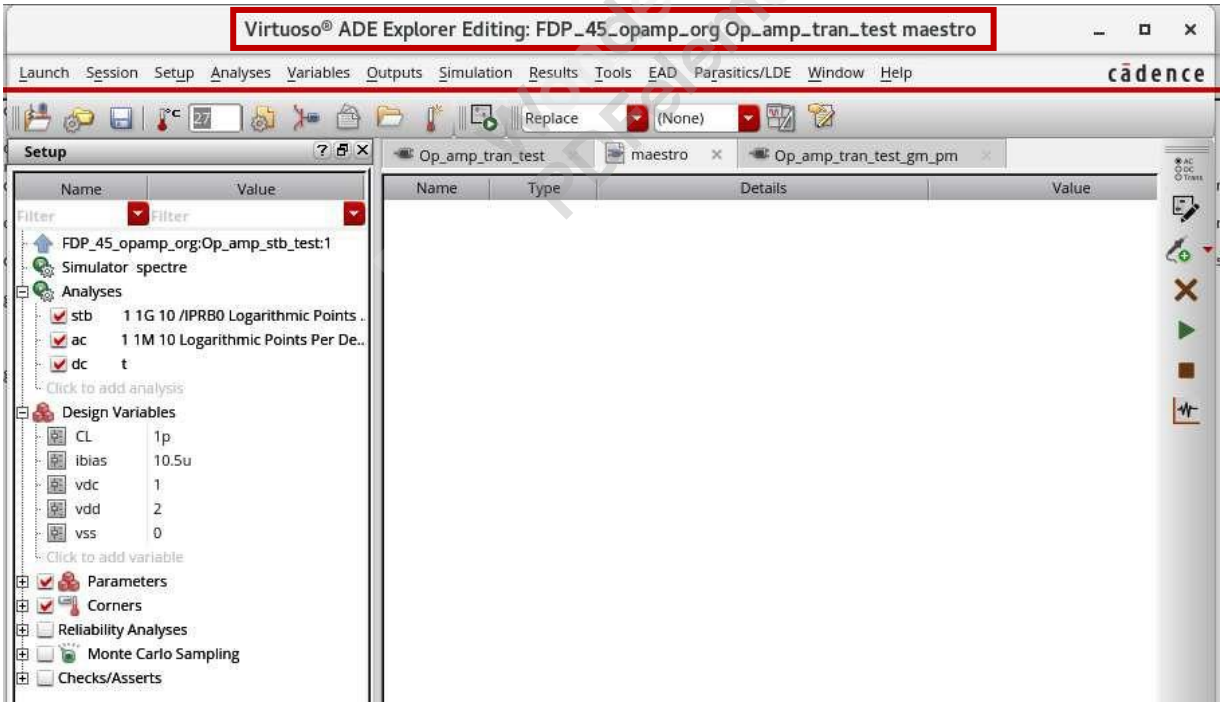


Figure – 4.73: ADE Explorer

Click on "**Simulation □ Netlist and Run**" similar to the selection in ADE L window. After the simulation, select "**Results □ Direct Plot □ Main Form**" as shown in Figure – 4.74. The "**Direct Plot Form**" pops up as shown in Figure – 4.75. Click on "**Stability Summary**" to print the values of Gain Margin and Phase Margin. Click on "**Plot**" to plot the graph.

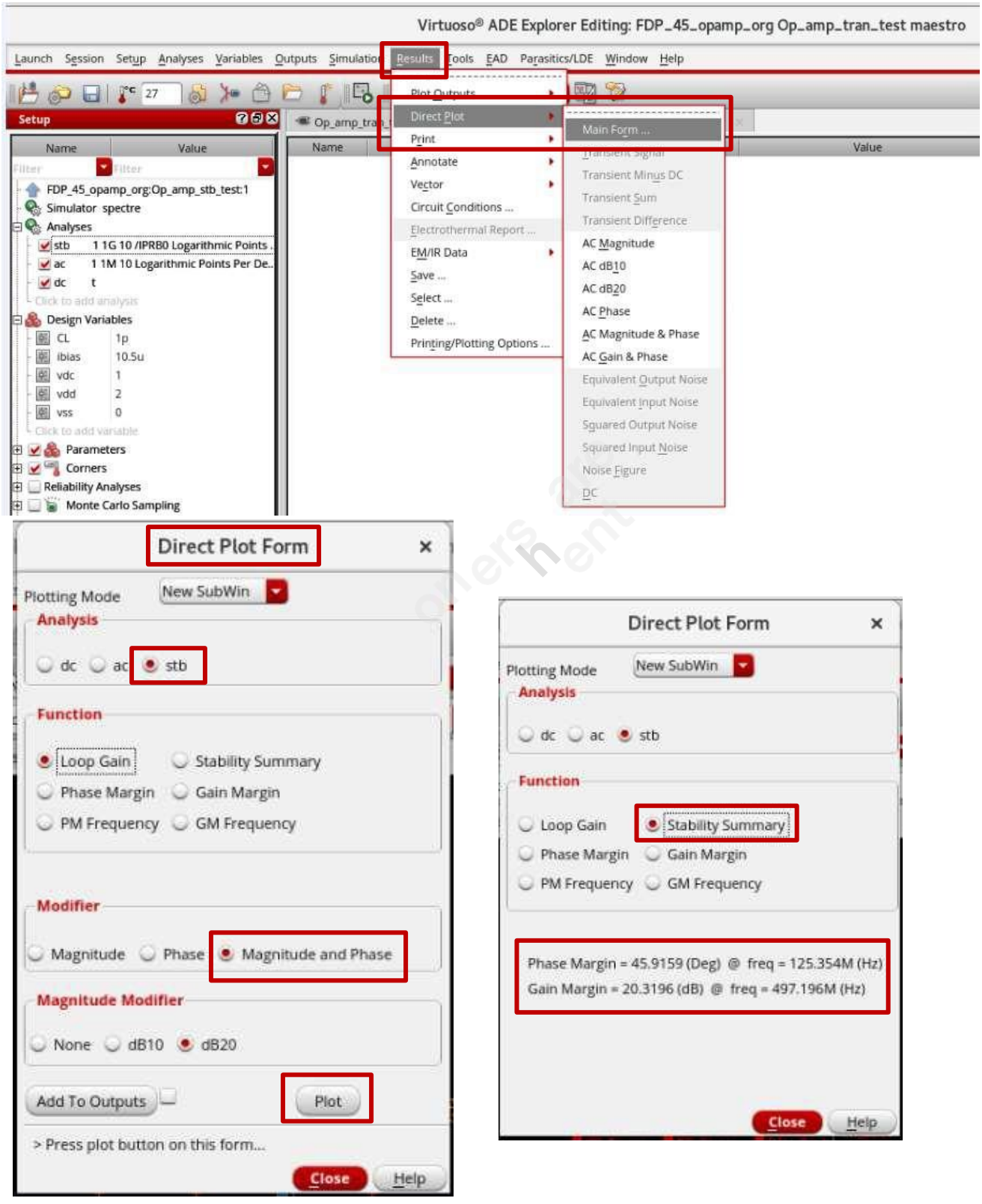


Figure – 4.75: Direct Plot Form and Stability Summary with Gain Margin and PhaseMargin

LAYOUT:

Follow the techniques demonstrated in Lab – 01 to open the Layout Editor, import the devices from the Schematic, place the devices as per the requirement and complete the routing. The completed layout can be seen as shown in Figure – 4.76.

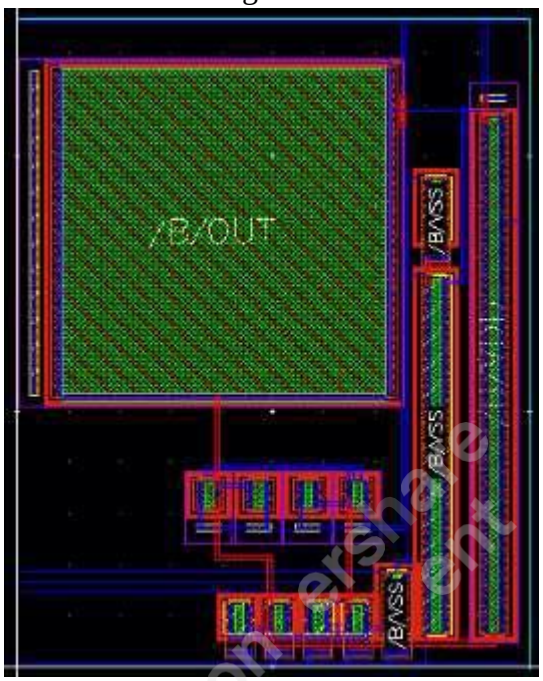


Figure – 4.76: Layout for 2 – Stage Operational Amplifier

DRC:

To check for the DRC violations, browse the "**assura_tech.lib**" file, select "**Assura □ Run DRC**", verify the Layout Design Source, mention a "**Run Name**", select "**Technology □ gpdk045**" and click on "**OK**" as demonstrated in Lab – 01.

LVS:

To check for the LVS violations, select "**Assura □ Run LVS**", verify the Schematic Design Source and the Layout Design Source, mention a "**Run Name**", select "**Technology □ gpdk045**" and click on "**OK**" as demonstrated in Lab – 01.

QRC:

To extract the Parasitics, select "**Assura □ Quantus**", select "**Technology □ gpdk180**", "**Output □ Extracted View**" from the "**Setup**" option, select "**Extraction Type □ RC**" and "**Ref Node □ VSS**" from the "**Extraction**" and click on "**OK**" as demonstrated in Lab – 01. The result can be checked from the Library Manager.

BACKANNOTATION:

Import the parasitics into the Test Schematic and re-run the simulation to check their impact by calculating the delay elements as demonstrated in Lab – 01.

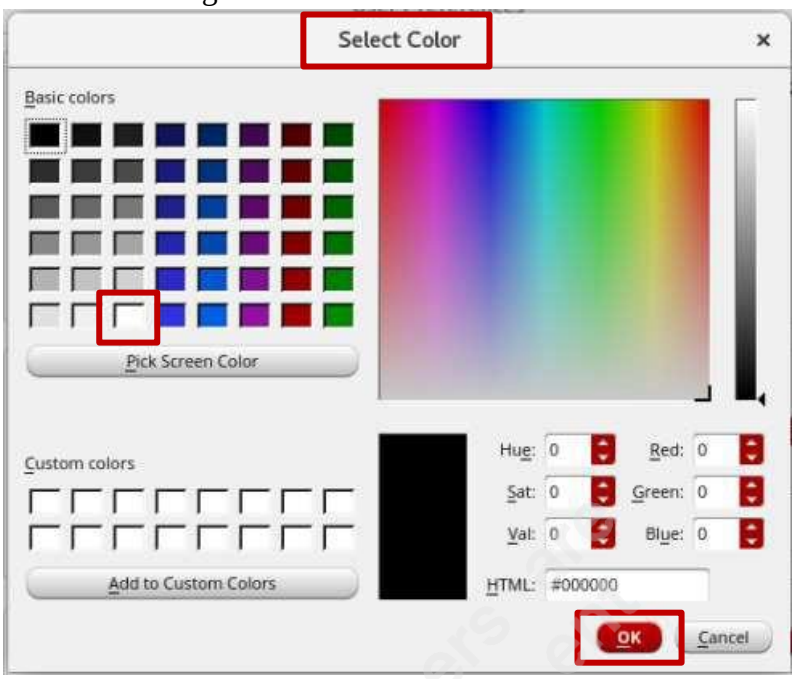
APPENDIX – 1: CHANGING BACKGROUND COLOR IN VIRTUOSO SCHEMATICEDITOR

To change the background color for the Virtuoso Schematic Editor, select "**Options □ User Preferences**" from the Command Interpreter Window as shown in Figure – a.

The "**User Preferences**" window pops up as shown in Figure – b.

Figure – b: User Preferences window

Click on "**Default Editor Background Color**" as shown in Figure – b. The "**Select Color**" window pops up as shown in Figure – c.



Select the Screen Color of interest and click on "**OK**" as shown in Figure – c.

Figure – d: Updated User Preferences window

The updated "**User Preferences**" window can be seen as shown in Figure – d. Click on "**Apply**" and click on "**OK**".

The updated "**Virtuoso Schematic Editor L Editing**" window can be seen as shown in Figure – e.

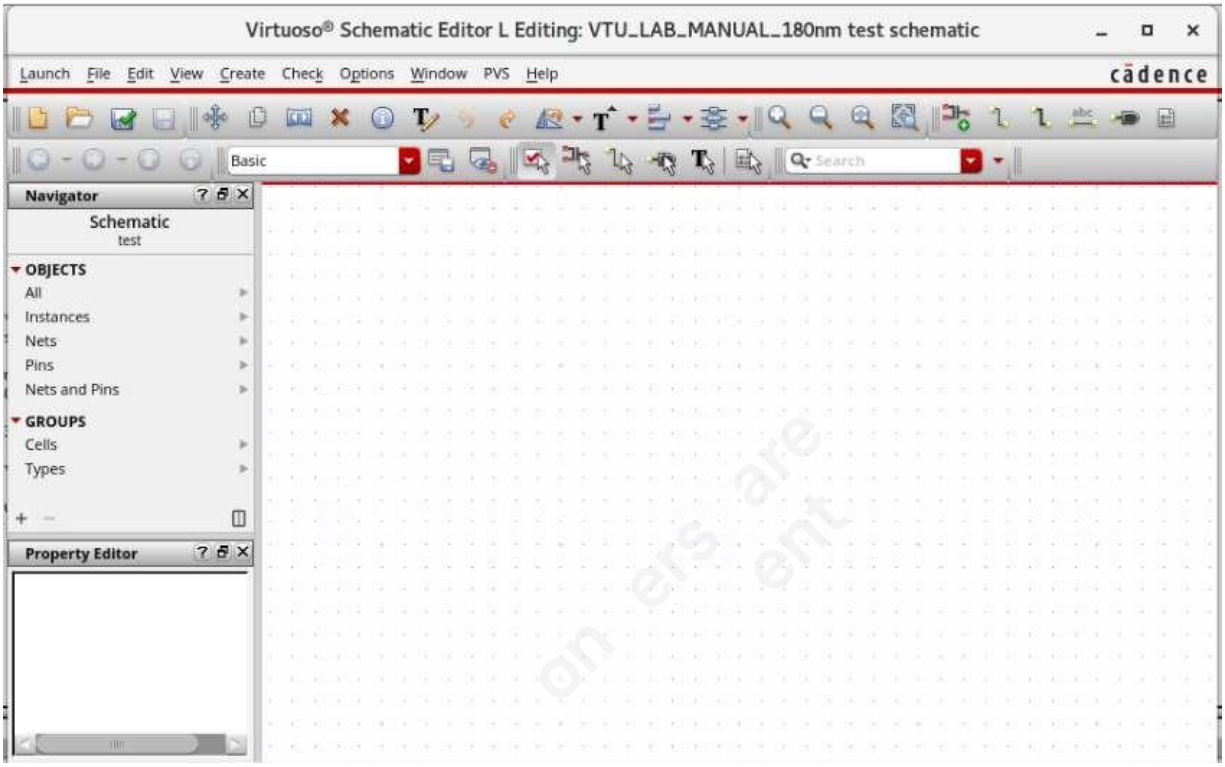


Figure – e: Updated Virtuoso Schematic Editor L Editing window

PART B

ASIC-Digital Design Flow

Introduction

PROCEDURE FOR CREATING DIGITAL SIMULATION USING VERILOG AND CADENCE DIGITAL TOOL

Open Terminal Window and use the following commands

Procedure for Simulation:

1**. Change directory to Cadence_Digital_labs/Workarea/Inverter using commands:**

>**csh**
>**source /home/install/cshrc**

2. Type the programs design code and testbench code using editor in workarea folder and save

Create the design file (Verilog module) using editor gedit
>**gedit filename.v**
The file name can be changed with respect to the experiments.

Create the testbench file (Verilog module) using editor gedit
>**gedit tb_filename.v**
The testbench file name can be changed with respect to the design file.

3. **Compile the source Descriptions: (i) Compile the Inverter description with the -messages option:**
>ncvlog –mess (**name of the file**) (**name of the testbench file**)

Example: >**ncvlog -mess inv.v tb_inv.v**

Note: Check out for error and warnings. If any, then go back to text editor and edit and the compile

4. **Elaborate the top level Design using the command:**
>**ncelab -mess -access +rwc (name of the testbench file without file type)**

Example: **>ncelab -mess –access +rwc tb_inv //Do not include file type (.v) for testbench file**

5. **Simulate the Top-Level Design in Non GUI mode**
>**ncsim -mess -gui (name of the testbench file)**

Example:>ncsim -mess -gui tb_inv

Now a console and Design Browser windows of Simvision are opened and click on the waveform button in the toolbar to send the selected objects to waveform window.

Waveform Window opens and Press run to run the simulation for a time period specified in the time field.

Procedure for Synthesis using Cadence Tool

The Verilog code, technology library and the constraint files are input to the logical synthesis. The tool will generate the gate level netlist and gives the output as: gatelevel schematic and the report of area, power and timing.

6. Create a file named Contraints_sdc.sdc

The timing constraints are defined in this file. Example of one such file is as shown –

```
create_clock -name clk -period 10 -waveform {0 5} [get_ports "clk"]
set_clock_transition -rise 0.1 [get_clocks "clk"]
set_clock_transition -fall 0.1 [get_clocks "clk"]
set_clock_uncertainty 1.0 [get_ports "clk"]
set_input_delay -max 1.0 [get_ports "A"] -clock [get_clocks "clk"]
set_input_delay -max 1.0 [get_ports "B"] -clock [get_clocks "clk"]
set_output_delay -max 1.0 [get_ports "sum"] -clock [get_clocks "clk"]
```

There are three different parts in the constraint file:

a. Clock definition and clock constraints –

```
Clock definition
create_clock -name clk -period 10 -waveform {0 5} [get_ports "clock"]
Clock rise time
set_clock_transition -rise 0.1 [get_clocks " clk "]
 Clock fall time
set_clock_transition -fall 0.1 [get_clocks " clk "]
Uncertainties of Clock
set_clock_uncertainty 1.0 [get_ports " clk "]
```

b. Input port timing constraints –

```
set_input_delay -max 1.0 [get_ports " A"]  -clock  [get_clocks " clk "]
Input port delay
set_input_delay  -max 1.0 [get_ports " B"]  -clock  [get_clocks " clk "]
```

Input port delay

c. Output port timing constraints –

Output port delay

set_output_delay -max 1.0 [get_ports " sum "] -clock [get_clocks " clk "]

The port names that are used in the constraint file (bolded) must match with the names that are used in the Verilog program of the main design module. The constraints are defined for all the ports in the design.

7. Synthesis the top level design

The verilog file to be synthesized must be copied into this directory.

Copy genus.tcl file that created for simulation in desktop into user directory

genus.tcl file contains

```
read_lib
read_hdl  counter_8bit.v   (example)
elaborate
read_sdc  top.sdc
syn_generic
write_hdl
syn_map
write_hdl

write_hdl>syn_netlist.v
write_sdc>syn_sdc.sdc

gui_show
report  area>area_rep
report  gates>gates_rep
report   timing>timing_rep
```

Synthesize the top level design

Open the genus.tcl file in gedit
>gedit genus.tcl

Change the filename in read_hdl

```
read_hdl  {filename.v}      // filename.v is the Design code file
          {inv.v}             // for example inv.v as filename.v
```

Then invoke genus command
>genus -f genus.tcl

Script file contains the Verilog RTL code, standard library file for a particular technology,

Introduction to Constraints and Synthesis Commands

SDC

Synopsys Design Constraint (SDC) format is used to specify the design intent, including the timing and area constraints of the design.

1. create_clock

Usage

create_clock [-add] [-name <clock_name>] -period <value> [-waveform <edge_list>]
<targets>

Options

-add: Adds clock to a node with an existing clock
-name <clock_name>: Clock name of the created clock
-period <value>: Speed of the clock in terms of clock period
-waveform <edge_list>: List of edge values
<targets>: List or collection of targets

Description

Defines a clock. If the -name option is not used; the clock name is the same as the first target in the list or collection. The clock name is used to refer to the clock in other commands.

The -period option specifies the clock period. It is also possible to use this option to specify a frequency to define the clock period. This can be done by using -period option followed by either <frequency>MHz or "<frequency> MHz".

The -waveform option specifies the rising and falling edges (duty cycle) of the clock, and is specified as a list of two time values: the first rising edge and the next falling edge. The rising edge must be within the range [0, period]. The falling edge must be within one clock period of the rising edge. The waveform defaults to (0, period/2).

If a clock with the same name is already assigned to a given target, the create_clock command will overwrite the existing clock. If a clock with a different name exists on the given target, the create_clock command will be ignored unless the -add option is used. The -add option can be used to assign multiple clocks to a pin or port.

If the target of the clock is internal (i.e. not an input port), the source latency is zero by default. If a clock is on a path after another clock, then it blocks or overwrites the previous clock from that point forward.

Example

```
# Create a simple 10ns with clock with a 60% duty cycle
create_clock -period 10 -waveform {0 6} -name clk [get_ports clk]

# Create a clock with a falling edge at 2ns, rising edge at 8ns,
# falling at 12ns, etc.
create_clock -period 10 -waveform {8 12} -name clk [get_ports clk]
```

1. get_clocks

   Usage

   get_clocks [-nocase] [-nowarn] <filter>

   Options

   -nocase: Specifies the matching of node names to be case-insensitive
   -nowarn: Do not issue warnings messages about unmatched patterns

<filter>: Valid destinations (string patterns are matched using Tcl string matching)

Description
Returns a collection of clocks in the design. When used as an argument to another command, such as the -from or -to options of set_multicycle_path, each node in the clock represents all nodes driven by the clocks in the collection.

2. get_ports
   Usage
   get_ports [-nocase] [-nowarn] <filter>
   Options
   -nocase: Specifies case-insensitive node name matching
   -nowarn: Do not issue warnings messages about unmatched patterns
   <filter>: Valid destinations (string patterns are matched using Tcl string matching)

Description
Returns a collection of ports (design inputs and outputs) in the design. The filter for the collection is a Tcl list of wildcards, and must follow standard Tcl

3. set_clock_uncertainty
   Usage
   set_clock_uncertainty [-add] [-fall_from <fall_from_clock>] [-fall_to <fall_to_clock>] [-from <from_clock>] [-hold] [-rise_from <rise_from_clock>] [-rise_to <rise_to_clock>] [-setup] [-to <to_clock>] <uncertainty>

   Options
   -add: Specifies that this assignment is an addition to the clock uncertainty derived by derive_clock_uncertainty call
   -fall_from <fall_from_clock>: Valid destinations (string patterns are matched using Tcl string matching)
   -fall_to <fall_to_clock>: Valid destinations (string patterns are matched using Tcl string matching)
   -from <from_clock>: Valid destinations (string patterns are matched using Tcl string matching)
   -hold: Specifies the uncertainty value (applies to clock hold or removal checks)
   -rise_from <rise_from_clock>: Valid destinations (string patterns are matched using Tcl string matching)
   -rise_to <rise_to_clock>: Valid destinations (string patterns are matched using Tcl string matching)
   -setup: Specifies the uncertainty value (applies to clock setup or recovery checks) (default)
   -to <to_clock>: Valid destinations (string patterns are matched using Tcl string matching)
   <uncertainty>: Uncertainty

Description
Specifies clock uncertainty or skew for clocks or clock-to-clock transfers. You can specify uncertainty separately for setup and hold, and can specify separate rising and falling clock transitions. The setup uncertainty is subtracted from the data required time for each applicable path, and the hold uncertainty is added to the data required time for each applicable path.

The values for the -from, -to, and similar options are either collections or a Tcl list of wildcards used to create collections of appropriate types. The values used must follow standard Tcl

When -add option is used, clock uncertainty assignment is treated as an addition to the value calculted by derive_clock_uncertainty command for a particular clock transfer.

Note that when -add option is not used and derive_clock_uncertainty is called, user specified clock uncertainty assignment will take priority. When derive_clock_uncertainty command is not used, specifying -add option to set_clock_uncertainty command will not have any effect.

4. set_input_delay
   Usage
   set_input_delay [-add_delay] -clock <name> [-clock_fall] [-fall] [-max] [-min] [-reference_pin <name>] [-rise] [-source_latency_included] <delay> <targets>
   Options
   -add_delay: Add to existing delays instead of overriding them
   -clock <name>: Clock name
   -clock_fall: Specifies that input delay is relative to the falling edge of the clock
   -fall: Specifies the falling input delay at the port
   -max: Applies value as maximum data arrival time
   -min: Applies value as minimum data arrival time
   -reference_pin <name>: Specifies a port in the design to which the input delay is relative
   -rise: Specifies the rising input delay at the port
   -source_latency_included: Specifies that input delay includes added source latency
   <delay>: Time value
   <targets>: List of input port type objects

Description
Specifies the data arrival times at the specified input ports relative the clock specified by the -clock option. The clock must refer to a clock name in the design.
Input delays can be specified relative to the rising edge (default) or falling edge (-clock_fall) of the clock.

If the input delay is specified relative to a simple generated clock (a generated clock with a single target), the clock arrival times to the generated clock are added to the data arrival time.

Input delays can be specified relative to a port (-reference_pin) in the clock network. Clock arrival times to the reference port are added to data arrival times. Non-port reference pins are not supported.

Input delays can already include clock source latency. By default the clock source latency of the related clock is added to the input delay value, but when the -source_latency_included option is specified, the clock source latency is not added because it was factored into the input delay value.

The maximum input delay (-max) is used for clock setup checks or recovery checks and the minimum input delay (-min) is used for clock hold checks or removal checks. If only -min or -max (or neither) is specified for a given port, the same value is used for both.

Separate rising (-rise) and falling (-fall) arrival times at the port can be specified. If only one of -rise and -fall are specified for a given port, the same value is used for both.

By default, set_input_delay removes any other input delays to the port except for those with the same -clock, -clock_fall, and -reference_pin combination. Multiple input delays relative to different clocks, clock edges, or reference pins can be specified using the -add_delay option.

5. set_input_transition
   Usage
   set_input_transition [-clock <name>] [-clock_fall] [-fall] [-max] [-min] [-rise]
   <transition> <ports>
   Options
   -clock <name>: Clock name
   -clock_fall: Specifies that input delay is relative to the falling edge of the clock
   -fall: Specifies the falling output delay at the port
   -max: Applies value as maximum data required time
   -min: Applies value as minimum data required time
   -rise: Specifies the rising output delay at the port
   <transition>: Time value
   <ports>: Collection or list of input or bidir ports

Description
It only affects PrimeTime analysis or HardCopy II devices. If you set this constraint in TimeQuest the constraint is written out to the SDC file when you call write_sdc

6. set_output_delay
   Usage
   set_output_delay [-add_delay] -clock <name> [-clock_fall] [-fall] [-max] [-min]
   [-reference_pin <name>] [-rise] [-source_latency_included] <delay> <targets>
   Options
   -add_delay: Add to existing delays instead of overriding them
   -clock <name>: Clock name
   -clock_fall: Specifies output delay relative to the falling edge of the clock
   -fall: Specifies the falling output delay at the port
   -max: Applies value as maximum data required time
   -min: Applies value as minimum data required time
   -reference_pin <name>: Specifies a port in the design to which the output delay is relative
   -rise: Specifies the rising output delay at the port
   -source_latency_included: Specifies input delay already includes added source latency
   <delay>: Time value
   <targets>: Collection or list of output ports

Description
Specifies the data required times at the specified output ports relative the clock specified by the -clock option. The clock must refer to a clock name in the design.
Output delays can be specified relative to the rising edge (default) or falling edge (-clock_fall) of the clock.

If the output delay is specified relative to a simple generated clock (a generated clock with a single target), the clock arrival times to the generated clock are added to the data required time. Output delays can be specified relative to a port (-reference_pin) in the clock network. Clock arrival times to the reference port are added to the data required time. Non-port reference pins are not supported.
Output delays can include clock source latency.

By default the clock source latency of the related clock is added to the output delay value, but when the -source_latency_included option is specified, the clock
source latency is not added because it was factored into the output delay value.
The maximum output delay (-max) is used for clock setup checks or recovery checks and the minimum output delay (-min) is used for clock hold checks or removal checks. If only one of -min and -max (or neither) is specified for a given port, the same value is used for both.

Separate rising (-rise) and falling (-fall) required times at the port can be specified. If only one of -rise and -fall are specified for a given port, the same value is used for both.

By default, set_output_delay removes any other output delays to the port except for those with the same -clock, -clock_fall, and -reference_pin combination. Multiple output delays relative to different clocks, clock edges, or reference pins can be specified using the -add_delay option.

TCL Command:
read_sdc
Usage
read_sdc [-hdl] <file_name>
Options
-hdl: Read SDC commands embedded in HDL
<file_name>: Name of the SDC file

Description
Reads an SDC file with all current constraints and exceptions. If an SDC file is specified, read_sdc only reads that SDC file. If the -hdl option is specified, read_sdc only reads SDC commands that were embedded in HDL.

If no arguments are specified, read_sdc reads the default SDC files along with any SDC commands that were embedded in HDL. If one or more SDC_FILE assignments exists in the QSF, read_sdc reads all of them in order. Otherwise, read_sdc reads the file <revision>.sdc if it exists.

Example
project_new test
create_timing_netlist
Read SDC commands from test_constraints.sdc
read_sdc test_constraints.sdc

Experiment-1
Title: 4-bit Up/Down Asynchronous Reset Counter

Problem Statement: To write Verilog code for asynchronous counter circuit and its test bench for verification, observe the waveform and synthesize the code with technological library with given Constraints.

Objective:

Write Verilog code for 4-bit up/down asynchronous reset counter and carry out the following:

a. Verify the functionality using test bench

b. Synthesize the design by setting area and timing constraint. Obtain the gate level netlist, find the critical path and maximum frequency of operation. Record the area requirement in terms of number of cells required and properties of each cell in terms of driving strength, power and area requirement.

c. Perform the above for 32-bit up/down counter and identify the critical path, delay of critical path, and maximum frequency of operation, total number of cells required and total area.

TOOL REQUIRED: Cadence Tool

THEORY:

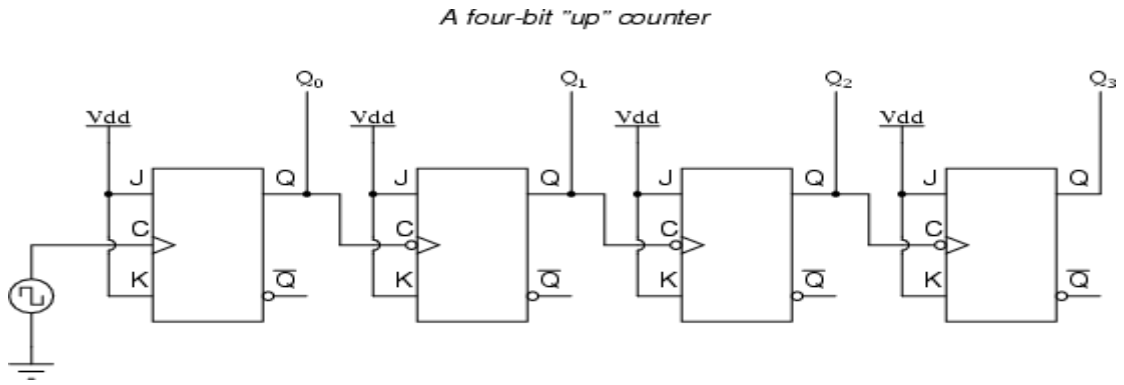
Asynchronous counters are those whose output is free from the clock signal. Because the flip flops in asynchronous counters are supplied with different clock signals, there may be delay in producing output. The required number of logic gates to design asynchronous counters is very less. So they are simple in design. Another name for Asynchronous counters is “Ripple counters”.

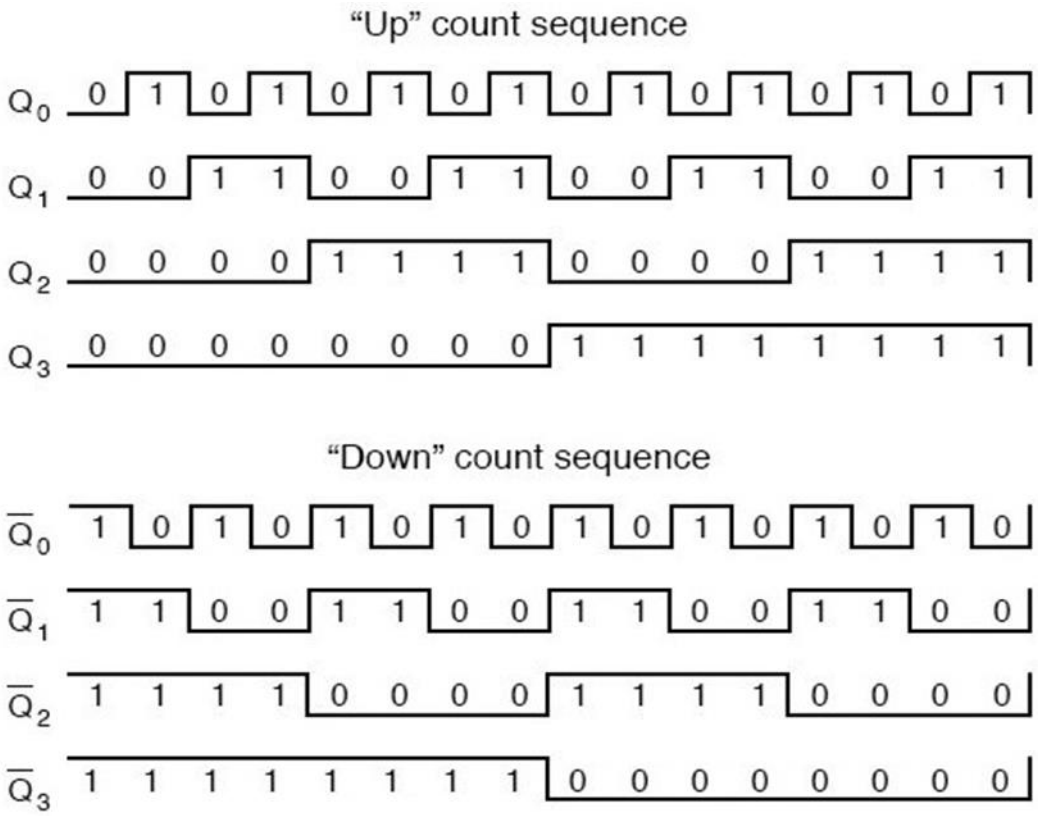
A ripple counter is an asynchronous counter where only the first flip-flop is clocked by an external clock. All subsequent flip-flops are clocked by the output of the preceding flip-flop. Asynchronous counters are also called ripple-counters because of the way the clock pulse ripples it way through the flip-flops.

4-bit up/down Counter with loadable count

A counter is a sequential circuit which is used to count clock pulses. In other words, a counter is a sequential machine constructed with the help of flip-flops & changes its state according to state diagram. A 4-bit Up/Down counter counts the clock pulses in ascending as well as descending order and recycle again from its initial value.

A graphical schematic for a 4-bit up/down counter is depicted in the given figure.





Timing diagram of Asynchronous Counter

/*Asynchronous counter program Verilog code for design file name counter-4bit.v*/

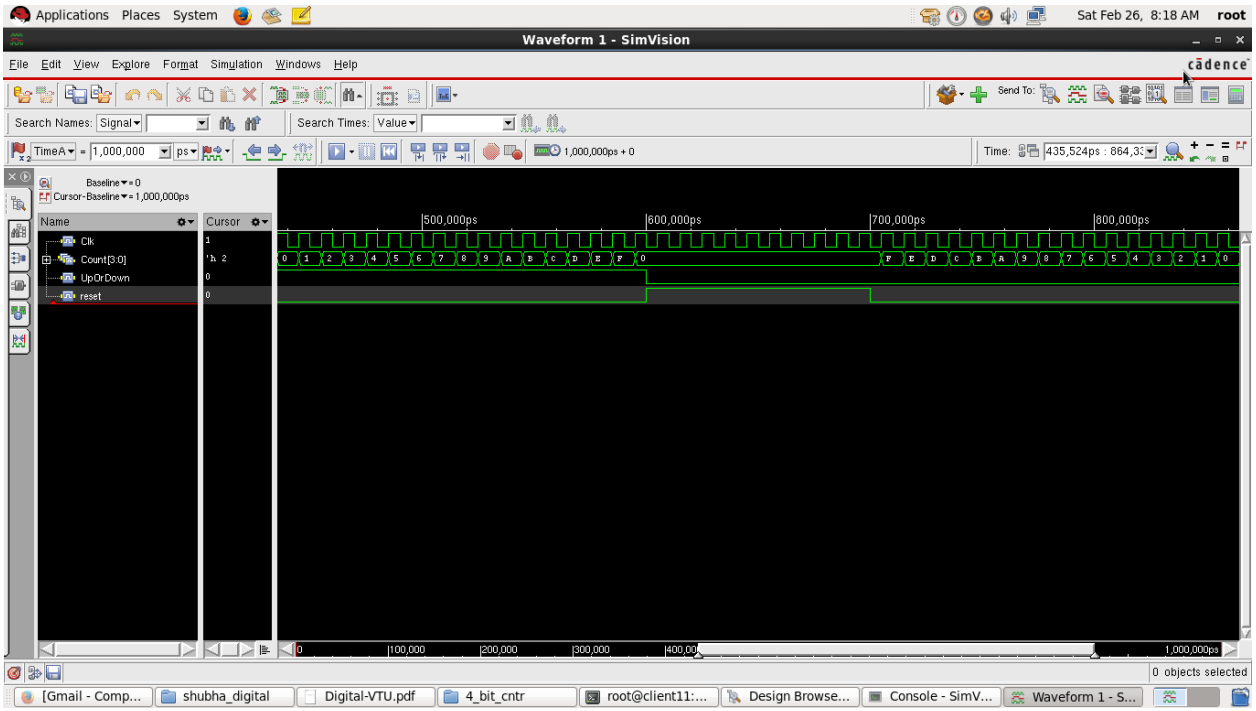
```
timescale 1ns/1ps                    // Defining a Timescale for Precision
module counter(clk,rst,m,count);     // Defining Module and Port List
input clk,rst,m;                     // Defining Inputs
output  [3:0]count;                  // Defining 32-bit Output as Reg type
 reg [3 : 0] count = 0;
always@(posedge clk or posedge rst)  // The Block is executed when begin
                                     // EITHER of positive edge of clock or Neg Edge of Rst arrives
                                     // Both are independent events
begin
if(rst==1)
count<=0;
else if(m==1)
count<=count+1;
else
count<=count-1;
end
endmodule
```

/*Asynchronous counter program Verilog code for testbench-file name tb_counter-4bit.v*/

```
`timescale 1ns/1ps                   //Creating Time Scale as in Source Code
module counter_test;                 //Defining Module Name without Port List
   reg clk;                          //Defining I/P as Registers [to Hold Values]
   reg rst; /                        /Defining O/P as Wires [To Probe Waveforms] initial
```

```
    reg m;
   wire [3:0] count;
counter counter1(clk,m,rst, count);          //Instantiation of Source Code always
initial clk = 0;
    always #2 clk = ~clk;

    initial begin                            //Up-Down counting is allowed at posedge clk
        // Apply Inputs
        rst = 0;#5;
        m = 0;
        #200;
        m= 1;                                //Condition for Up-Count
      #200;
        rst = 1;
        m = 0;                               //Condition for Down-Count
        #100;
        rst = 0;
end
initial
#2000 $finish;
endmodule
```

Waveform:



b) Synthesize the design using Constraints and Analyze reports, critical path and Max Operating Frequency.

Step 1: Getting Started

- Make sure you close out all the Incisive tool windows first.

- Synthesis requires three files as follows,
    - Liberty Files (.lib)
    - Verilog/VHDL Files (.v or .vhdl or .vhd)
    - SDC (System Design Constraint) File (.sdc)

Step 2 : Creating an SDC File

- In your terminal type “gedit counter_top.sdc” to create an SDC File if you do not have one.
- The SDC File must contain the following commands;

i. create_clock -name clk -period 2 -waveform {0 1} [get_ports "clk"]

ii. set_clock_transition -rise 0.1 [get_clocks "clk"]

iii.set_clock_transition -fall 0.1 [get_clocks "clk"]

iv.set_clock_uncertainty 0.01 [get_ports "clk"]

v. set_input_delay -max 1.0 [get_ports "rst"] -clock [get_clocks "clk"]

vi.set_output_delay -max 1.0 [get_ports "count"] -clock [get_clocks "clk"]

i→ Creates a Clock named “clk” with Time Period 2ns and On Time from t=0 to t=1.

ii, iii → Sets Clock Rise and Fall time to 100ps.

iv → Sets Clock Uncertainty to 10ps.

v, vi → Sets the maximum limit for I/O port delay to 1ps.

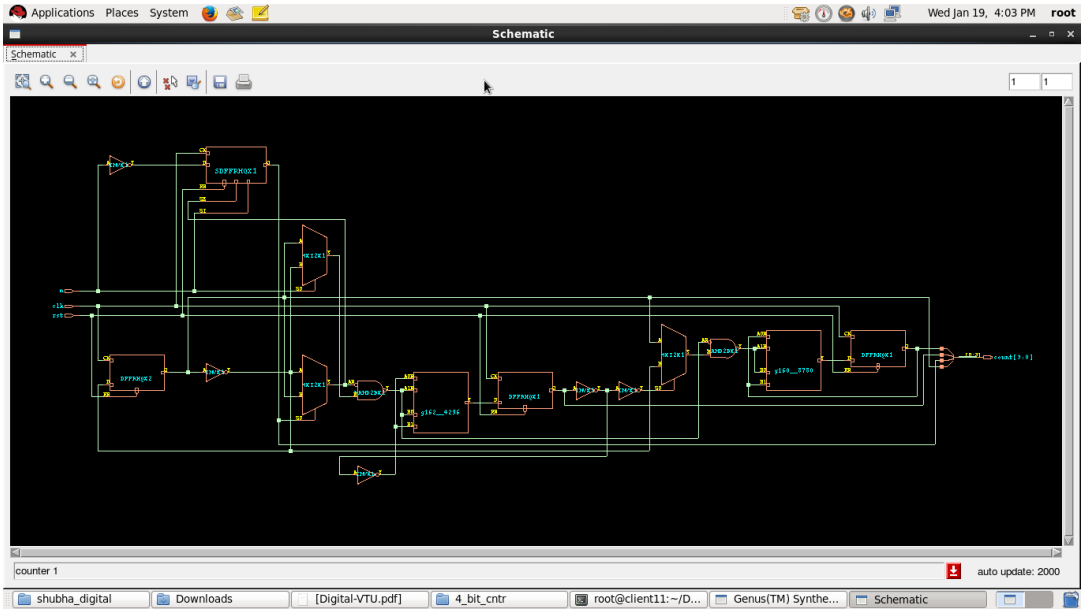
Step 3 : Performing Synthesis

The Liberty files are present in the below path,

/home/install/FOUNDRY/digital/<Technology_Node_number>nm/dig/lib/

Synthesis RTL Schematic

The procedure for Simulation and Synthesis remains same as mentioned earlier



PROGRAM:

/*32-bit up/down asynchronous reset counter Verilog code-file name counter.v*/

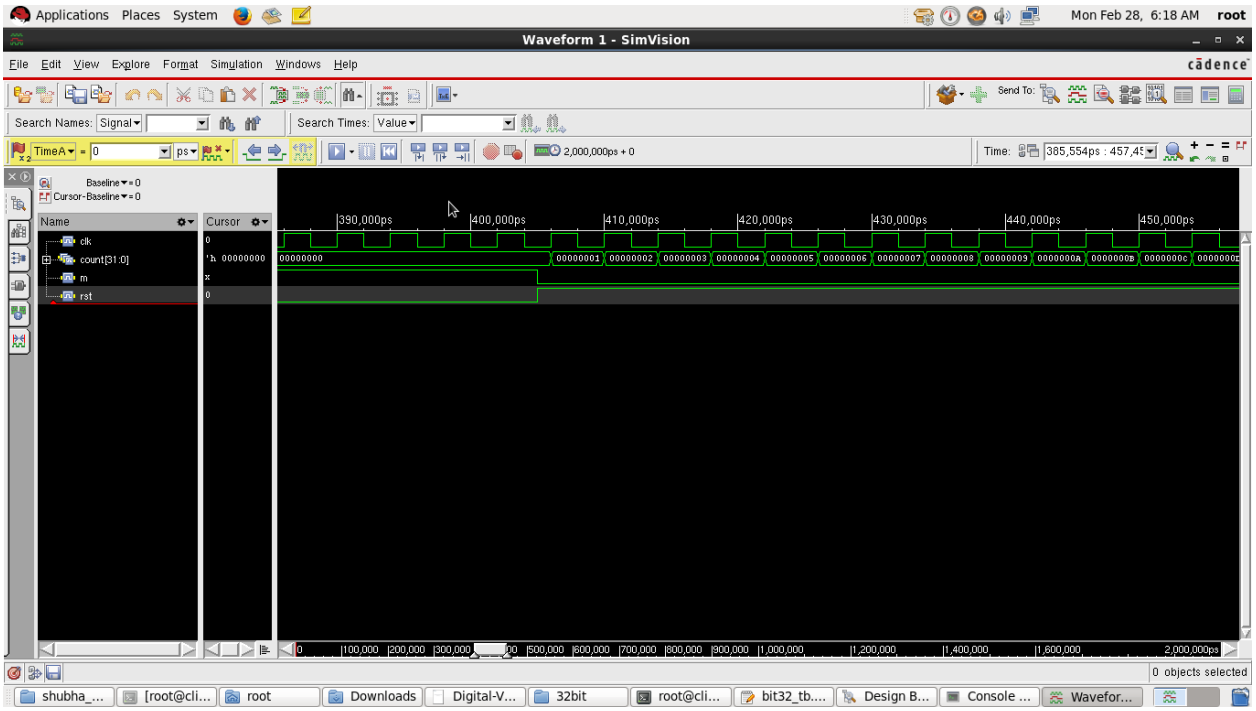
```
timescale 1ns/1ps                    // Defining a Timescale for Precision
module counter(clk,rst,m,count);     // Defining Module and Port List
input clk,rst,m;                     // Defining Inputs
output  [31:0]count;                 // Defining 32-bit Output as Reg type
 reg [31 : 0] count = 0;
always@(posedge clk or posedge rst)  // The Block is executed when begin
                                     // EITHER of positive edge of clock or Neg Edge of Rst arrives
                                     // Both are independent events
begin
if(rst==1)
count<=0;
else if(m==1)
count<=count+1;
else
count<=count-1;
end
endmodule
```

/* **32-bit up/down asynchronous reset counter for testbench-file name tb_counter.v*/**

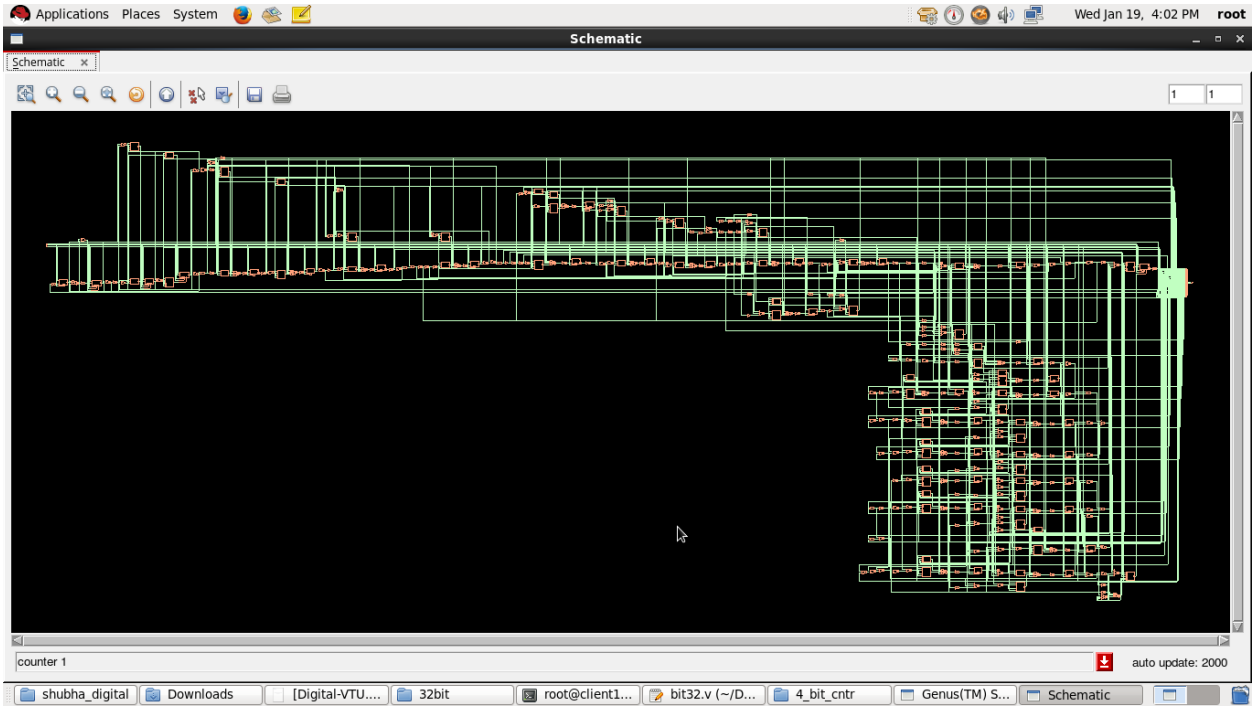
```
`timescale 1ns/1ps                   //Creating Time Scale as in Source Code
module counter_test;                 //Defining Module Name without Port List
   reg clk;                          //Defining I/P as Registers [to Hold Values]
   reg rst; /                        /Defining O/P as Wires [To Probe Waveforms] initial
   reg m;

   wire [31:0] count;
counter counter1(clk,m,rst, count);   //Instantiation of Source Code always
initial clk = 0;
   always #2 clk = ~clk;
```

```
    initial begin                          //Up-Down counting is allowed at posedge clk
        // Apply Inputs
        rst = 0;#5;
        m = 0;
        #200;
        m= 1;                              //Condition for Up-Count
      #200;
        rst = 1;

        m = 0;                             //Condition for Down-Count

        #100;
        rst = 0;
end
initial
#2000 $finish;
endmodule
```

Waveform:



Synthesis RTL Schematic:



RESULT: Verilog code for an Asynchronous 4-bit Counter circuit and its test bench for verification is written, the waveform is observed and the code is synthesized with the technological library and is verified.

Experiment-2

Title: 4-bit Adder

Problem Statement: To develop the source code for 4-bit Adder by using VERILOG and obtain the simulation and its test bench for verification, observe the waveform, synthesize the code with technological library with given Constraints to generate into a netlist and place and route and implement it.

Objectives:

Write Verilog code for 4-bit Adder and verify its functionality using test bench.

Synthesize the design by setting proper constraints and obtain the net list.

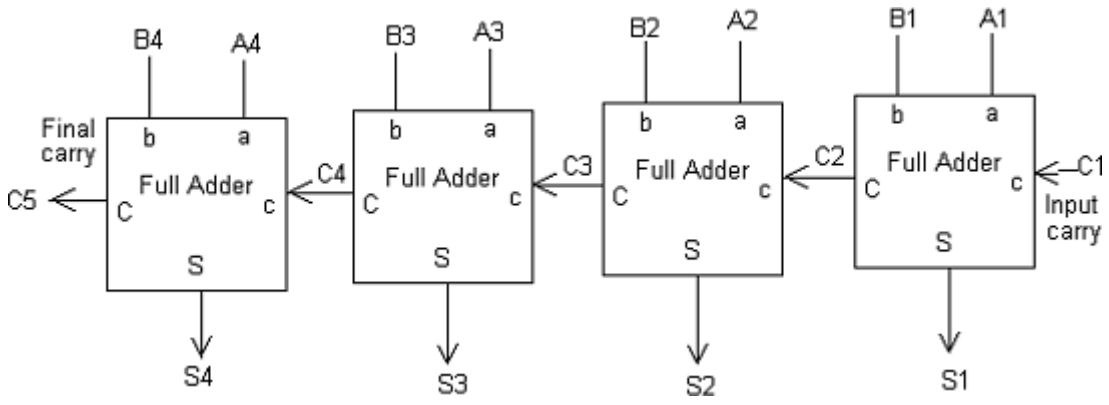
From the report generated, identify critical path, maximum delay, total number of cells, power requirement and total area required. Change the constraints and obtain optimum synthesis results.

TOOL REQUIRED: Cadence Tool

THEORY:

4-bit Adder is a combinatorial circuit (not clocked, does not have any memory and feedback) adding every bit position of the operands in the same time. Thus it is requiring number of bit-Adders (full adders + 1 half adder) equal to the number of bits to be added. The 4-bit adder is constructed by cascading full adders

(FA) blocks in series. One full adder is responsible for the addition of two binary digits at any stage of the ripple carry. The carryout of one stage is fed directly to the carry-in of the next stage.



PROGRAM:

Three Codes are written for implementation of 4-bit Adder

- fa.v → Single Bit 3-input Full Adder [Sub-Module / Function]
- fa_4bit.v → Top Module for Adding 4-bit inputs
- fa_test.v → Test bench code for testing of 4-bit Adder design

/* fa.v → Single Bit 3-input Full Adder */

```
module full_adder( A,B,CIN,S,COUT);
input A,B,CIN;
output   S,COUT;
assign S = A^B^CIN;
assign COUT = (A&B) | (CIN&(A^B));
endmodule
```

/* fa_4bit.v → Top Module for Adding 4-bit Inputs */

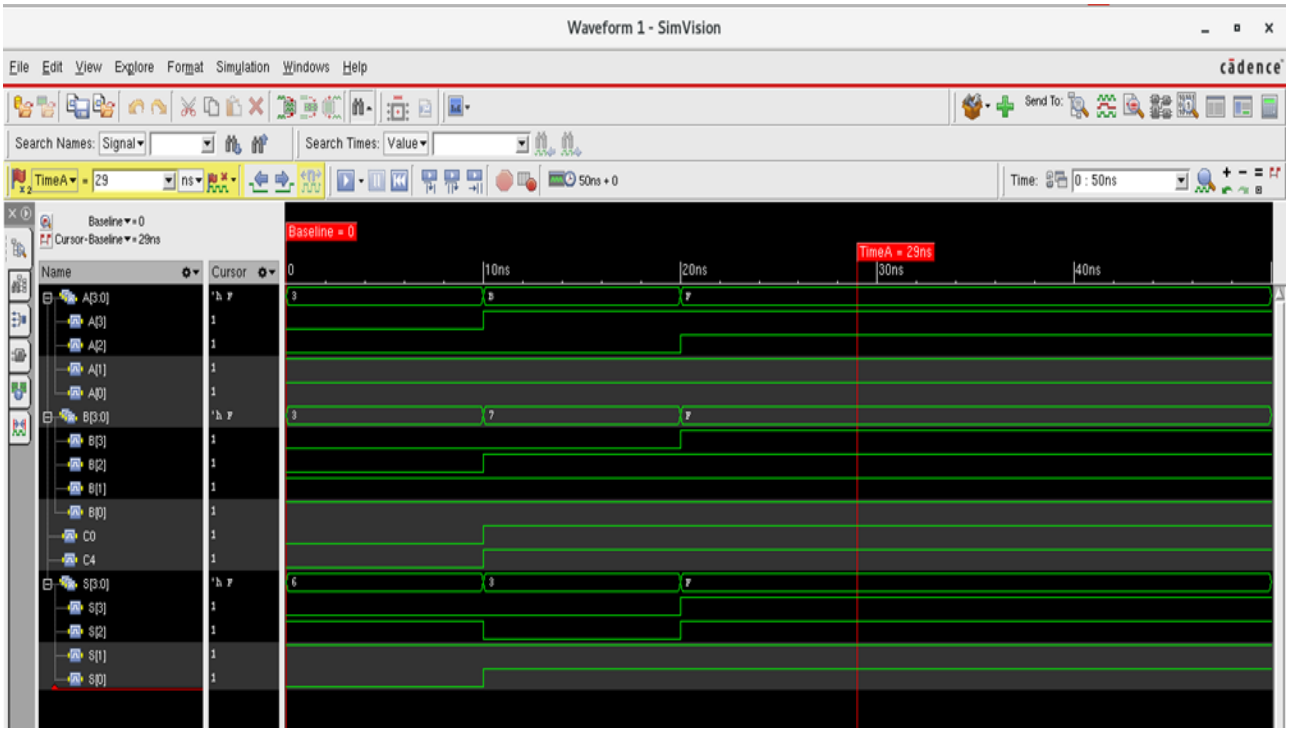
```
module four_bit_adder(A,B,C0,S,C4);
input [3:0] A,[3:0] B,C0;
output [3:0] S,C4;
wire C1,C2,C3;

full_adder fa0 (A[0],B[0],C0,S[0],C1);
full_adder fa1 (A[1],B[1],C1,S[1],C2);
full_adder fa2 (A[2],B[2],C2,S[2],C3);
full_adder fa3 (A[3],B[3],C3,S[3],C4);
endmodule
```

/* Test Bench – fa_test.v */

```
module test_4_bit;
reg [3:0] A;
reg [3:0] B; reg C0;
wire [3:0] S; wire C4;
four_bit_adder dut(A,B,C0,S,C4);
initial begin
A = 4'b0011;B=4'b0011;C0 = 1'b0; #10;
```

```
A = 4'b1011;B=4'b0111;C0 = 1'b1; #10;
A = 4'b1111;B=4'b1111;C0 = 1'b1; #10;
end
initial
#50 $finish;
Endmodule
```

Waveform :



b) Synthesis and Report/Output Analysis

Step 1: Getting Started

- Make sure you close out all the Incisive tool windows first.
- Synthesis requires three files as follows,
  - Liberty Files (.lib)
  - Verilog/VHDL Files (.v or .vhdl or .vhd)
  - SDC (System Design Constraint) File (.sdc)

Step 2 : Creating an SDC File

- As the Full Adder Program does not contain any Clock, SDC can be skipped as an Input if you wish to, else you could include only Area Constraints in form of commands during synthesis such as, set_dont_use *XL (Indicating not to use Larger Cells)

Step 3 : Performing Synthesis

- The Liberty files are present in the below path,
- /home/install/FOUNDRY/digital/<Technology_Node_number>nm/dig/lib/
- The Available technology nodes are 180nm ,90nm and 45nm. In the terminal, initialise the tools with the following commands if a new terminal is being used.

- csh
- source /home/install/cshrc

- The tool used for Synthesis is “Genus”. Hence, type “genus -gui” to open the tool.
- The Following are commands to proceed,

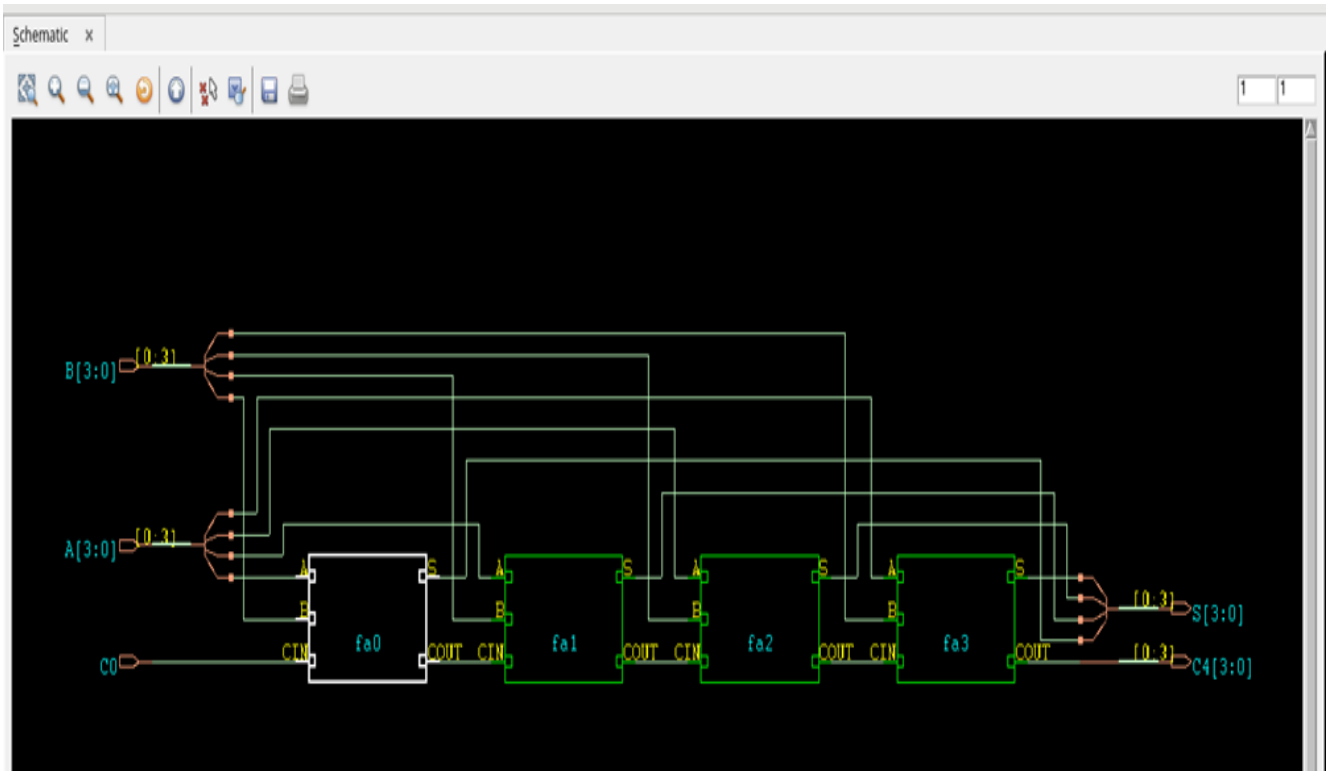
```
1. read_libs /home/install/FOUNDRY/digital/90nm/dig/lib/slow.lib
2. read_hdl {fa.v fa_4bit.v} //Reading multiple Verilog Files
3. elaborate
4. set_top_module four_bit_adder                    //Differentiating Top & Sub Module
5. set_dont_use *XL                                 //Dont Use Cells with High Driving Strength
6. synthesize -to_mapped -effort medium             //Performing Synthesis Mapping and Optimization
7. report_timing -unconstrained > counter_timing.rep
                                                    // Generates Timing report for worst datapath and dumps into file
                                                    //-Unconstrained is to be given as no timing constraints are given
8. report_area > counter_area.rep                   //Generates Synthesis Area report and dumps into a file
9. report_power > counter_power.rep
   //Generates Power Report [Pre-Layout]

10. write_hdl > counter_netlist.v                   //Creates readable Netlist File.
11. write_sdc > counter_sdc.sdc                     //Creates Block Level SDC
```

Commands 1-5 are intended for Synthesis process while 6-10 for Generating reports and Outputs.

Note 1:-

1. The Cells given in the netlist can be checked in the .lib files for their properties.
2. The Max Operating Frequency does not apply for Purely Combinational Circuit.

Synthesis RTL Schematic:



Some Common Constraints are given below for

Common SDC Constraints

Operating conditions
- set_operating_conditions

Wire-load models
- set_wire_load_mode
- set_wire_load_model
- set_wire_load_selection_group

Environmental
- set_drive
- set_driving_cell
- set_load
- set_fanout_load
- set_input_transition
- set_port_fanout_number

Design rules
- set_max_capacitance
- set_max_fanout
- set_max_transition

Timing
- create_clock
- create_generated_clock
- set_clock_latency
- set_clock_transition
- set_disable_timing
- set_propagated_clock
- set_clock_uncertainty
- set_input_delay
- set_output_delay

Exceptions
- set_false_path
- set_max_delay
- set_multicycle_path

Power
- set_max_dynamic_power
- set_max_leakage_power

Note-2:-

1. Tabulate Area, Power and Timing Constraints using any of the SDC Constraints as instructed.
2. Make sure, during synthesis the Report File Names are changed so that the latest reports do not overwrite the earlier ones.

RESULT: Verilog code for the 4-bit Adder circuit and its test bench for verification is written, the waveform is observed and the code is synthesized with the technological library and is verified.

Experiment No: 3

Title: UART

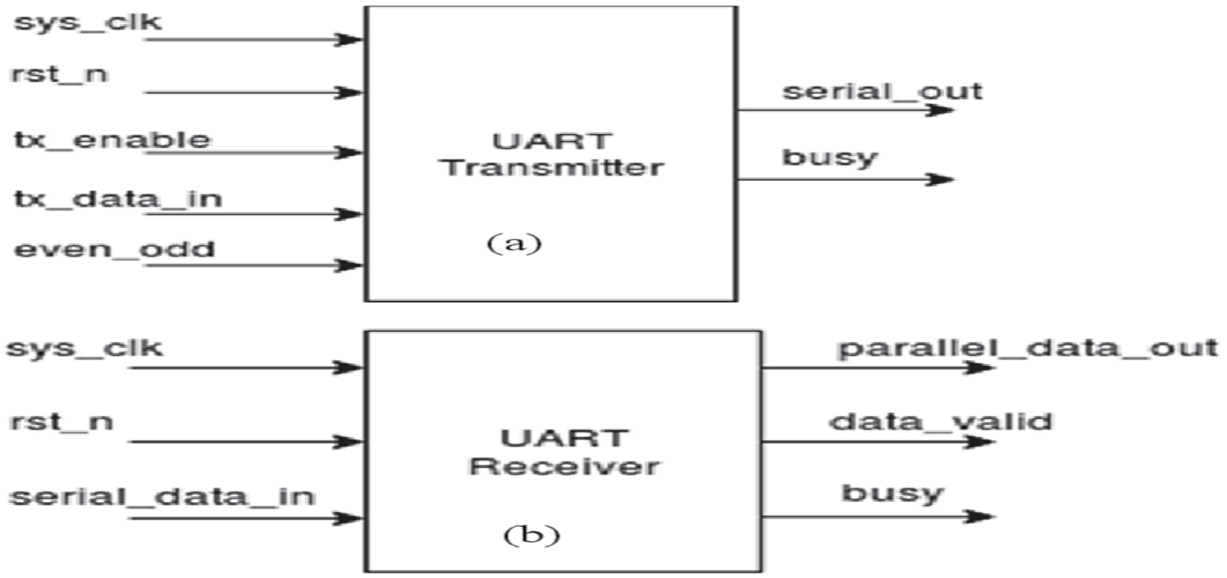
Problem statement: To develop the source code for UART by using VERILOG and obtain the simulation and its test bench for verification, observe the waveform, synthesize the code with technological library with given Constraints to generate into a netlist and place and route and implement it.

Objective:

a) To Verify the Functionality using test Bench
b) Synthesize Design using constraints
c) Tabulate Reports using various Constraints
d) Identify Critical Path and calculate Max Operating Frequency

Theory:

UART (Universal Asynchronous Receiver and Transmitter) is a serial communication protocol. Basically this protocol is used to permit short distance, low cost and reliable full duplex communication. It is used to exchange data between the processor and peripherals. The UART has three main parts- receiver, transmitter and baud rate generator. Baud rate generator generates the clock frequency for transmitter and receiver at a specific baud rate. The UART design has used a baud rate of 115200 bps with 25 MHz Clock.



g. 1 Interface diagram of UART Protocol (a) Transmitter of UART (b) Receiver o

Creating a Workspace:
Create a new sub-Directory for the Design and open a terminal from the Sub-Directory.

a)Functional Verification using Test Bench

```
// Source code – Transmitter:
// This code contains the UART Transmitter. This transmitter is able to transmit 8 bits of serial data, one
// start bit, one stop bit, and no parity bit.
// When transmit is complete o_Tx_done will be driven high for one clock cycle.
//
// Set Parameter CLKS_PER_BIT as follows:
// CLKS_PER_BIT = (Frequency of i_Clock)/(Frequency of UART)
// Example: 25 MHz Clock, 115200 baud UART
// (25000000)/(115200) = 217
```

```
module uart_tx
  #(parameter CLKS_PER_BIT)
  (
   input       i_Clock,
   input       i_Tx_DV,
   input [7:0] i_Tx_Byte,
   output      o_Tx_Active,
   output reg  o_Tx_Serial,
   output      o_Tx_Done
   );

  parameter s_IDLE         = 3'b000;
  parameter s_TX_START_BIT = 3'b001;
  parameter s_TX_DATA_BITS = 3'b010;
  parameter s_TX_STOP_BIT  = 3'b011;
  parameter s_CLEANUP      = 3'b100;

  reg [2:0]    r_SM_Main     = 0;
  reg [7:0]    r_Clock_Count = 0;
  reg [2:0]    r_Bit_Index   = 0;
  reg [7:0]    r_Tx_Data     = 0;
  reg          r_Tx_Done     = 0;
  reg          r_Tx_Active   = 0;

  always @(posedge i_Clock)
    begin

      case (r_SM_Main)
        s_IDLE :
          begin
            o_Tx_Serial   <= 1'b1;         // Drive Line High for Idle
            r_Tx_Done     <= 1'b0;
            r_Clock_Count <= 0;
            r_Bit_Index   <= 0;

            if (i_Tx_DV == 1'b1)
              begin
                r_Tx_Active <= 1'b1;
                r_Tx_Data   <= i_Tx_Byte;
                r_SM_Main   <= s_TX_START_BIT;
              end
            else
              r_SM_Main <= s_IDLE;
          end // case: s_IDLE


        // Send out Start Bit. Start bit = 0
        s_TX_START_BIT :
          begin
            o_Tx_Serial <= 1'b0;
```

```
    // Wait CLKS_PER_BIT-1 clock cycles for start bit to finish
    if (r_Clock_Count < CLKS_PER_BIT-1)
      begin
        r_Clock_Count <= r_Clock_Count + 1;
        r_SM_Main     <= s_TX_START_BIT;
      end
    else
      begin
        r_Clock_Count <= 0;
        r_SM_Main     <= s_TX_DATA_BITS;
      end
  end // case: s_TX_START_BIT


// Wait CLKS_PER_BIT-1 clock cycles for data bits to finish
s_TX_DATA_BITS :
  begin
    o_Tx_Serial <= r_Tx_Data[r_Bit_Index];

    if (r_Clock_Count < CLKS_PER_BIT-1)
      begin
        r_Clock_Count <= r_Clock_Count + 1;
        r_SM_Main     <= s_TX_DATA_BITS;
      end
    else
      begin
        r_Clock_Count <= 0;

        // Check if we have sent out all bits
        if (r_Bit_Index < 7)
          begin
            r_Bit_Index <= r_Bit_Index + 1;
            r_SM_Main   <= s_TX_DATA_BITS;
          end
        else
          begin
            r_Bit_Index <= 0;
            r_SM_Main   <= s_TX_STOP_BIT;
          end
      end
  end // case: s_TX_DATA_BITS


// Send out Stop bit.  Stop bit = 1
s_TX_STOP_BIT :
  begin
    o_Tx_Serial <= 1'b1;

    // Wait CLKS_PER_BIT-1 clock cycles for Stop bit to finish
    if (r_Clock_Count < CLKS_PER_BIT-1)
```

```
            begin
              r_Clock_Count <= r_Clock_Count + 1;
              r_SM_Main     <= s_TX_STOP_BIT;
            end
          else
            begin
              r_Tx_Done     <= 1'b1;
              r_Clock_Count <= 0;
              r_SM_Main     <= s_CLEANUP;
              r_Tx_Active   <= 1'b0;
            end
        end // case: s_Tx_STOP_BIT


      // Stay here 1 clock
      s_CLEANUP :
        begin
          r_Tx_Done <= 1'b1;
          r_SM_Main <= s_IDLE;
        end

      default :
        r_SM_Main <= s_IDLE;

    endcase
  end

  assign o_Tx_Active = r_Tx_Active;
  assign o_Tx_Done   = r_Tx_Done;

endmodule// Source code – Receiver :

// This file contains the UART Receiver. This receiver is able to receive 8 bits of serial data, one start bit,
// one stop bit, and no parity bit. When receive is complete o_rx_dv will be driven high for one clock cycle.
// Set Parameter CLKS_PER_BIT as follows:
// CLKS_PER_BIT = (Frequency of i_Clock)/(Frequency of UART)
// Example: 25 MHz Clock, 115200 baud UART
// (25000000)/(115200) = 217


    module uart_rx
      #(parameter CLKS_PER_BIT)
      (
       input        i_Clock,
       input        i_Rx_Serial,
       output       o_Rx_DV,
       output [7:0] o_Rx_Byte
       );
      parameter s_IDLE         = 3'b000;
      parameter s_RX_START_BIT = 3'b001;
      parameter s_RX_DATA_BITS = 3'b010;
```

```
parameter s_RX_STOP_BIT  = 3'b011;
parameter s_CLEANUP      = 3'b100;

reg          r_Rx_Data_R = 1'b1;
reg          r_Rx_Data   = 1'b1;

reg [7:0]    r_Clock_Count = 0;
reg [2:0]    r_Bit_Index   = 0; //8 bits total
reg [7:0]    r_Rx_Byte     = 0;
reg          r_Rx_DV       = 0;
reg [2:0]    r_SM_Main     = 0;

// Purpose: Double-register the incoming data.
// This allows it to be used in the UART RX Clock Domain.
// (It removes problems caused by metastability)
always @(posedge i_Clock)
  begin
    r_Rx_Data_R <= i_Rx_Serial;
    r_Rx_Data   <= r_Rx_Data_R;
  end

   // Purpose: Control RX state machine
always @(posedge i_Clock)
  begin

    case (r_SM_Main)
      s_IDLE :
        begin
          r_Rx_DV       <= 1'b0;
          r_Clock_Count <= 0;
          r_Bit_Index   <= 0;

          if (r_Rx_Data == 1'b0)          // Start bit detected
            r_SM_Main <= s_RX_START_BIT;
          else
            r_SM_Main <= s_IDLE;
        end

      // Check middle of start bit to make sure it's still low
      s_RX_START_BIT :
        begin
          if (r_Clock_Count == (CLKS_PER_BIT-1)/2)
            begin
              if (r_Rx_Data == 1'b0)
                begin
                  r_Clock_Count <= 0;  // reset counter, found the middle
                  r_SM_Main     <= s_RX_DATA_BITS;
                end
              else
                r_SM_Main <= s_IDLE;
            end
```

```
    else
      begin
        r_Clock_Count <= r_Clock_Count + 1;
        r_SM_Main     <= s_RX_START_BIT;
      end
  end // case: s_RX_START_BIT


// Wait CLKS_PER_BIT-1 clock cycles to sample serial data
s_RX_DATA_BITS :
  begin
    if (r_Clock_Count < CLKS_PER_BIT-1)
      begin
        r_Clock_Count <= r_Clock_Count + 1;
        r_SM_Main     <= s_RX_DATA_BITS;
      end
    else
      begin
        r_Clock_Count          <= 0;
        r_Rx_Byte[r_Bit_Index] <= r_Rx_Data;

        // Check if we have received all bits
        if (r_Bit_Index < 7)
          begin
            r_Bit_Index <= r_Bit_Index + 1;
            r_SM_Main   <= s_RX_DATA_BITS;
          end
        else
          begin
            r_Bit_Index <= 0;
            r_SM_Main   <= s_RX_STOP_BIT;
          end
      end
  end // case: s_RX_DATA_BITS


// Receive Stop bit.  Stop bit = 1
s_RX_STOP_BIT :
  begin
    // Wait CLKS_PER_BIT-1 clock cycles for Stop bit to finish
    if (r_Clock_Count < CLKS_PER_BIT-1)
      begin
        r_Clock_Count <= r_Clock_Count + 1;
        r_SM_Main     <= s_RX_STOP_BIT;
      end
    else
      begin
        r_Rx_DV       <= 1'b1;
        r_Clock_Count <= 0;
        r_SM_Main     <= s_CLEANUP;
      end
```

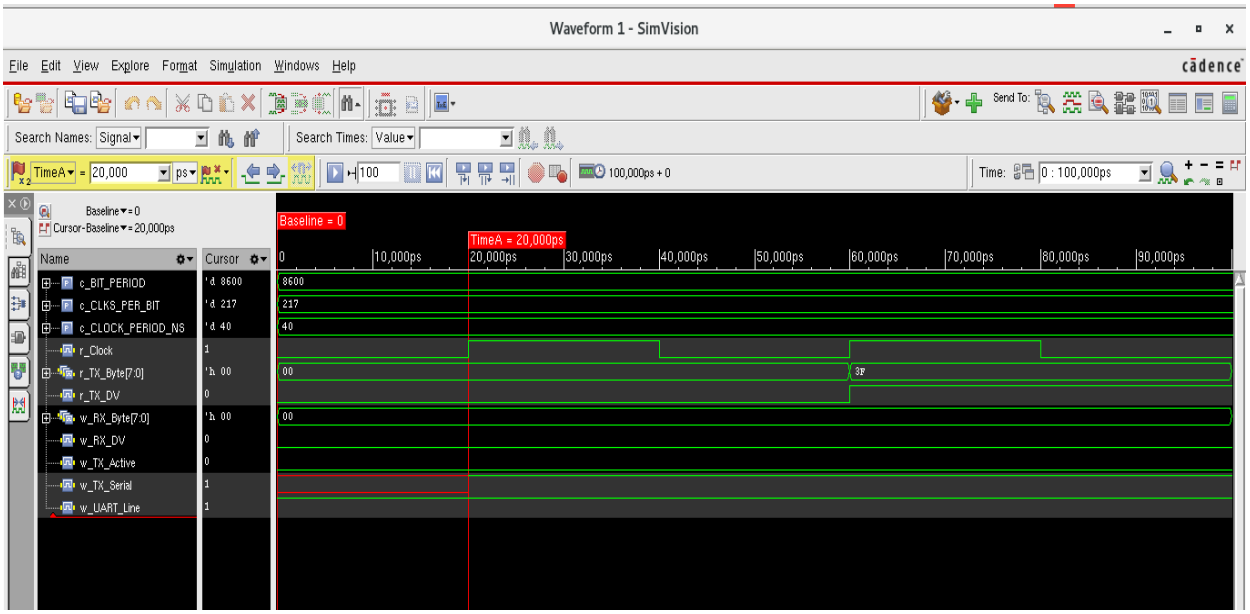
```
          end // case: s_RX_STOP_BIT


        // Stay here 1 clock
        s_CLEANUP :
          begin
            r_SM_Main <= s_IDLE;
            r_Rx_DV   <= 1'b0;
          end


        default :
          r_SM_Main <= s_IDLE;

      endcase
    end

  assign o_Rx_DV   = r_Rx_DV;
  assign o_Rx_Byte = r_Rx_Byte;

endmodule // uart_rx
```

Test bench:

```
// This testbench will exercise the UART RX.
// It sends out byte 0x37, and ensures the RX receives it correctly.

`timescale 1ns/10ps

`include "uart_tx.v"
`include "uart_rx.v"

module UART_TB();

// Testbench uses a 25 MHz clock
// Want to interface to 115200 baud UART
// 25000000 / 115200 = 217 Clocks Per Bit.

parameter c_CLOCK_PERIOD_NS = 40;
parameter c_CLKS_PER_BIT= 217;
parameter c_BIT_PERIOD    = 8600;

reg r_Clock = 0; reg r_TX_DV = 0;
wire w_TX_Active, w_UART_Line;
wire w_TX_Serial;
reg [7:0] r_TX_Byte = 0;
wire [7:0] w_RX_Byte;

UART_RX #(.CLKS_PER_BIT(c_CLKS_PER_BIT)) UART_RX_Inst
(.i_Clock(r_Clock),
.i_RX_Serial(w_UART_Line),
.o_RX_DV(w_RX_DV),
```

```
.o_RX_Byte(w_RX_Byte)
);

UART_TX #(.CLKS_PER_BIT(c_CLKS_PER_BIT)) UART_TX_Inst
(.i_Clock(r_Clock),
.i_TX_DV(r_TX_DV),
.i_TX_Byte(r_TX_Byte),
.o_TX_Active(w_TX_Active),
.o_TX_Serial(w_TX_Serial),
.o_TX_Done()
);

// Keeps the UART Receive input high (default) when  UART transmitter is not active
assign w_UART_Line = w_TX_Active ? w_TX_Serial : 1'b1;

always
#(c_CLOCK_PERIOD_NS/2) r_Clock <= !r_Clock;
// Main Testing:
Initial begin
// Tell UART to send a command (exercise TX) @(posedge r_Clock);
@(posedge r_Clock); r_TX_DV <= 1'b1;
r_TX_Byte <= 8'h3F; @(posedge r_Clock); r_TX_DV <= 1'b0;
end
endmodule
```

Waveform:



b) Synthesize the Design

1. read_libs /home/install/FOUNDRY/digital/90nm/dig/lib/slow.lib
2. read_hdl {uart_tx.v / uart_rx.v} //Choose any one
3. elaborate
4. read_sdc constraints_top.sdc //Reading Top Level SDC
5. synthesize -to_mapped -effort medium //Performing Synthesis Mapping and Optimization
6. report_timing > uart_timing.rep

//Generates Timing report for worst datapath and dumps into file

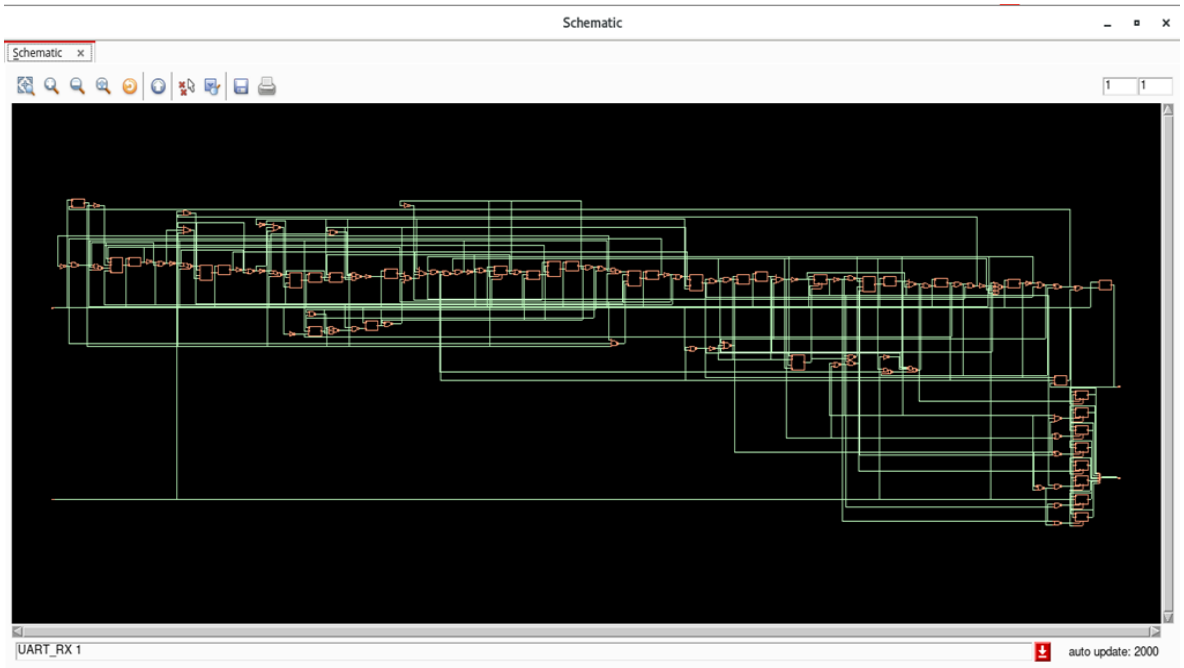
7. report_area > uart_area.rep //Generates Synthesis Area report and dumps into a file
8. report_power > uart_power.rep //Generates Power Report [Pre-Layout]
9. write_hdl > uart_netlist.v //Creates readable Netlist File
10. write_sdc > uart_sdc.sdc //Creates Block Level SDC

Synthesis RTL Schematic:

Constraints file for Synthesis:

```
create_clock -name clk -period 10 -waveform {0 5} [get_ports "clk"]
set_clock_transition -rise 0.1 [get_clocks "clk"]
set_clock_transition -fall 0.1 [get_clocks "clk"]
set_clock_uncertainty 1.0 [get_ports "clk"]
set_input_delay -max 1.0 [get_ports "rst"] -clock [get_clocks "clk"]
set_input_delay -max 1.0 [get_ports "din"] -clock [get_clocks "clk"]
set_output_delay -max 1.0 [get_ports "q"] -clock [get_clocks "clk"]
set_output_delay -max 1.0 [get_ports "qb"] -clock [get_clocks "clk"]
```

Synthesis RTL Schematic:



Note:-

1. Tabulate Area, Power and Timing Constraints using any of the SDC Constraints as instructed.
2. Make sure, during synthesis the Report File Names are changed so that the latest reports do not overwrite the earlier ones.

RESULT: Verilog code for the UART circuit and its test bench for verification is written, the waveform is observed and the code is synthesized with the technological library with given constraints and is verified.

Experiment-4
Title: 32-bit ALU

Problem statement: To develop the source code for 32-bit ALU by using VERILOG and obtain the simulation and its test bench for verification, observe the waveform, synthesize the code with technological library with given Constraints to generate into a netlist, place and route and implement it.

Objectives:
a) To Verify the Functionality using Test Bench
b) Synthesize and compare the results using if and case statements
c) Identify Critical Path and constraints

Creating a Work space:
Create a new sub-Directory for the Design and open a terminal from the Sub-Directory.

Theory:
An Arithmetic Logic Unit (ALU) is a digital circuit used to perform arithmetic and logic operations. It represents the fundamental building block of the central processing unit (CPU) of a computer. Most of the operations of a CPU are performed by one or more ALUs, which load data from input registers.

A 32-bit ALU is a combinational circuit taking two 32-bit data words A and B as inputs, and producing a 32-bit output Y by performing a specified arithmetic or logical function on the A and B inputs.

Arithmetic Logic Unit (ALU) using these simple logic gates AND, OR, NOT, XOR and other components. The ALU will take in two 32-bit values, and two control lines. Depending on the value of the control lines, the output will be the addition, subtraction, bitwise AND or bitwise OR of the inputs. The Fig. 1 shows the block diagram of 32-bit ALU.

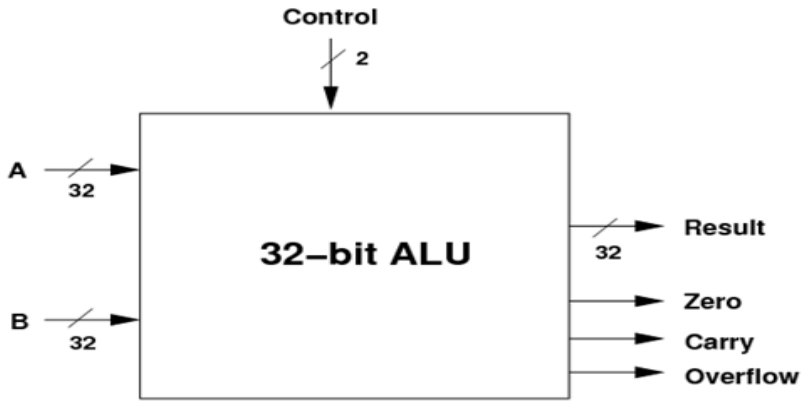


Fig-1: 32-bit ALU

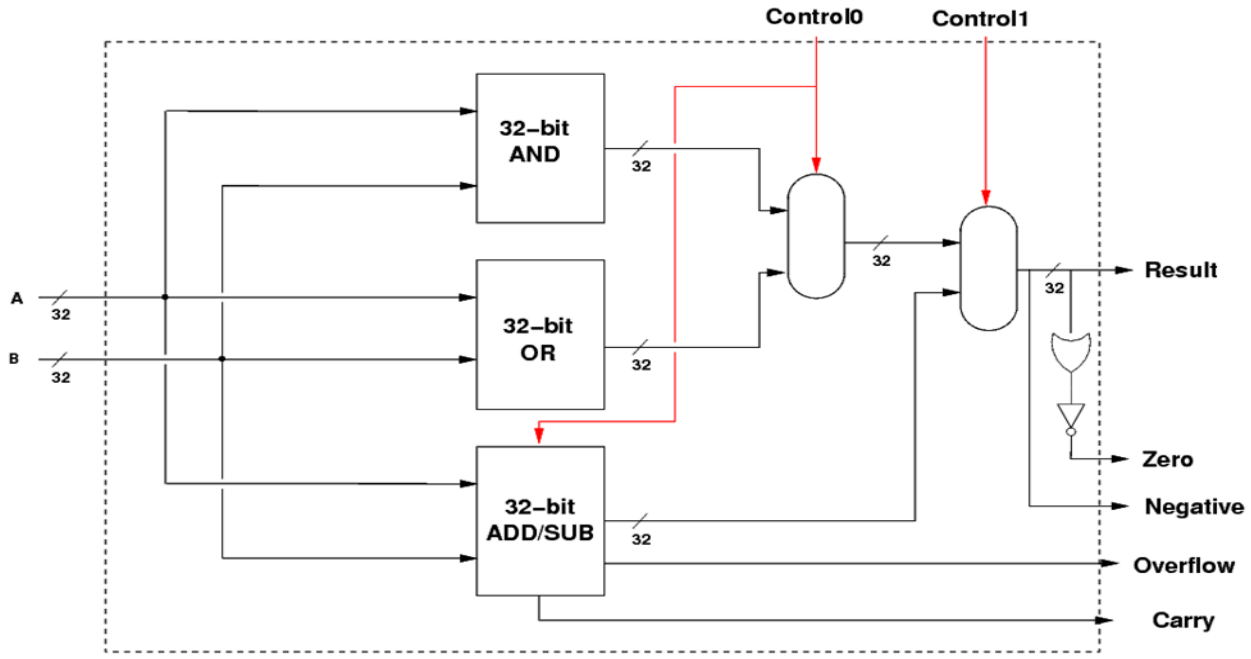


Fig-2: Logic Diagram of 32-bit ALU

Basic components:

a	b	out
0	0	0
0	1	0
1	0	0
1	1	1

a	b	out
0	0	0
0	1	1
1	0	1
1	1	1

a	b	out
0	0	0
0	1	1
1	0	1
1	1	0

in	out
0	1
1	0

a)To Verify the Functionality using Test Bench

Source Code – Using Case Statement:

```
module alu_32bit_case(y,a,b,f);
input [31:0]a;
input [31:0]b;
input [2:0]f;
output reg [31:0]y;
always@(*)
```

```
begin
case(f)
3'b000:y=a&b;        //AND Operation
3'b001:y=a|b;        //OR Operation
3'b010:y=~(a&b);     //NAND Operation
3'b011:y=~(a|b);     //NOR Operation
3'b010:y=a+b;        //Addition
3'b011:y=a-b;        //Subtraction
3'b100:y=a*b;        //Multiply
default:y=32'bx;

endcase
end
endmodule
```

Test Bench:

```
module alu_32bit_tb_case;

reg [31:0]a;
reg [31:0]b;
reg [2:0]f;
wire [31:0]y;
alu_32bit_case test2(.y(y),.a(a),.b(b),.f(f));
initial begin
a=32'h00000000; b=32'hFFFFFFFF;
 #10 f=3'b000;
#10 f=3'b001;
#10 f=3'b010;
#10 f=3'b100;
end

initial
#50 $finish;
endmodule
```

B:Source Code - Using If Statement:

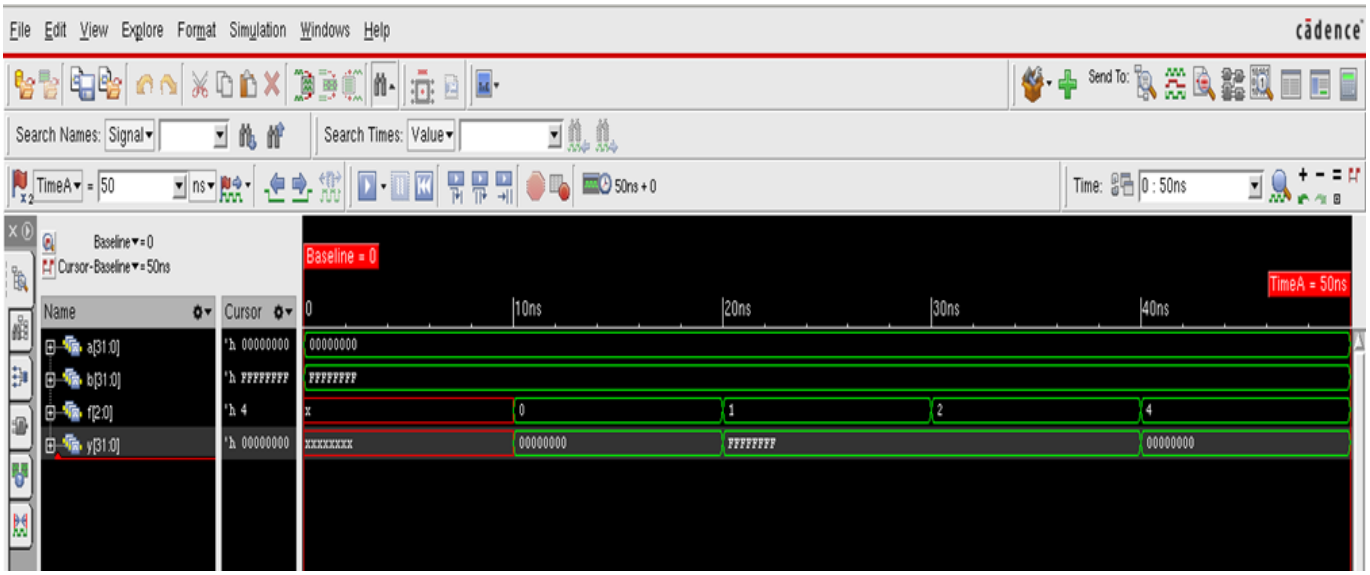
```
module alu_32bit_if(y,a,b,f);
input [31:0]a;
input [31:0]b;
input [2:0]f; output reg [31:0]y;
always@(*)
begin
if(f==3'b000)
y=a&b;               //AND Operation else if (f==3'b001)
y=a|b;               //OR Operation else if (f==3'b010)
y=a+b;               /Addition else if (f==3'b011)
```

```
y=a-b;                  /Subtraction else if (f==3'b100)
y=a*b;                  //Multiply else
y=32'bx;
end
endmodule
```

Test bench:

```
module alu_32bit_tb_if;
reg [31:0]a;
reg [31:0]b;
reg [2:0]f;
wire [31:0]y;

alu_32bit_if test(.y(y),.a(a),.b(b),.f(f));
initial begin
a=32'h00000000;

b=32'hFFFFFFFF; #10 f=3'b000;
#10 f=3'b001;
#10 f=3'b010;
#10 f=3'b100;
end initial
#50 $finish;
endmodule
```

Waveforms:



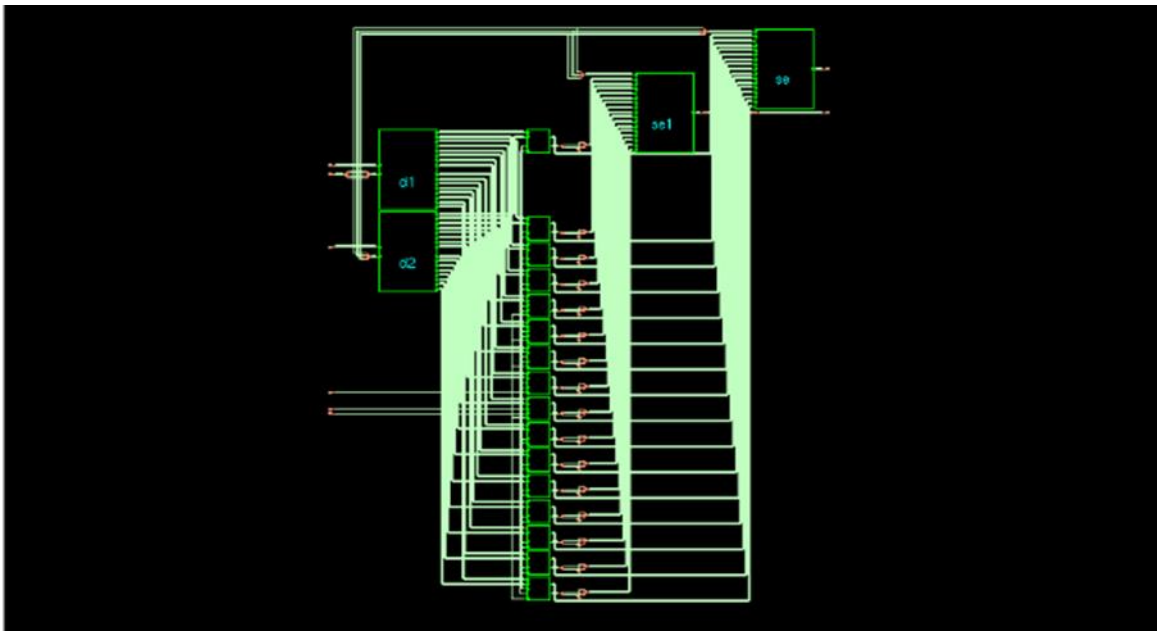
b) Synthesize Design

•Run the synthesis Process one time for each code and make sure the output File names are changed accordingly.

Synthesis Process:

1. read_libs /home/install/FOUNDRY/digital/90nm/dig/lib/slow.lib
2. read_hdl {alu_32bit_if.v (OR) alu_32bit_case.v} //Choose any one
3. elaborate
4. read_sdc constraints_top.sdc //Optional-Reading Top Level SDC
5. synthesize -to_mapped -effort medium //Performing Synthesis Mapping and Optimization
6. report_timing > alu_timing.rep //Generates Timing report for worst datapath and dumps into file
7. report_area > alu_area.rep //Generates Synthesis Area report and dumps into a file
8. report_power > uart_power.rep //Generates Power Report [Pre-Layout]
9. write_hdl > uart_netlist.v //Creates readable Netlist File
10. write_sdc > uart_sdc.sdc //Creates Block Level SDC

RTL Schematic of 32-bit Floating Point ALU with Pipelining



Note:-

1. Tabulate Area, Power and Timing Constraints using any of the SDC Constraints as instructed.
2. Make sure, during synthesis the Report File Names are changed so that the latest reports do not overwrite the earlier ones.

RESULT: Verilog code for the 32-bit ALU circuit and its test bench for verification is written, the waveform is observed and the code is synthesized with the technological library and given constraints and is verified.

Experiment No: 5
Title: Latches and Flip Flops

Problem Statement: Write verilog code for D, SR, JK Latch and Flip-flop, Synthesize the design and compare the synthesis report.

Objectives:

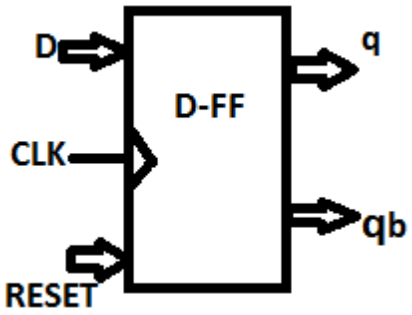
1. Write Verilog code for D, SR, JK Latch
2. Write Verilog code for D, SR, JK Flip Flop
3. Synthesize the designs

5a. i) D flipflop

Theory:

D flip-flop The D flip-flop is widely used. It is also known as a "data" flip-flop. The D flip-flop captures the value of the D-input at a definite portion of the clock cycle (such as the rising edge of the clock). That captured value becomes the Q output. At other times, the output Q does not change. The D flip-flop can be viewed as a memory cell, a zero-order hold, or a delay line.

Block diagram:



Truth Table:

Reset	Clock	D	Q	Qb
0	↑	1	1	0
0	↑	0	0	1
0	0	X	X	Hold
1	↑	X	0	1

Design Equation/ calculations (if any):

Q(t+1) = D(t)

Verilog Code with Comments:

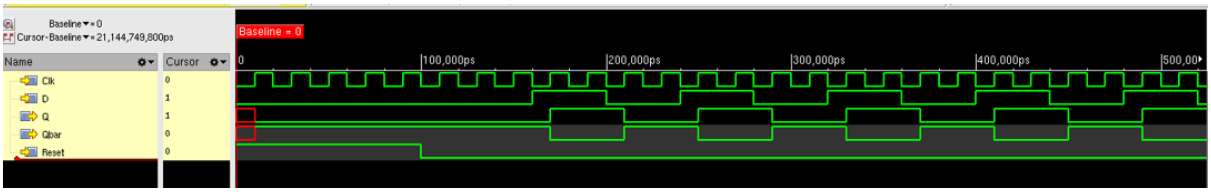
```
`timescale 1ns / 10ps
module DFF( Q,Qbar,D,Clk,Reset);
output reg Q;
output Qbar;
input D,Clk,Reset;
always @(posedge Clk)
begin
if (Reset == 1'b1)        //If at reset
Q <= 1'b0;
else
```

```
Q <= D;
end
assign Qbar = ~Q;
endmodule
```

Test bench code:

```
`timescale 1ns / 10ps
module DFF_tb;
reg D, Clk, Reset;                  // Inputs
wire Q, Qbar;              // Outputs
// Instantiate the Unit Under Test (UUT)
DFF uut ( .Q(Q),  .Qbar(Qbar),  .D(D),  .Clk(Clk), .Reset(Reset));
initial begin
D  = 1'b0; // Initialize Inputs
Clk  = 1'b0;
Reset   = 1'b1;
// Wait 100 ns for global reset to finish
#100;
// Add stimulus here
Reset = 1'b0;
#20;
forever #40 D = ~ D;
end
always #10 Clk = ~Clk;
endmodule
```

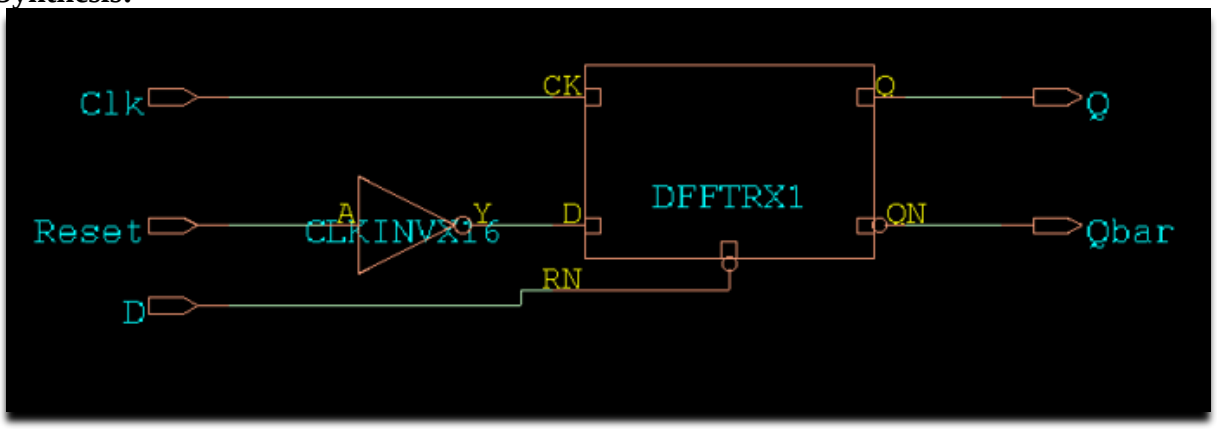
Simulation output:



Constraint File:

```
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_transition -
set_clock_uncertainty 0.01 [get_clocks "clk"]
set_input_delay -max 1 [get_ports "Reset"] -clock [get_clocks "clk"]
set_input_delay -max 1 [get_ports "D"] -clock [get_clocks "clk"]
set_output_delay -max 1 [get_ports "Q"] -clock [get_clocks "clk"]
set_output_delay -max 1 [get_ports "Qbar"] -clock [get_clocks "clk"]
```

Synthesis:



Gatelevel netlist, gate, area, timing, power reports are generated and analysed by the students

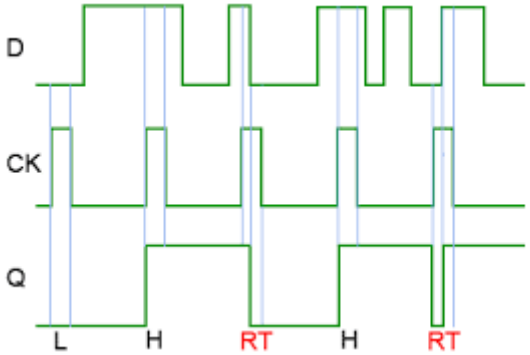
Expected Outcomes:

The students will be able to

- Write the Verilog code and analyze it with various testcases for the given input and output
- Analyse the reports generated with different input clock frequency

Additional Experimental Questions:

Given the test inputs as given in the figure below and verify the outputs



Change the following constraints and analysis the timing, power, area and gate reports

- **Clock inputs as 2ns, 4ns, 0.5ns in the constraint file and analyze the result and reports**

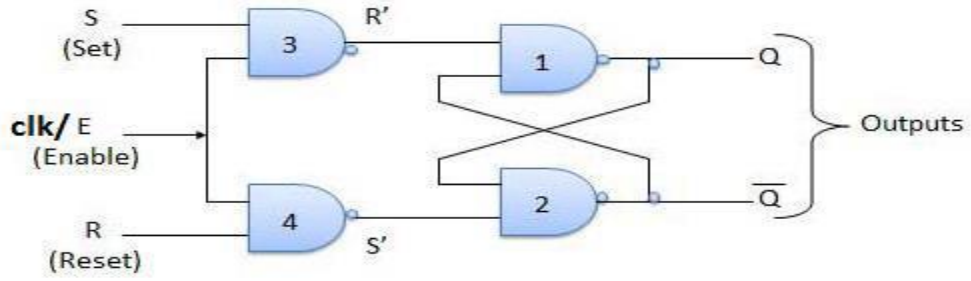
Change the rising time and falling time to max of 0.5 and 1.5ns both at the input and /or output and analyze the result and reports

5a ii) S R Flip Flop

Theory:

It can be seen that when both inputs S = "1" and R = "1" the outputs Q and Q can be at either logic level "1" or "0", depending upon the state of the inputs S or R BEFORE this input condition existed. Therefore the condition of S = R = "1" does not change the state of the outputs Q and Q. However, the input state of S = "0" and R = "0" is an undesirable or invalid condition and must be avoided. The condition of S = R = "0" causes both outputs Q and Q to be HIGH together at logic level "1" when we would normally want Q to be the inverse of Q. The result is that the flip-flop loses control of Q and Q, and if the two inputs are now switched "HIGH" again after this condition to logic "1", the flip-flop becomes unstable and switches to an unknown data state based upon the unbalance.

Logic diagram:



Truth Table:

Clock	R	S	Q	Qb	Comments
↑	0	0	Hold		No change
↑	0	1	1	0	Set
0	1	0	0	1	Reset
↑	1	1	z	z	Indeterminate
0	X	X	Hold		No change

Design Equation/ calculations (if any):

Q(t+1) = R'(t)Q(t) + S(t) ; S(t)R(t) = 0

Verilog Code with Comments:

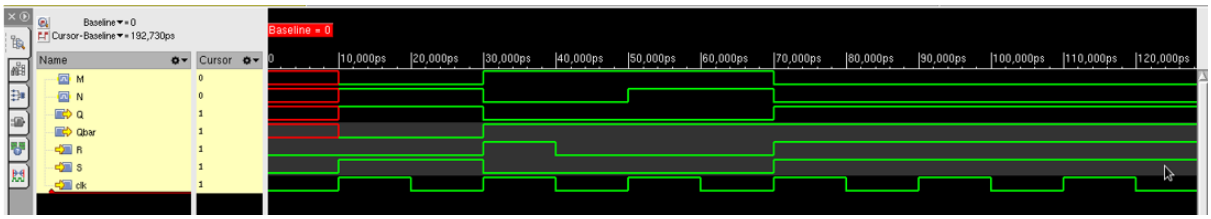
```
`timescale 1ns / 10ps
module SR_FF(S,R,clk,Q,Qbar);
input S,R,clk;
output Q,Qbar;
reg M,N;
always @(posedge clk)
begin
M = !(S & clk);
N = !(R & clk);
end
assign Q = !(M & Qbar);
assign Qbar = !(N & Q);
endmodule
```

Test bench code:

```
`timescale 1ns / 10ps
 module SRFF_tb;
 reg S, R; reg clk;
// Outputs
 wire Q, Qbar;
 // Instantiate the Unit Under Test (UUT)
 SR_FF uut (  .S(S), .R(R),  .clk(clk), .Q(Q), .Qbar(Qbar) );
initial begin
// Initialize Inputs
 clk  = 1'b0;
 S =1'b0;
R=1'b0;
```

```
// Wait 100 ns for global reset to finish
#10;
// Add stimulus here
S =1'b1;R=1'b0;
#20   S =1'b0; R=1'b1;
#10   S =1'b0;   R=1'b0;
#30   S =1'b1; R=1'b1;
#200 $finish;
 end
always #10 clk = ~clk;
endmodule
```

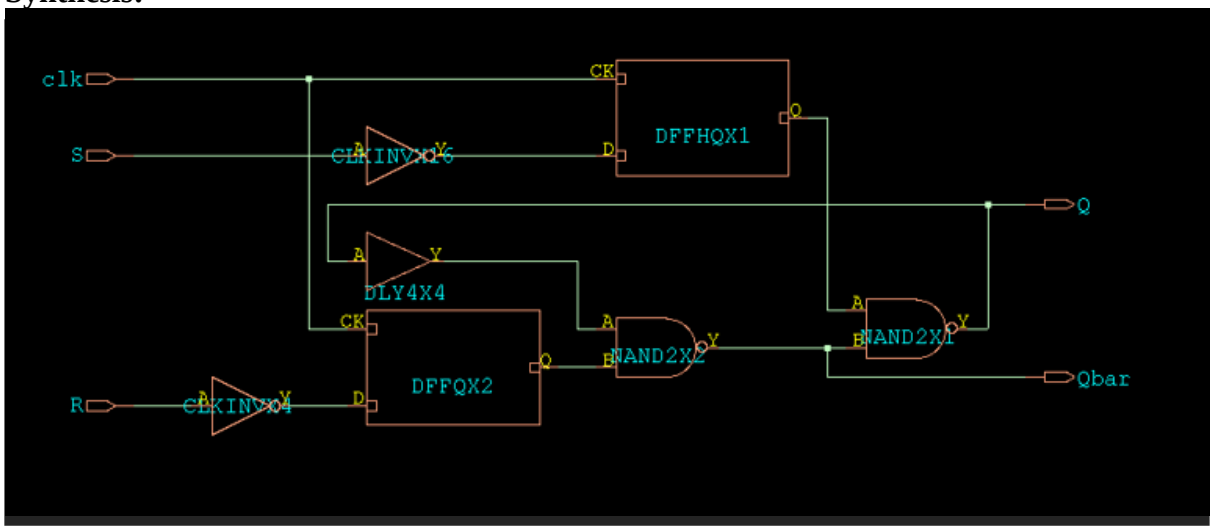
Simulation output:



Constraint File:

```
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_transition -
set_clock_uncertainty 0.01 [get_clocks "clk"]
set_input_delay -max 1 [get_ports "S"] -clock [get_clocks "clk"]
set_input_delay -max 1 [get_ports "R"] -clock [get_clocks "clk"]
set_output_delay -max 1 [get_ports "Q"] -clock [get_clocks "clk"]
set_output_delay -max 1 [get_ports "Qbar"] -clock [get_clocks "clk"]
```

Synthesis:

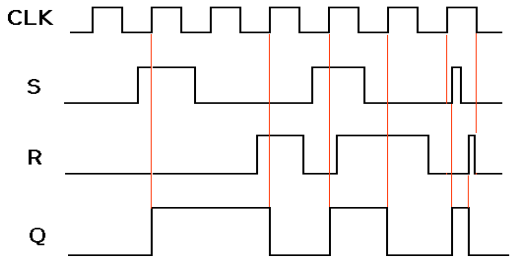


Gatelevel netlist, gate, area, timing, power reports are generated and analysed by the students

Expected Outcomes:
The students will be able to

- Write the Verilog code and analyze it with various testcases for the given input and output
- Analyse the reports generated with different input clock frequency

Additional Experimental Questions:
Given the test inputs as given in the figure below and verify the outputs

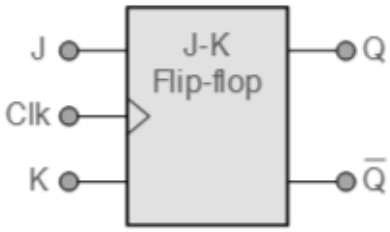


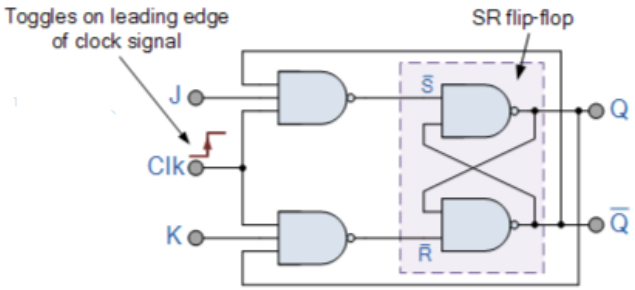
Change the following constraints and analysis the timing, power, area and gate reports

- **Clock inputs as 2ns, 4ns, 0.5ns in the constraint file and analyze the result and reports**
- **Change the rising time and falling time to max of 0.5 and 1.5ns both at the input and /or output and analyze the result and reports**

5a iii) JK Flip Flop
Theory:
Then the JK flip-flop is basically an SR flip-flop with feedback which enables only one of its two input terminals, either SET or RESET to be active at any one time thereby eliminating the invalid condition seen previously in the SR flip-flop circuit. Also when both the J and the K inputs are at logic level "1" at the same time, and the clock input is pulsed either "HIGH", the circuit will "toggle" from its SET state to a RESET state, or visa-versa. This result in the JK flip-flop acting more like a T-type toggle flip-flop when both terminals are "HIGH"
Although this circuit is an improvement on the clocked SR flip-flop it still suffers from timing problems called "race" if the output Q changes state before the timing pulse of the clock input has time to go "OFF". To avoid this the timing pulse period (T) must be kept as short as possible (high frequency). As this is sometimes not possible with modern TTL IC's the much improved MasterSlave JK Flip-flop was developed

Block and Logic diagram:





Truth Table:

Clock	J	K	Q	Qm	Comments
↑	0	0	Hold		No change
↑	0	1	1	0	Set

0	1	0	0	1	Reset
↑	1	1	0	1	Toggle
			1	0	
0	X	X	Hold		No change

Design Equation/ calculations (if any):

Q(t+1) = K'(t)Q(t) + J(t)Q'(t)

Verilog Code with Comments:

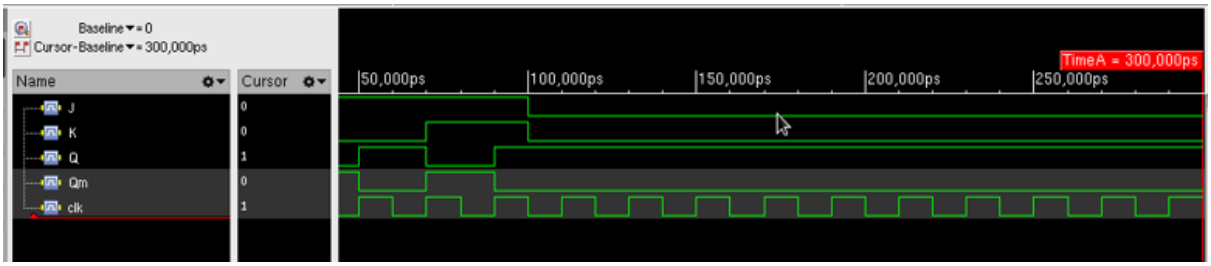
```
`timescale 1ns / 10ps
module jkff(J, K, clk, Q, Qm);
input J, K, clk;
output reg Q;
output Qm;
always @(posedge clk)
begin
if(J == 0 && K == 0)
Q <= 0;
else if(J == 1 && K == 0)
Q <= 1;
else if(J == 0 && K == 1)
Q <= 0;
else if(J == 1 && K == 1)
Q <= ~Q;
end
assign Qm = ~Q;
endmodule
```

Test bench code:

```
`timescale 1ns / 10ps
 module JKFF_tb;
// Inputs
 reg J, K,,clk;
// Outputs
wire Q, Qm;
// Instantiate the Unit Under Test (UUT)
jkff uut ( .J(J), .K(K),  .clk(clk),  .Q(Q), .Qm(Qm)      );
initial begin
 // Initialize Inputs
 clk  = 1'b0;
 J =1'b0;
K=1'b0;
// Wait 100 ns for global reset to finish
#10;
// Add stimulus here
 J =1'b1; K=1'b0;
#20 J =1'b1;    K=1'b1;
#10 J =1'b1;    K=1'b0;
#30 J =1'b1;    K=1'b1;
#30 J =1'b0;    K=1'b0;
```

```
#200 $finish;
 end
 always #10 clk = ~clk;
 endmodule
```

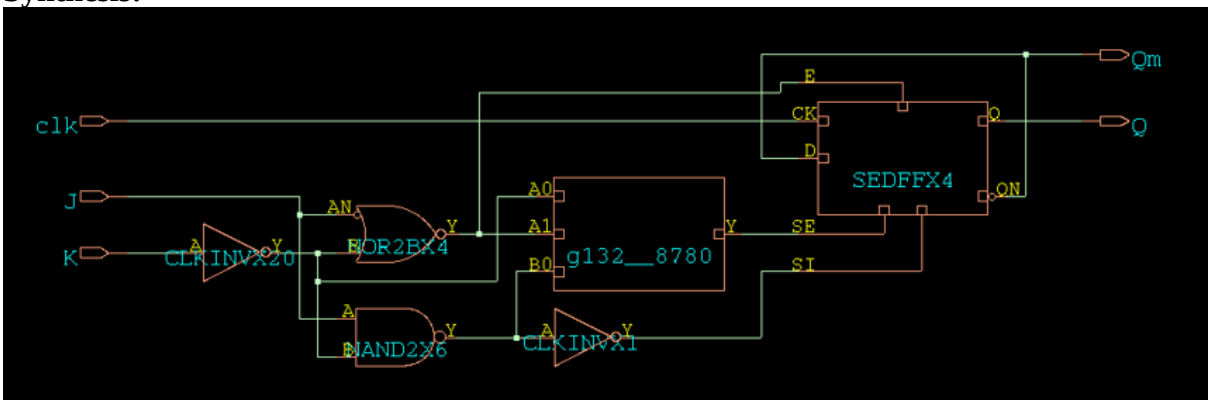
Simulation output:



Constraint File:

```
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_transition -
set_clock_uncertainty 0.01 [get_clocks "clk"]
set_input_delay -max 1 [get_ports "J"] -clock [get_clocks "clk"]
set_input_delay -max 1 [get_ports "K"] -clock [get_clocks "clk"]
set_output_delay -max 1 [get_ports "Q"] -clock [get_clocks "clk"]
set_output_delay -max 1 [get_ports "Qm"] -clock [get_clocks "clk"]
```

Synthesis:



Gatelevel netlist, gate, area, timing, power reports are generated and analysed by the students

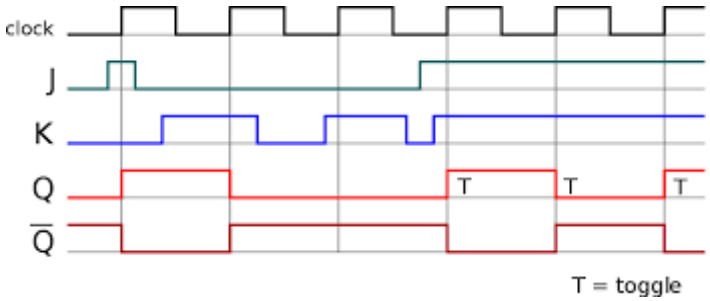
Expected Outcomes:

The students will be able to

- Write the Verilog code and analyze it with various testcases for the given input and output
- Analyse the reports generated with different input clock frequency

Additional Experimental Questions:

Given the test inputs as given in the figure below and verify the outputs



Change the following constraints and analysis the timing, power, area and gate reports

- **Clock inputs as 2ns, 4ns, 0.5ns in the constraint file and analyze the result and reports**
- **Change the rising time and falling time to max of 0.5 and 1.5ns both at the input and /or output and analyze the result and reports**

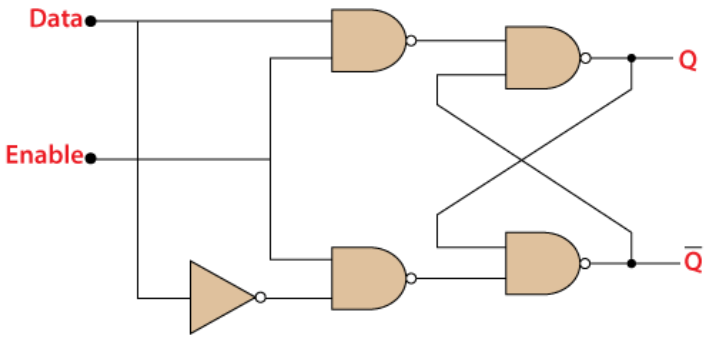
5b i) D Latch

Theory:

The **Gated D Latch** is another special type of gated latch having two inputs, i.e., DATA and ENABLE. When the enable input set to 1, the input is the same as the Data input. Otherwise, there is no change in output.

We can design the gated D latch by using gated SR latch. The set and reset inputs are connected together using an inverter. By doing this, the outputs will be opposite to each other. Below is the circuit diagram of the Gated D latch**.**

Logic diagram:



Truth Table:

Enable(en)	D	Q	Qbar
0	0	Latch	latch
0	1	Latch	latch
1	0	0	1
1	1	1	0

Verilog Code with Comments:

```
`timescale 1ns/10ps
module DFF( Q,Qbar,D,en,Reset);
output reg Q;
output Qbar;
```

```
input D,en,Reset;
always @ (en or Reset or D)
 if (!Reset)
 Q <= 0;
 else
 if (en)
 Q <= D;
assign Qbar = ~Q;
endmodule
```

Test bench code:

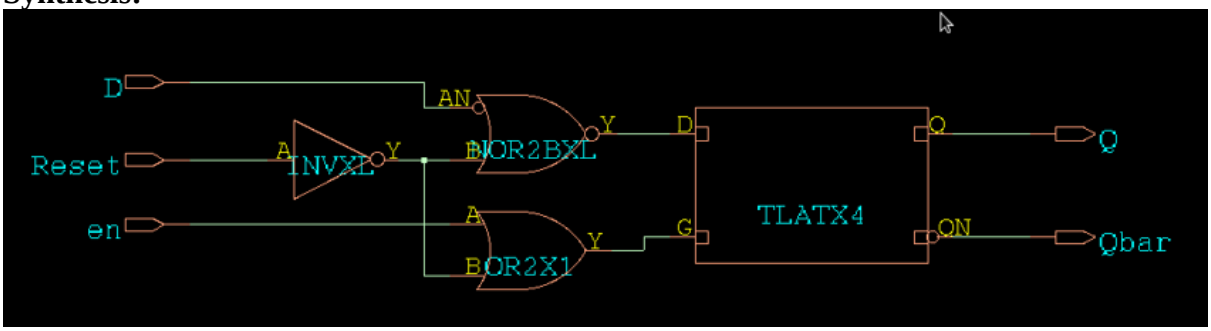
```
`timescale 1ns / 10ps
module DFF_tb;
reg D, en, Reset;     // Inputs
wire Q, Qbar; // Outputs
// Instantiate the Unit Under Test (UUT)
DFF uut ( .Q(Q),  .Qbar(Qbar),  .D(D), .en(en), .Reset(Reset) );
initial begin
D  = 1'b0;        // Initialize Inputs
en =1'b1;
Reset   = 1'b1;
#50;              // Wait 100 ns for global reset to finish
// Add stimulus here
Reset =1'b0;
#10 D=1'b1;
#20 Reset =1'b1;
#50 en =1'b0;
#10 D=1'b0;
#10 D=1'b1;
#10 D=1'b1;
#50 en =1'b1;
#20 D=1'b1;
#10 D=1'b0;
#20 D=1'b1;
#10 D=1'b0;
#50 D=1'b1;
#400 $finish;
end
endmodule
```

Simulation output:

Constraint File:

```
set_input_delay -max 1 [get_ports "D"]
set_input_delay -max 1 [get_ports "en"]
set_output_delay -max 1 [get_ports "Q"]
set_output_delay -max 1 [get_ports "Qbar"]
```

Synthesis:



Gatelevel netlist, gate, area, timing, power reports are generated and analysed by the students

Expected Outcomes:

The students will be able to

- Write the Verilog code and analyze it with various testcases for the given input and output
- Analyse the reports generated with different input clock frequency

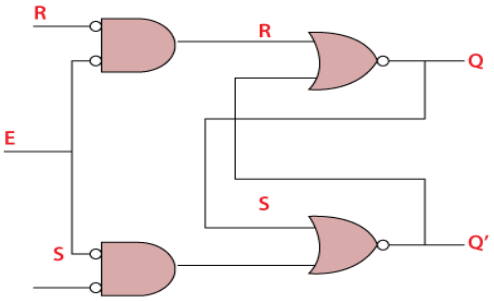
Note: Additional variations that can be done from this experiment
(Eg: for flip flops given the waveform, write the test bench and obtained the given output Possible change of values in the constraint file and analyze the result and reports)

5b ii) SR Latch

Theory:

A **Gated SR Latch** is a special type of **SR Latch** having three inputs, i.e., Set, Reset, and Enable. The enable input must be active for the SET and RESET inputs to be effective. The **ENABLE** input of gated SR Latch enables the operation of the SET and RESET inputs. This **ENABLE** input connects with a switch. The Set-Reset inputs are enabled when this switch is on. Otherwise, all the changes are ignored in the set and reset inputs. Below are the circuit diagram and the truth table of the Gated SR latch.

Logic diagram: //using NOR gate



Truth Table:

Enable(en)	S	R	Q	Qm
1	0	0	Latch	Latch
1	0	1	0	1
1	1	0	1	0
1	1	1	0	0
0	X	X	Latch	

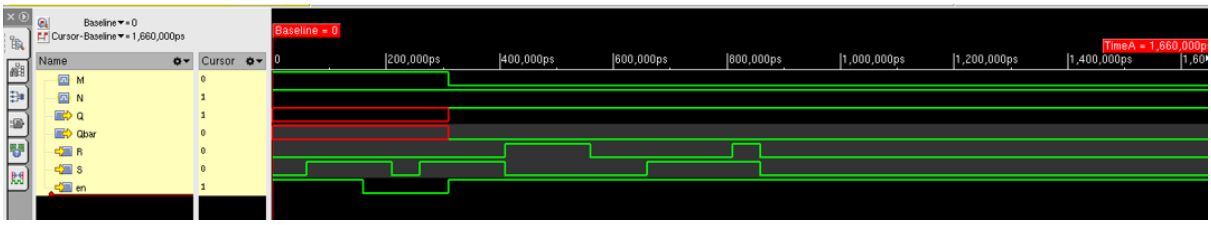
Verilog Code with Comments:

```
`timescale 1ns / 10ps
module SRLatch (S,R,en,Q,Qm);
input S,R,en;
output Q,Qm;
reg M,N;
always @(en)
begin
M <= !(S & en);
N <= !(R & en);
end
assign Q = !(M & Qm);
assign Qm = !(N & Q);
endmodule
```

Test bench code:

```
`timescale 1ns / 10ps
module SRlatch_tb;
reg S, R, en;              // Inputs
wire Q, Qm;    // Outputs
// Instantiate the Unit Under Test (UUT)
SRLatch uut (.S(S), .R(R), .en(en), .Q(Q), .Qm(Qm));
initial begin
S  = 1'b0;        // Initialize Inputs
R  = 1'b0;
en =1'b1;
#50;              // Wait 100 ns for global reset to finish
// Add stimulus here
#10 S=1'b1; R=1'b0;
#100 en =1'b0;
#50 S=1'b0; R=1'b0;
#50 S=1'b1; R=1'b0;
#50 en =1'b1;
#100 S=1'b0; R=1'b1;
#150 S=1'b0; R=1'b0;
#100 S=1'b1; R=1'b0;
#150 S=1'b1; R=1'b1;
#50 S=1'b0; R=1'b0;
#200 S=1'b0;R=1'b0;
#600 $finish;
end
endmodule
```

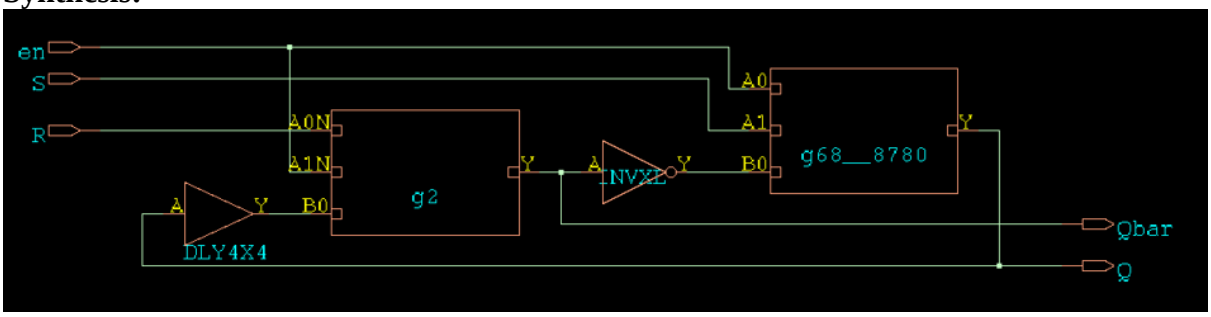
Simulation output:



Constraint File:

```
set_input_delay -max 1 [get_ports "R"]
set_input_delay -max 1 [get_ports "en"]
set_input_delay -max 1 [get_ports "S"]
set_output_delay -max 1 [get_ports "Q"]
set_output_delay -max 1 [get_ports "Qbar"]
```

Synthesis:



Gatelevel netlist, gate, area, timing, power reports are generated and analysed by the students

Expected Outcomes:

The students will be able to

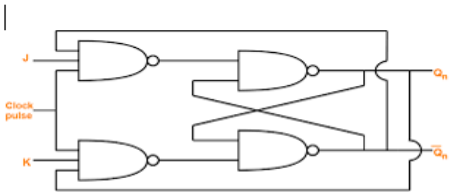
- Write the Verilog code and analyze it with various testcases for the given input and output
- Analyse the reports generated with different input clock frequency

Note: Additional variations that can be done from this experiment
(Eg: for flip flops given the waveform, write the test bench and obtained the given output Possible change of values in the constraint file and analyze the result and reports)

5b iii) JK Latch
Theory:

The **JK Latch** is the same as the **SR Latch**. In JK latch, the unclear states are removed, and the output is toggled when the JK inputs are high. The only difference between SR latch JK latches is that there is no output feedback towards the inputs in the SR latch, but it is present in the JK latch. The circuit diagram and truth table of the JK latch are as follows:

Logic diagram:



Truth Table:

Enable	J	K	Q	Qm	Comments
1	0	0	Q	Q	No change
1	0	1	1	0	Set
1	1	0	0	1	Reset
1	1	1	Qb	Q	Toggle
0	X	X	Latch		No change

Verilog Code with Comments:

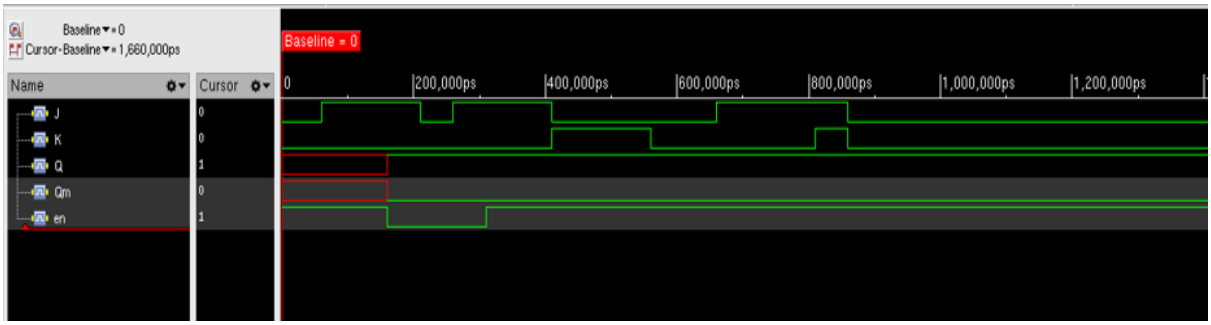
```
`timescale 1ns / 10ps
module JKLatch(J, K, en, Q,Qm);
input J, K, en;
output reg Q;
output Qm;
always @(en)
begin
if(J == 0 && K == 0)
Q <= Q;
else if(J == 1 && K == 0)
Q <= 1;
else if(J == 0 && K == 1)
Q <= 0;
else if(J == 1 && K == 1)
Q <= ~Q;
end
assign Qm=~Q;
endmodule
```

Test bench code:

```
`timescale 1ns / 10ps
module JKlatch_tb;
reg J, K, en;              // Inputs
wire Q, Qm;    // Outputs
// Instantiate the Unit Under Test (UUT)
JKLatch uut (.J(J), .K(K), .en(en), .Q(Q), .Qm(Qm));
initial begin
J  = 1'b0;        // Initialize Inputs
K  = 1'b0;
en =1'b1;
#50;              // Wait 100 ns for global reset to finish
// Add stimulus here
#10 J=1'b1; K=1'b0;
#100 en =1'b0;
```

```
#50 J=1'b0; K=1'b0;
#50 J=1'b1; K=1'b0;
#50 en =1'b1;
#100 J=1'b0; K=1'b1;
#150 J=1'b0; K=1'b0;
#100 J=1'b1; K=1'b0;
#150 J=1'b1; K=1'b1;
#50 J=1'b0; K=1'b0;
#200 J=1'b0; K=1'b0;
#600 $finish;
end
endmodule
```

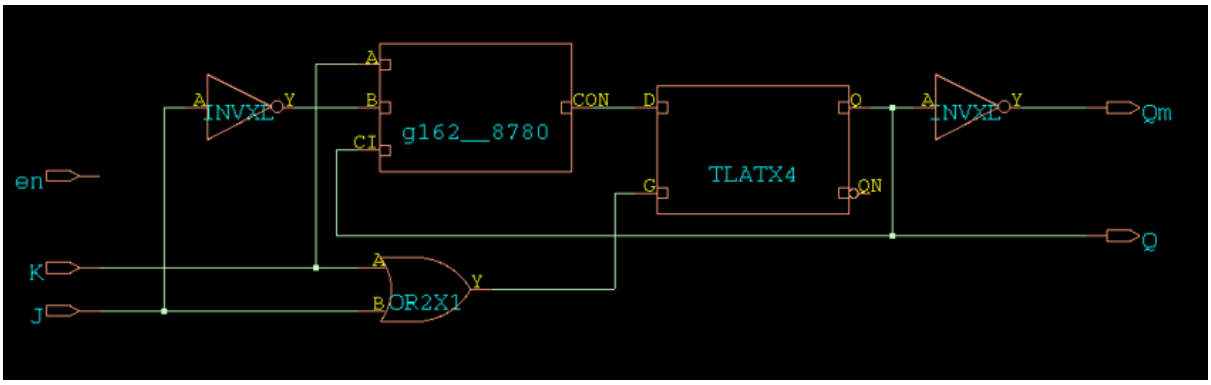
Simulation output:



Constraint File:

```
set_input_delay -max 1 [get_ports "J"]
set_input_delay -max 1 [get_ports "K"]
set_input_delay -max 1 [get_ports "en"]
set_output_delay -max 1 [get_ports "Q"]
set_output_delay -max 1 [get_ports "Qm"]
```

Synthesis:



Gatelevel netlist, gate, area, timing, power reports are generated and analysed by the students

Expected Outcomes:

The students will be able to

- Write the Verilog code and analyze it with various testcases for the given input and output
- Analyse the reports generated with different input clock frequency

Experiment No: 6
Title: Physical Design

Problem statement: To develop the Physical Design (Floor planning, Placement, Routing in Layout of any two experiments of 1 to 5

Objective:

For the synthesized netlist carry out the following for any two above experiments:

a. Floor planning (automatic), identify the placement of pads
b. Placement and Routing, record the parameters
c. Physical verification and record the LVS and DRC reports
d. Perform Back annotation and verify the functionality of the design
e. Generate GDSII and record the number of masks and its color composition

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).

- Make sure the Synthesis for the target design is done and open aterminal from the corresponding workspace.
- Initiate the Cadence tools 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

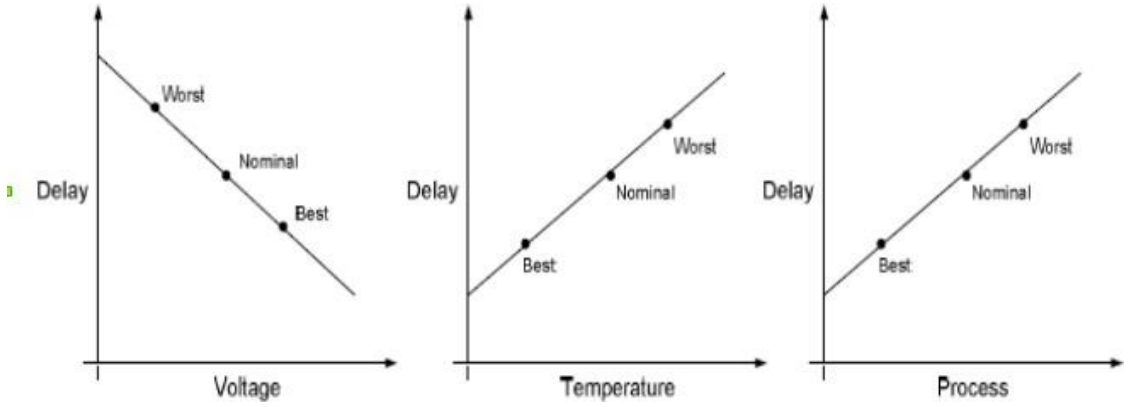
Importing Design

To Import Design, all the Mandatory Inputs are to be loaded and this canbe done either using script files named with .globals and .view/.tcl or through GUI as shown below.

The target design considered here is 4-bit counter.
The procedure shall remain the same for any other design from theabove discussed experiments.
Note:

1. For Synthesis, slow.lib was read as input. Each liberty file contains apre-defined Process, Voltage and Temperature (PVT) values which impact the ease of charge movement.

2.Process, Voltage and Temperature individually affect the ease ofcurrents as depicted below.



3. Hence, slow.lib contains PVT combination (corner) with **slow** charge movement => **Maximum** Delay => **Worst** Performance

4. Similarly, fast.lib contains PVT Combinationapplicable across its designs to give **Fast** charge movement => **Minimum** Delay => **Best** Performance.

5. When these corners are collaborated with the **sdc**, they can be used to analyze timing for setup in the worst case and hold in the best case.

6. All these analysis views are to be manually created either in the formof script or using the GUI.

```
#################################################################
#  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
```

Script under Default. Globals file

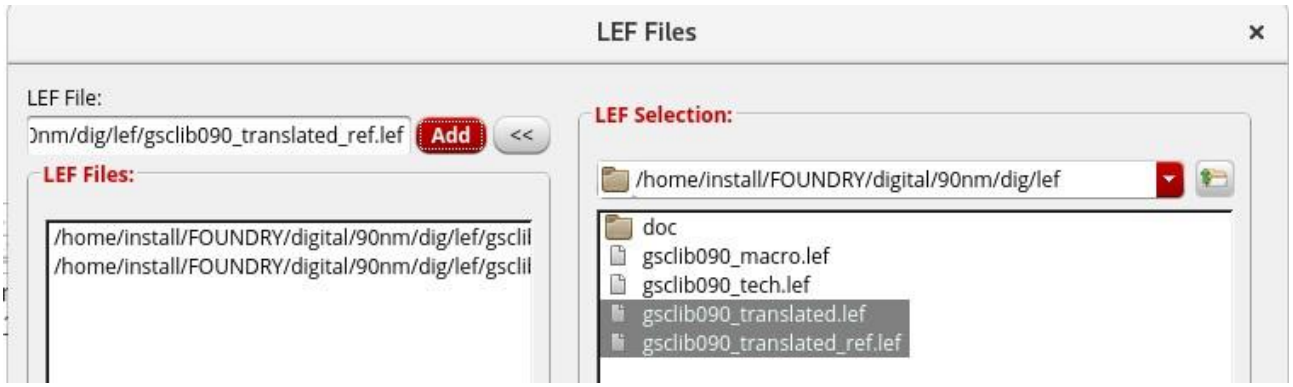
```
# Version:1.0 MMMC View Definition File
# Do Not Remove Above Line
create_library_set -name MAX_timing -timing {/root/Desktop/counter/lib/90/slow.lib}
create_library_set -name MIn_timing -timing {/root/Desktop/counter/lib/90/fast.lib}
create_constraint_mode -name Constraints -sdc_files {counter_sdc.sdc}
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}
```

Script under Default.view (or) Default.tcl file
Note: Check the paths to properly read in the input files.

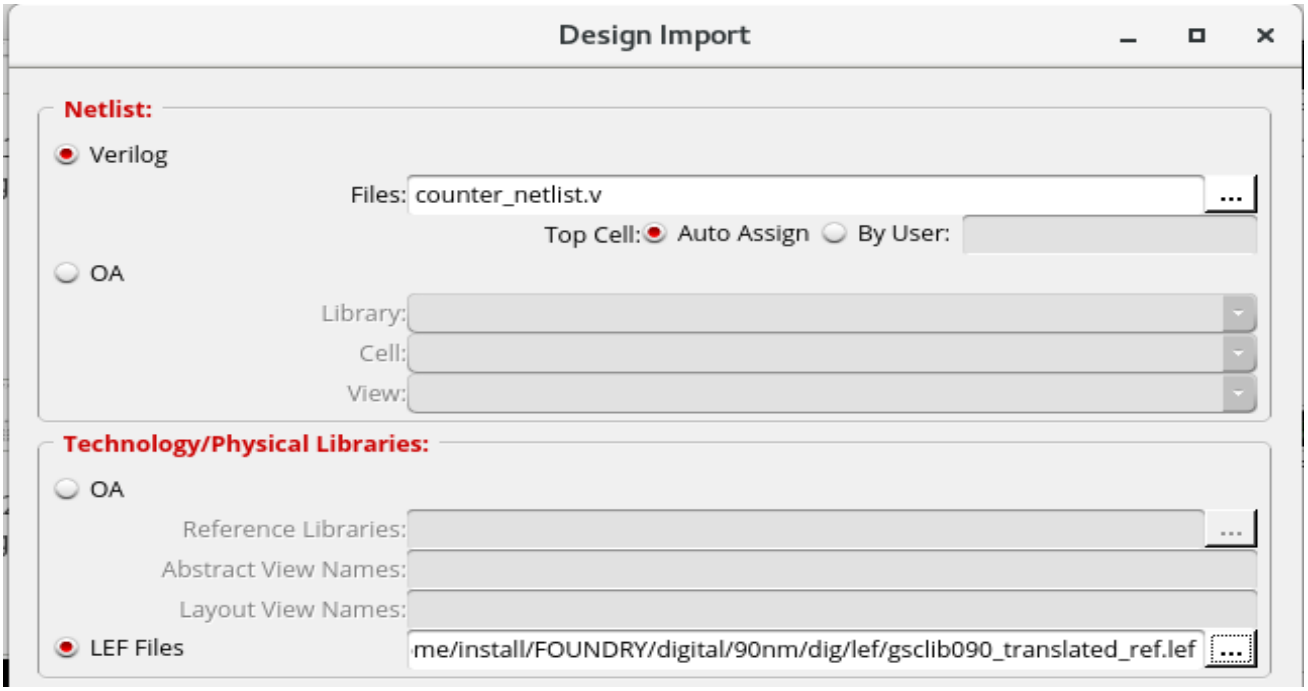
- Else, if you would like to import your design using GUI, open theInnovus tool and from the GUI, go to File → Import Design.
- A new pop-up window appears.

- First load the netlist. You can browse for the file and select “Topcell : Auto Assign”.

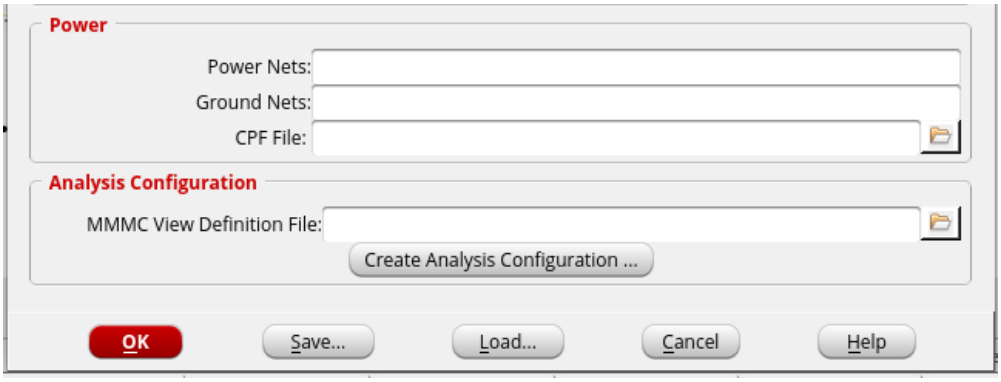
Similarly select your lef files from /home/install/FOUNDRY/digital/90nm/dig/lef/ as shown below.



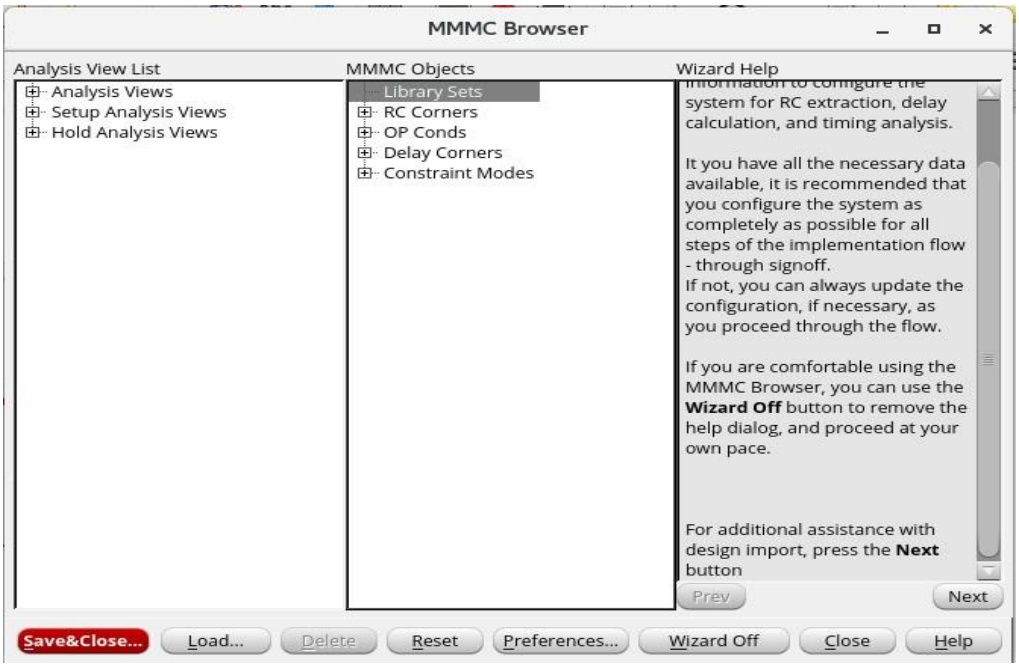
Once both the Netlist and LEF Files are load ed, your import design window is as follows.



- In order to load the Liberty File and SDC, create delay corners andanalysis view, select the “Create Analysis Configuration” option atthe bottom.



- An MMMC browser Pops Up.

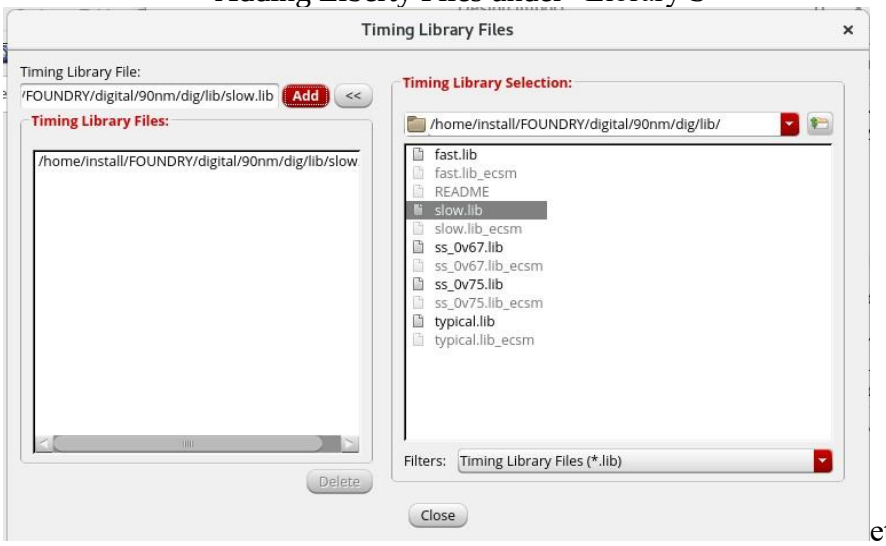


The order of adding the MMMC Objects is as follows.

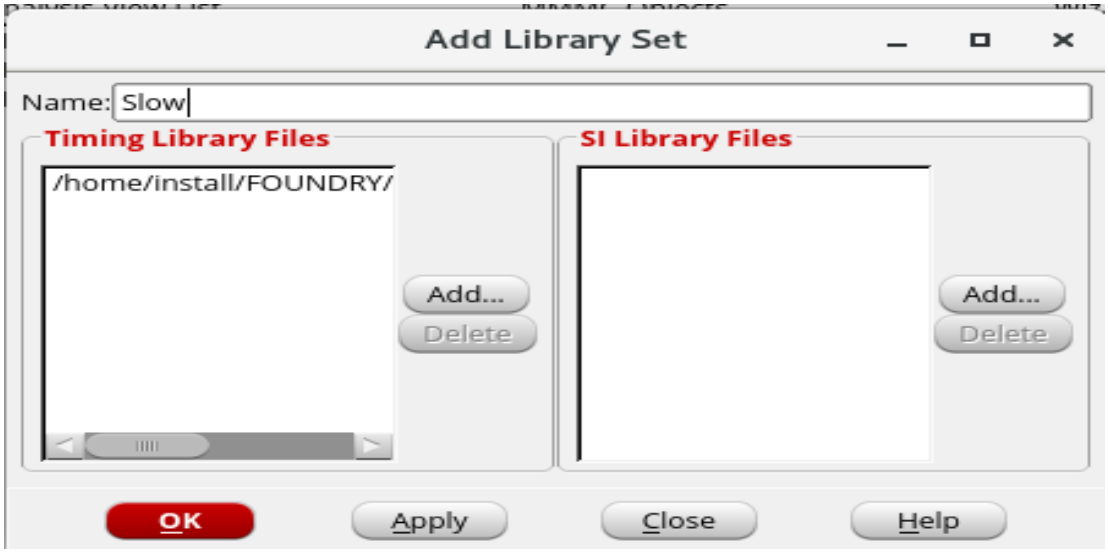
1. Library Sets
2. RC Corners
3. Delay Corners
4. Constraints (SDC)

Once all of them are added, Analysis Views are created and assigned toSetup and Hold.
In order to add any of the objects, make a right click on the correspondinglabel → Select New.

Adding Liberty Files under "Library S

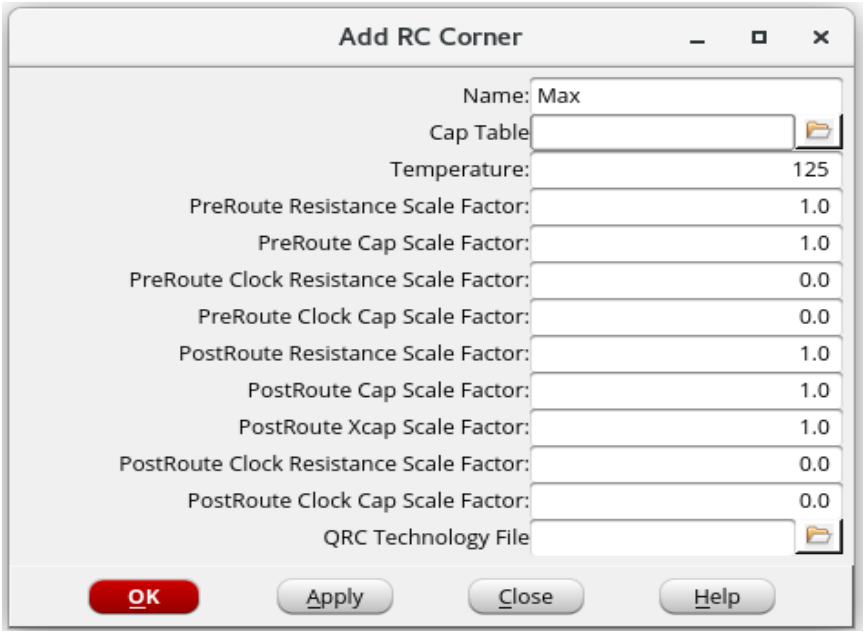


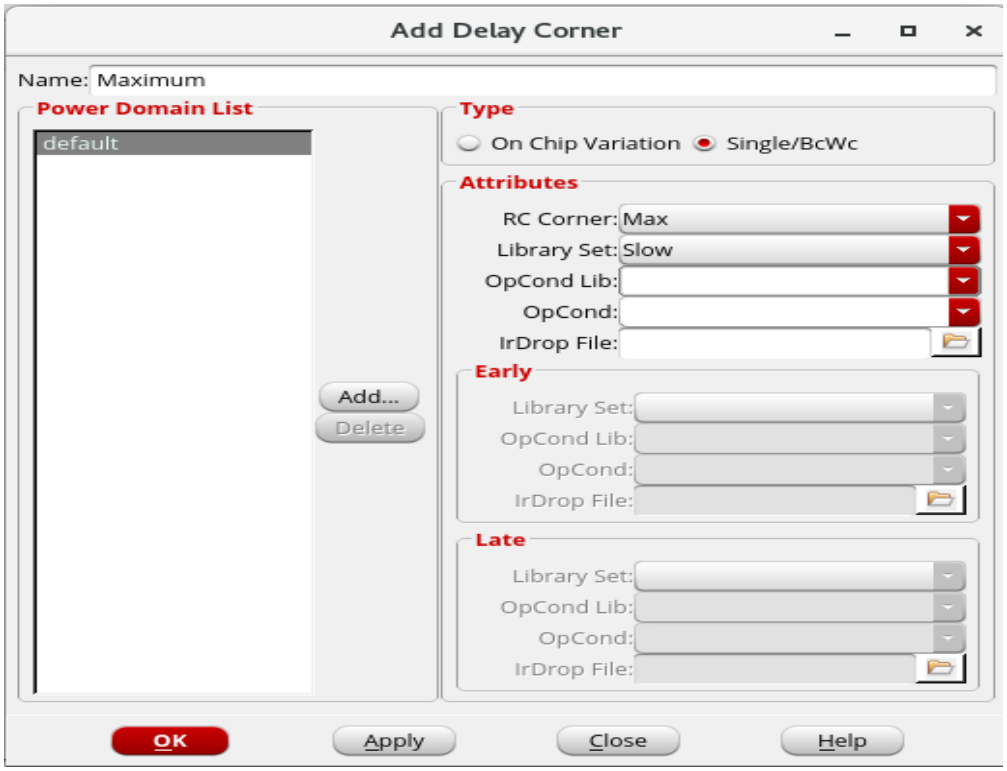
ets"



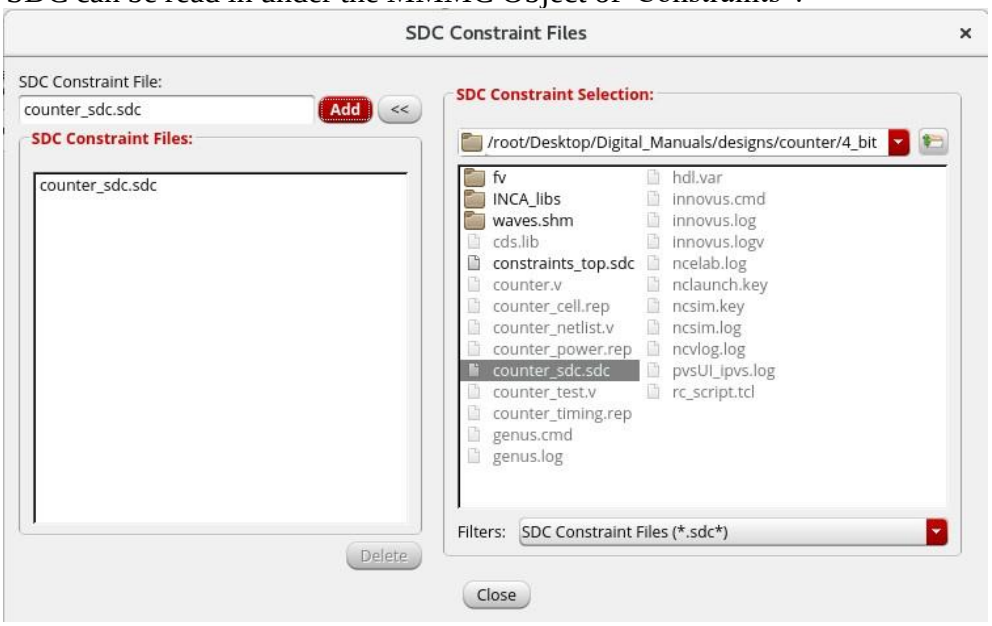
- Similarly, add fast.lib with a label Fast or anyidentifier of your own.
- Adding RC Corners can also be done in a similar process. The temperature value can be found under the corresponding liberty file.Also, cap table and RC Tech files can be added from Foundry whereavailable.

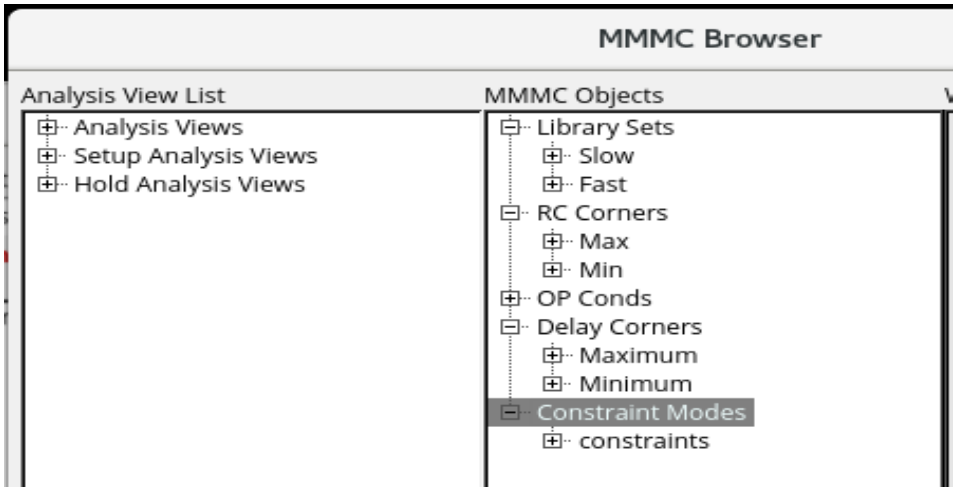
- Delay Corners are formed by combining LibrarySets with RC Corners.
- An example is shown below.



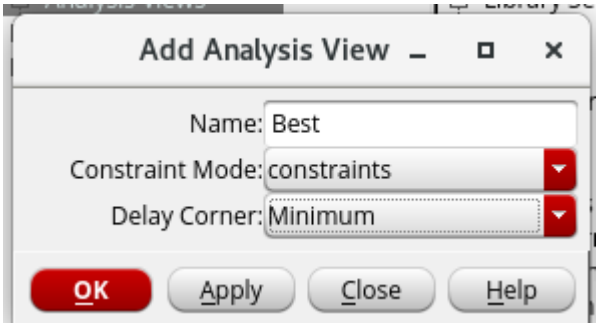


- Similarly, SDC can be read in under the MMMC Object of"Constraints".

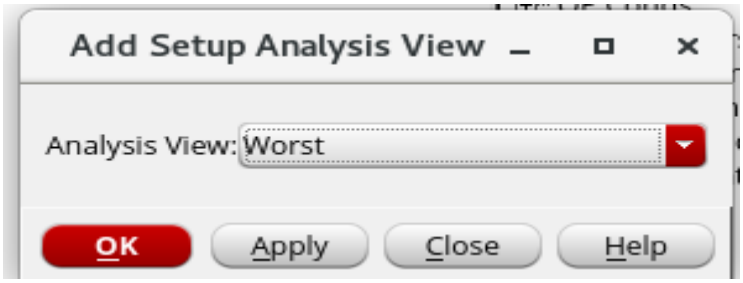


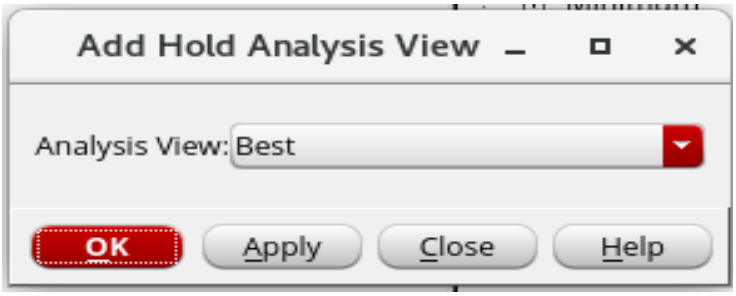


- Analysis Views are formed from combinations ofSDC and Delay Corner.



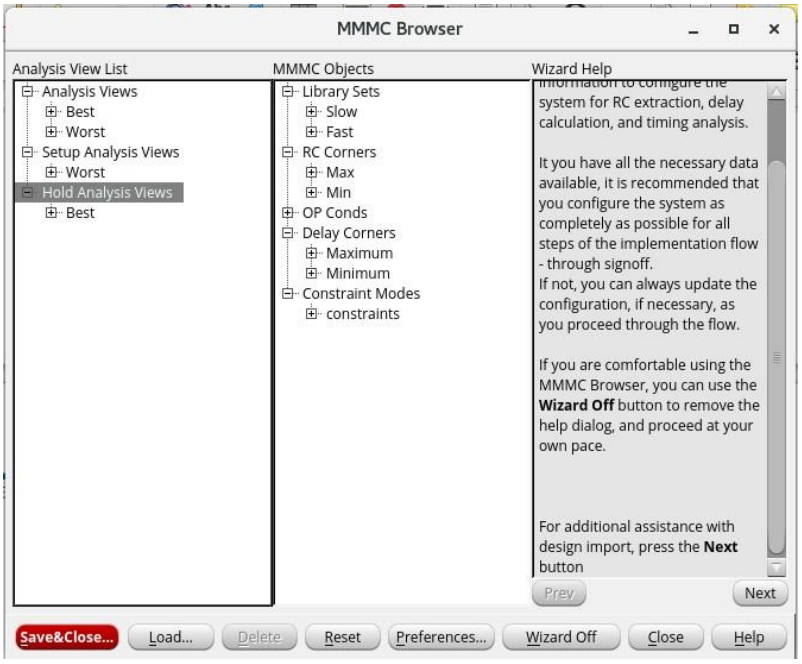
- Once “Best” and “Worst” Analysis views are created, assign them toSetup and Hold.

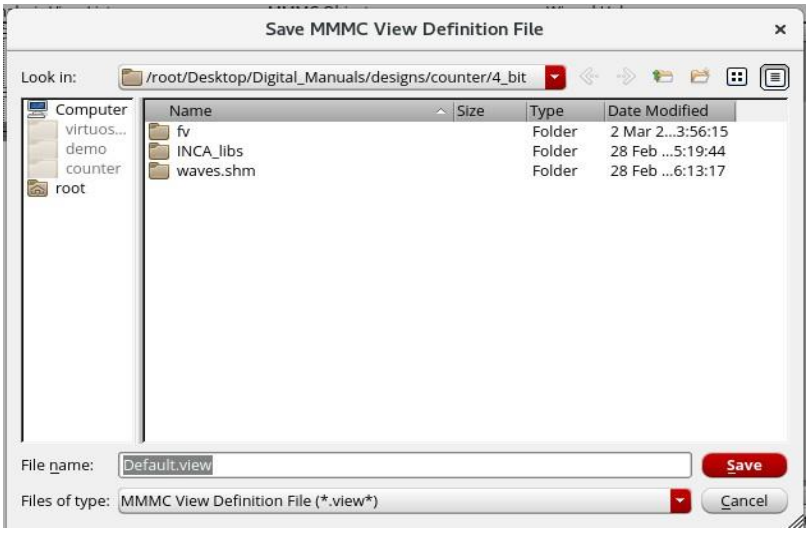




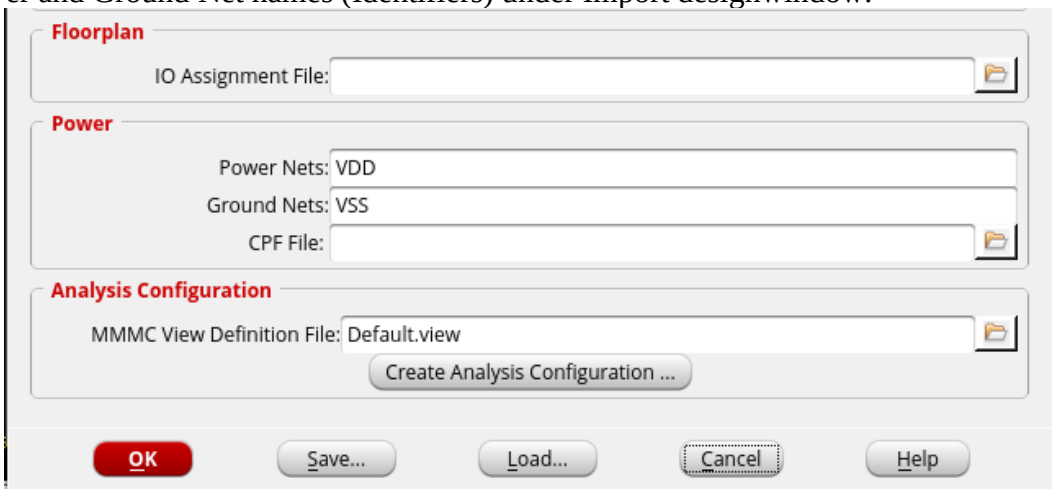
Once all the process is done, Click on "Save&Close" and save the script generated with any name of your choice.

- Make sure the file extension remains .view or .tcl
- After saving the script, go back to Import Design window and Click"OK" to load your design.




Save MMMC View Definition File
Look in:
/root/Desktop/Digital_Manuals/designs/counter/4_bit
Computer
virtuos...
demo
counter
root
Name
Size
Type
Date Modified
fv
Folder
2 Mar 2...3:56:15
INCA_libs
Folder
28 Feb ...5:19:44
waves.shm
Folder
28 Feb ...6:13:17
File name:
Default.view
Save
Files of type:
MMMC View Definition File (*.view*)
Cancel

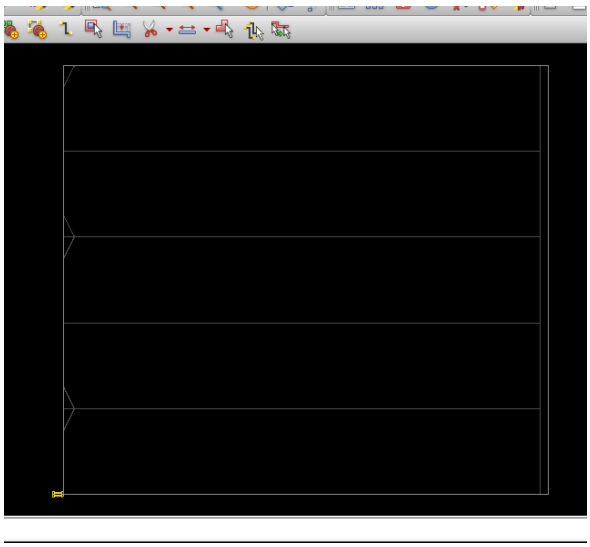
- Add Power and Ground Net names (Identifiers) under Import designwindow.



- A rectangular or square box appears in your GUI if and only if all theinputs are read properly.

If the box does not appear, check for errors in your log (Either onterminal or log file from pwd)

- The internal area of the box is called "Core Area".The horizontal lines running along the width of Core are "Standard Cell Rows". Every alternate of them are marked indicating alternateVDD and VSS rows.
- This setup is called "**Flipped Standard Cell Rows**".

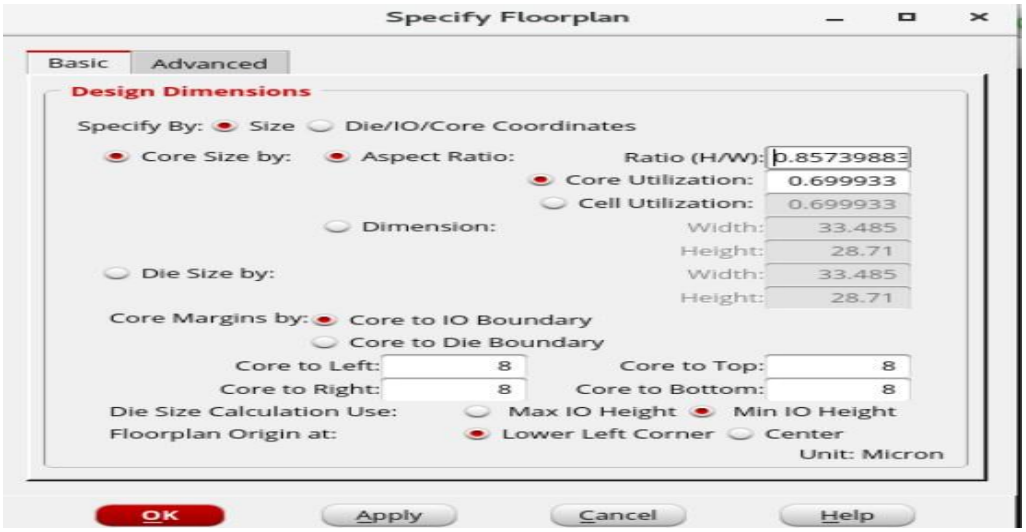


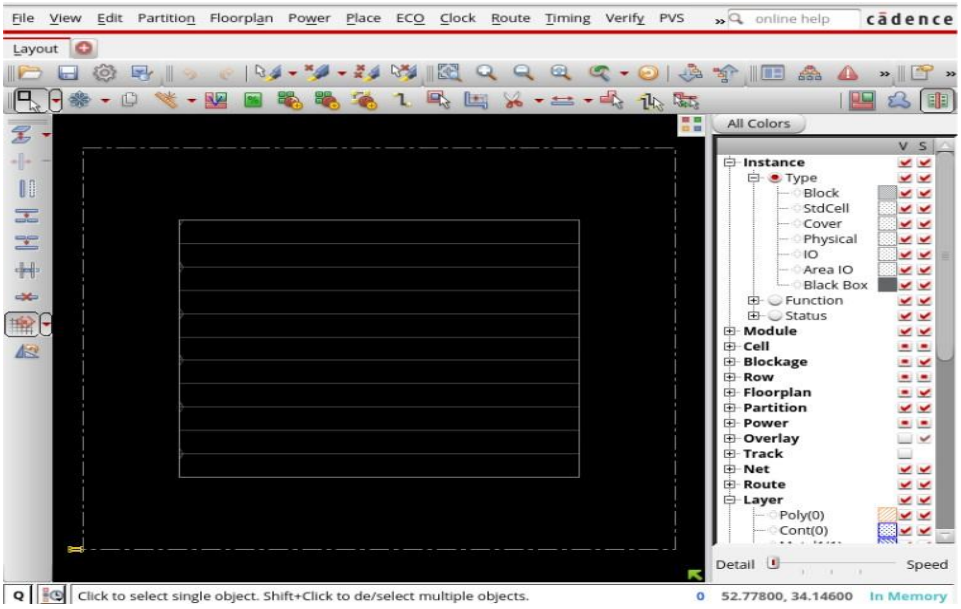
→ **Floorplan**

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 FloorPlanning]
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 alsomodified.





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

→ **Power Planning**

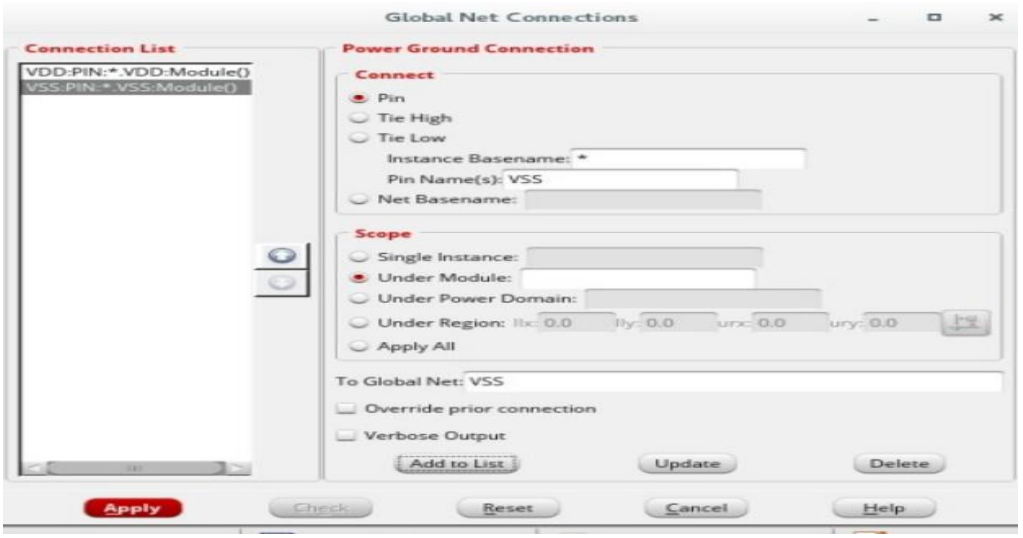
Steps under Power Planning :

1. Connect Global Net Connects
2. Adding Power Rings

3. Adding Power Strings
4. Special Route

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

1. Select Power → Connect Global Nets.. to create “Pin” and “Connectto Global Net” as shown and use “Add to list”.
2. Click on “Apply” to direct the tool in enforcing the Pins and Netconnects to Design and then Close the window.



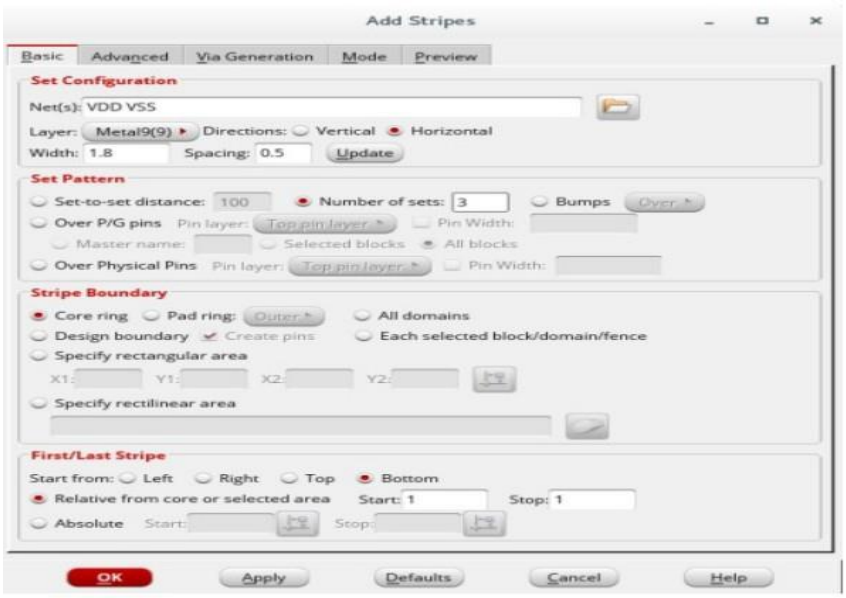
- In order to Tap in Power from a distant Power supply, Wider Nets and Parallel connections improve

efficiency. Moreover, the cells thatwould be placed inside the core area are expected to have shorter Nets for lower resistance.

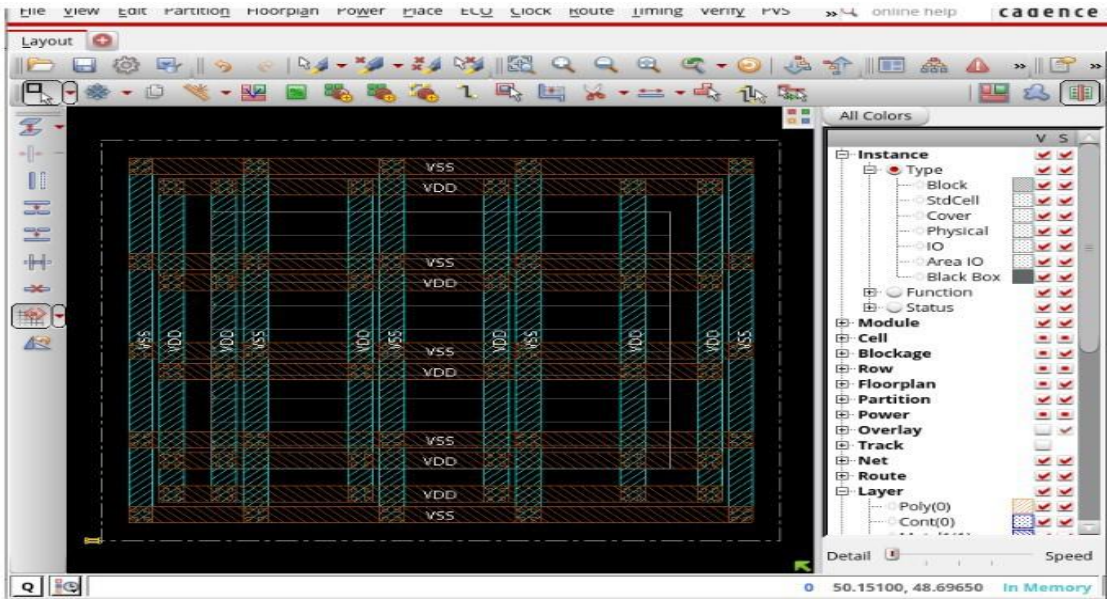
- Hence Power Rings [Around Core Boundary] and Power Stripes[Across Core Boundary] are added which satisfies the above conditions.
- Select Power → Power Planning → Add Rings to add Power rings'around Core Boundary'.

- Select the Nets from Browse option OR Directly type in the GlobalNet Names separated by a space being Case and Spelling Sensitive.
- 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 Widthsand thus Lowest Resistance.

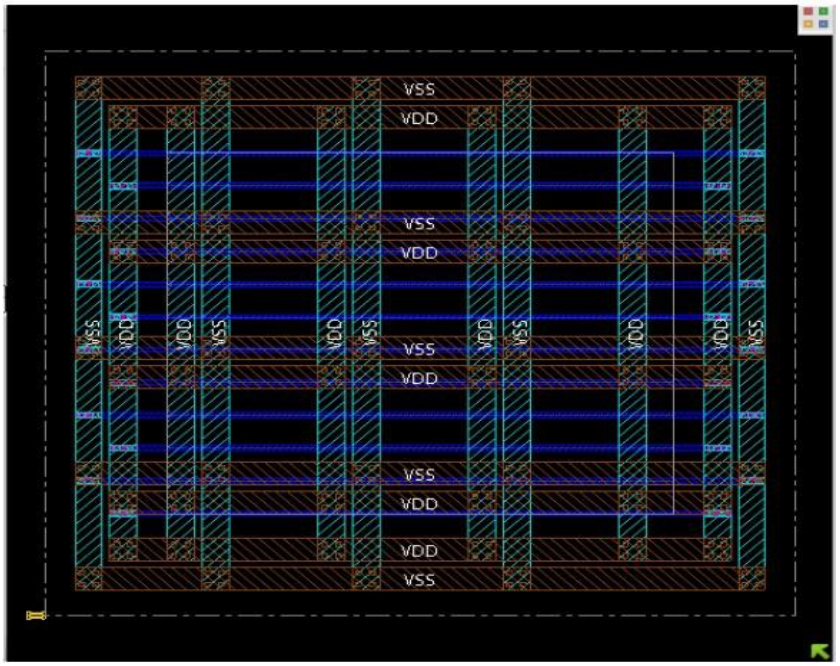
- Click on Update after the selection and "Set Offset : Centre in Channel" in order to get the Minimum Width and Minimum Spacingof the corresponding Metals and then Click "OK".
  - Similarly, Power Stripes are added using similar content to that ofPower Rings.



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 orMetal 8 directly with Metal 1 [VDD or VSS of Standard Cells aregenerally made up of Metal 1].
- The connection between the Highest and Lowest Metals is donethrough Stacking of Vias done using "Special Route".
- To perform Special Route, Select Route → Special Route → AddNets → OK.
- After the Special Route is complete, all the Standard Cell Rows turnto the Color coded for Metal 1 as shown below.



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

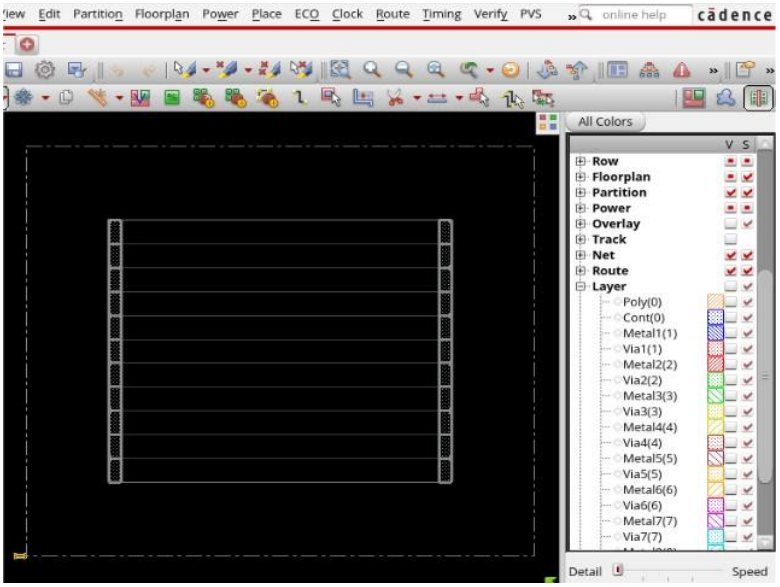
→ **Pre – Placement** :

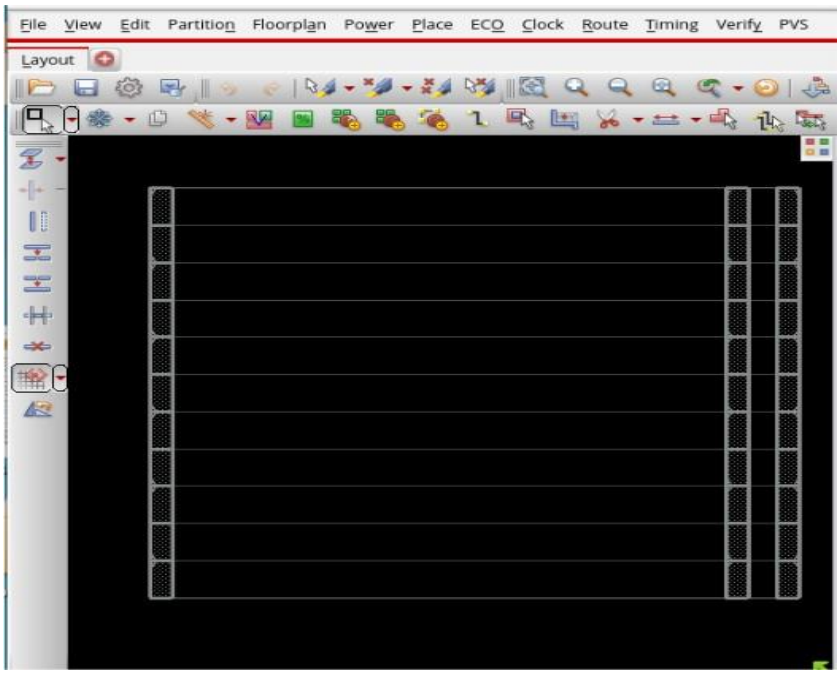
- After Power Planning, a few Physical Cells are added namely, EndCaps and Well Taps.

End Caps : They are Physical Cells which are added to the Left andRight Core Boundaries acting as blockages to avoid Standard Cellsfrom moving out of boundary.
Well Taps : They act like Shunt Resistance to avoid Latch Upeffects.
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 Capsare added below your Power Mesh.

- 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





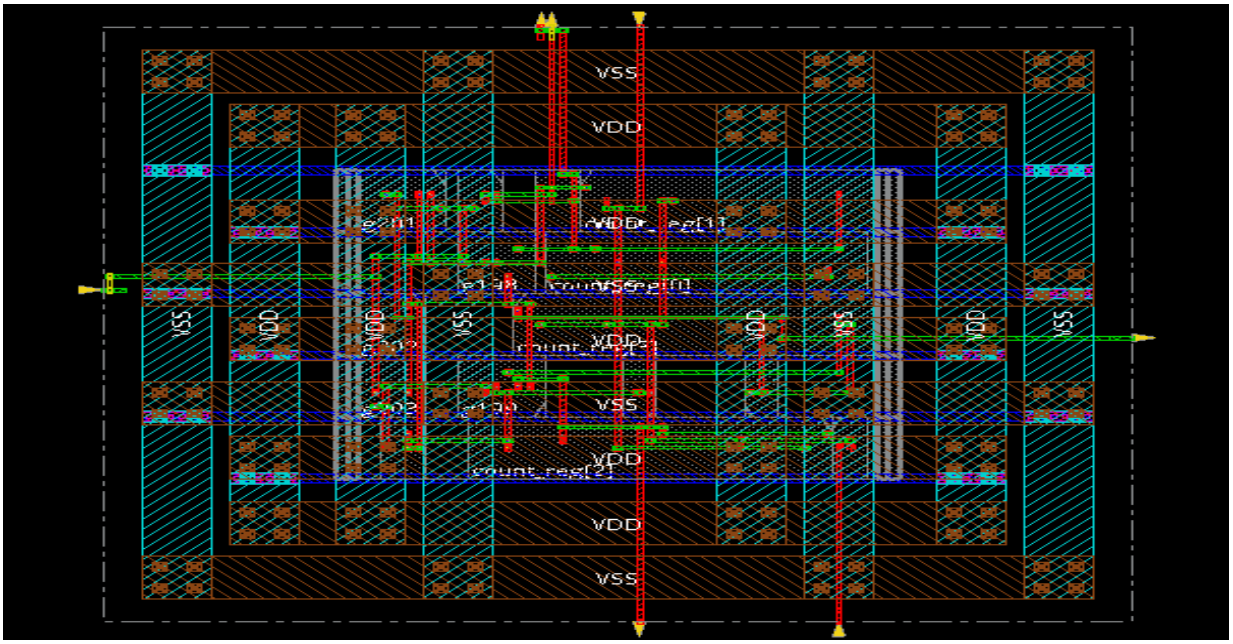
→ Placement

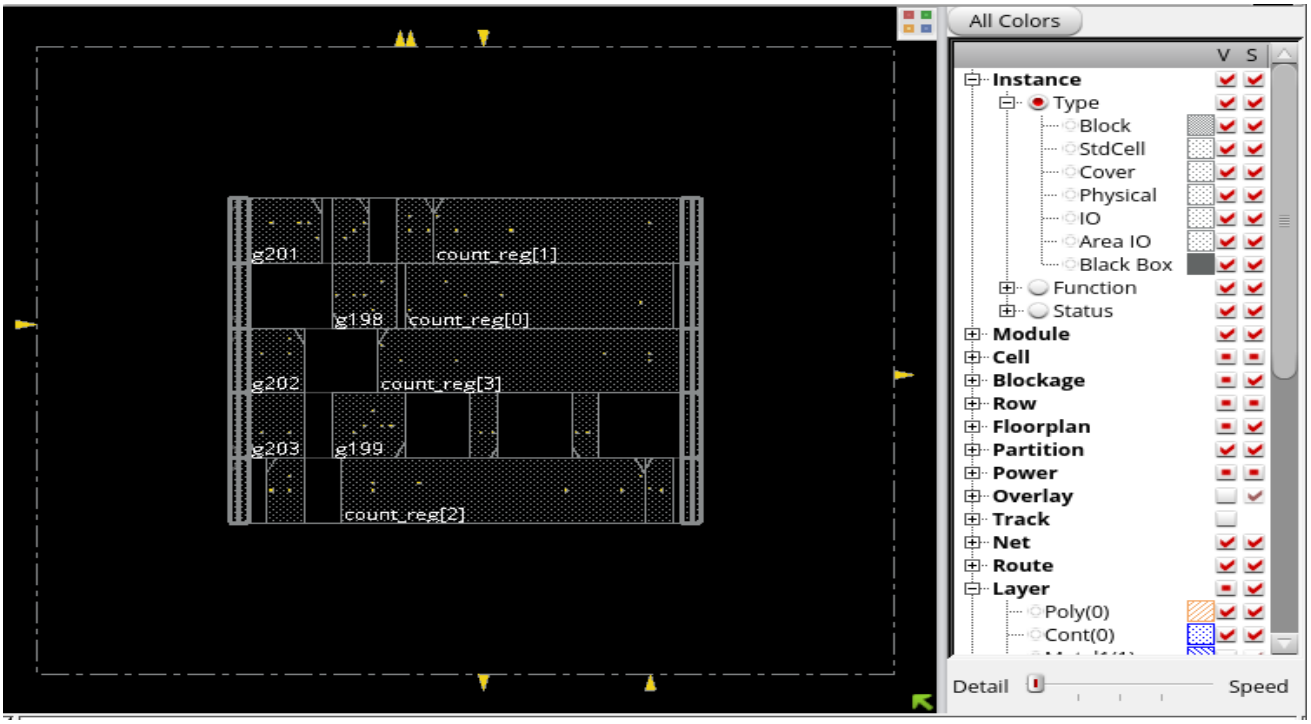
1. The Placement stage deals with Placing of Standard Cells as well asPins.
2. Select Place → Place Standard Cell → Run Full Placement → Mode

→ Enable 'Place I/O Pins' → OK → OK .

• All the Standard Cells and Pins are placed as per the communicationbetween them, i.e., Two communicating Cells are placed as close aspossible so that shorter Net lengths can be used for

connections as Shorter Net Lengths enable Better Timing Results.





You can toggle the Layer Visibility from the list on the Right. TheList of Layers available are shown on the right under “Layer” tabwith colour coding.

→ Report Generation and Optimization :

1. Timing Report :
    1. To generate Timing Report, Timing → Report Timing → DesignStage – PreCTS

2. Analysis Type – Setup → OK
3. The Timing report Summary can be seen on the Terminal.

2. Area Report :
   1. **cmd :** report_area
3. Power Report :
   1. **cmd :** report_power

```
innovus 9> innovus 9> report_area
Depth  Name       #Inst  Area (um^2)
-----------------------------------------
0      counter    15     134.7282
-
```

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

Setup views included:
 Worst

+--------------------+---------+---------+---------+
|     Setup mode     |   all   | reg2reg | default |
+--------------------+---------+---------+---------+
|           WNS (ns):|  0.605  |  0.845  |  0.605  |
|           TNS (ns):|  0.000  |  0.000  |  0.000  |
|    Violating Paths:|    0    |    0    |    0    |
|          All Paths:|   16    |    8    |   11    |
+--------------------+---------+---------+---------+

+----------------+--------------------------------+------------------+
|                |              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)       |
+----------------+------------------+-------------+------------------+
```

```
*       Power View : Worst
*
*       User-Defined Activity : N.A.
*
*       Activity File: N.A.
*
*       Hierarchical Global Activity: N.A.
*
*       Global Activity: N.A.
*
*       Sequential Element Activity: N.A.
*
*       Primary Input Activity: 0.200000
*
*       Default icg ratio: N.A.
*
*       Global Comb ClockGate Ratio: N.A.
*
*       Power Units = 1mW
*
*       Time Units = 1e-09 secs
*
*       report_power
*
-----------------------------------------------------------------------------------------


Total Power
-----------------------------------------------------------------------------------------
Total Internal Power:        0.04747067            90.0866%
Total Switching Power:       0.00449045             8.5217%
Total Leakage Power:         0.00073336             1.3917%
Total Power:                 0.05269448
```

- In case of any Violating paths, the design could be optimized in thefollowing way.
- To optimize the Design, **Select ECO → Optimize Design → DesignStage [PreCTS] → Optimization Type – Setup → OK**

- After you run the optimization, the terminal displays the latest Timing report and updated area and power reports can be checked.

- 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.

Clock Tree Synthesis

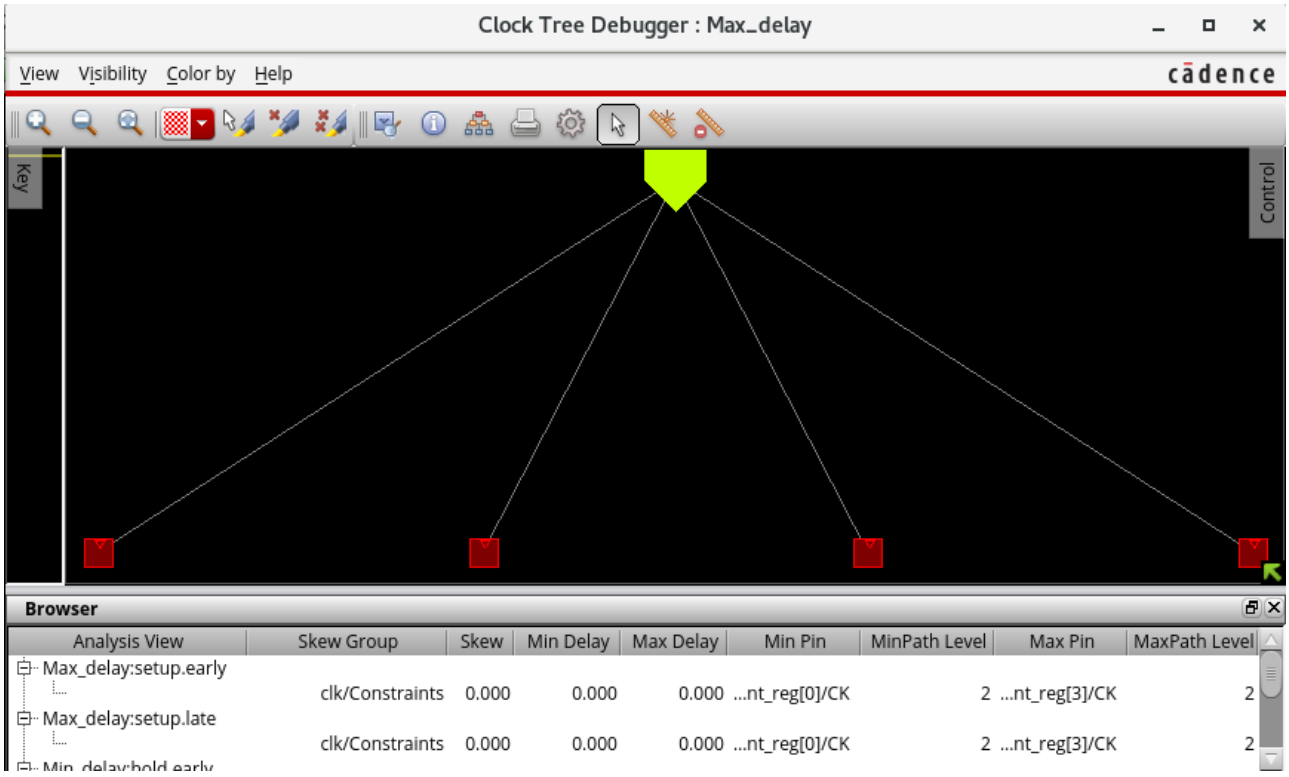
- The CTS Stage is meant to build a Clock Distribution Network suchthat every Register (Flip Flop) acquires Clock at the same time (Atleast Approximately) to keep them in proper communication.
- A Script can be used to Build the Clock Tree as follows :

```
extractRC

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
```

- 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.

```
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.
```

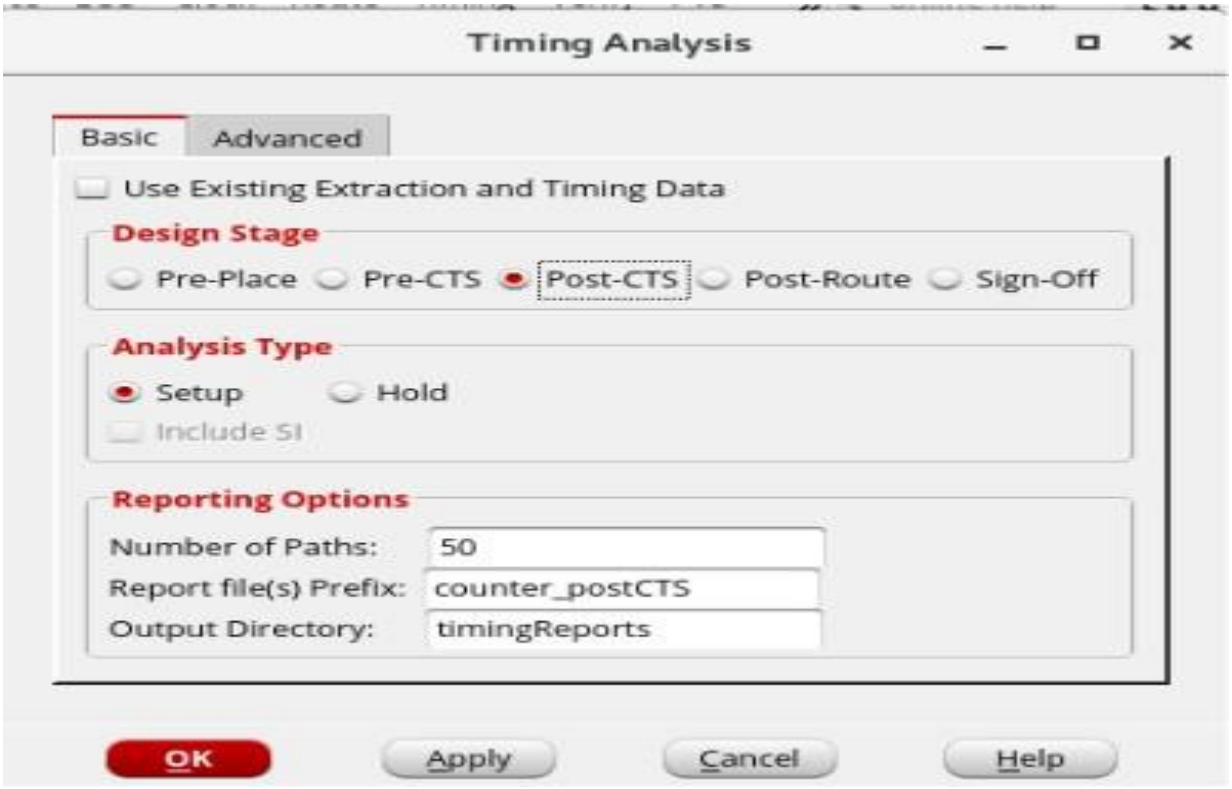


- The Red Boxes are the Clock Pins of various Flip Flops in the Design while Yellow Pentagon on the top represents Clock Source.
- The Clock Tree is built with Clock Buffers and Clock Invertersadded to boost up the Clock Signal.

Report Generation and Design Optimization :

- CTS Stage adds real clock into the Design and hence "Hold" Analysis also becomes prominent. Hence, Optimizations can be donefor both Setup & Hold, Timing Reports are to be Generated for Setupand Hold Individually.

Setup Timing Analysis :

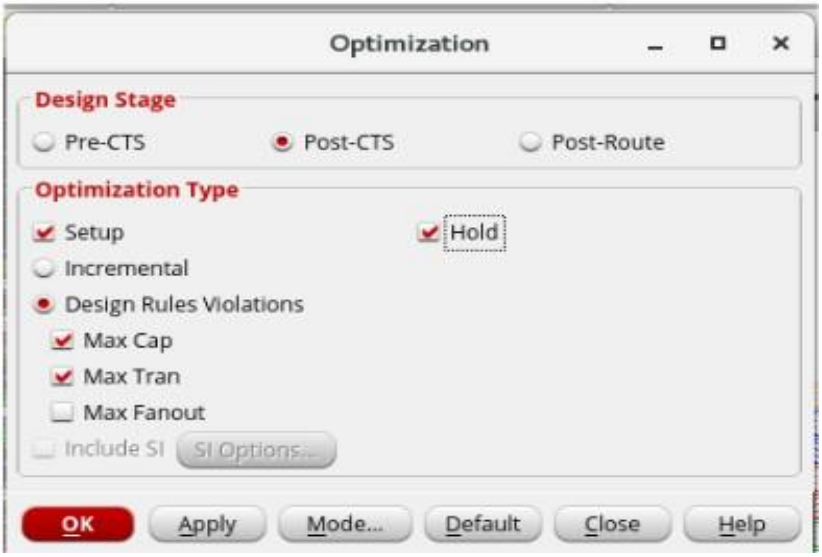


Hold Timing Analysis :

For Area and Power Report Generation,

report_area & report_power commands can be used.

Design Optimizations :

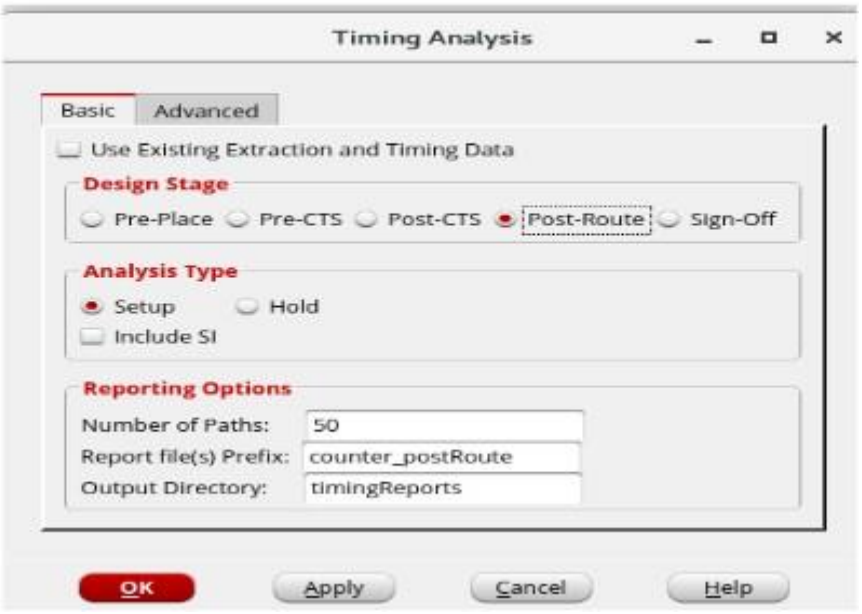


Routing :

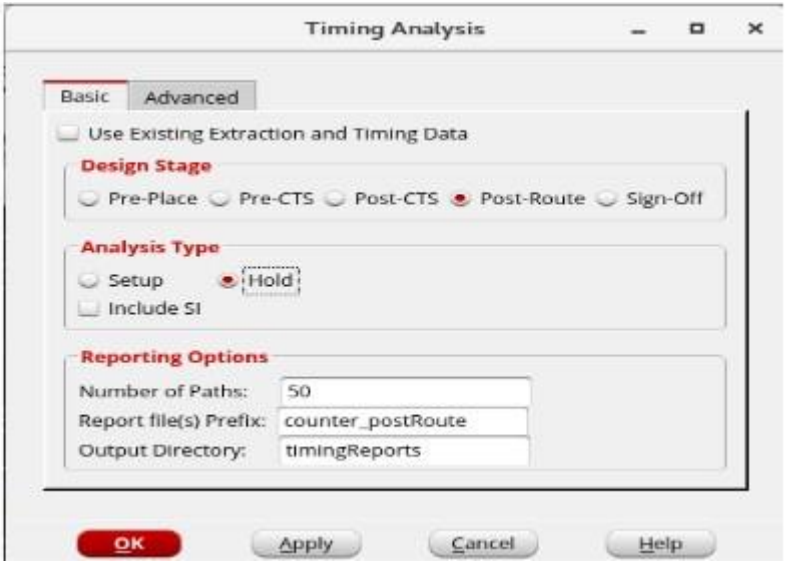
1. All the net connections shown in the GUI till CTS are only based onthe Logical connectivity.
2. These connections are to be replaced with real Metals avoiding Opens, Shorts, Signal Integrity [Cross Talks], Antenna Violations etc.
3. To run Routing, Select Route → Nano Route → Route and enableTiming Driven and SI Driven for Design Physical Efficiency and Reliability.

Report Generation and Design Optimization :

Setup Report :



Hold Report :



Area and Power Reports :

Use the commands report_area and report_power for Area and PowerReports respectively.

Design Optimization :

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

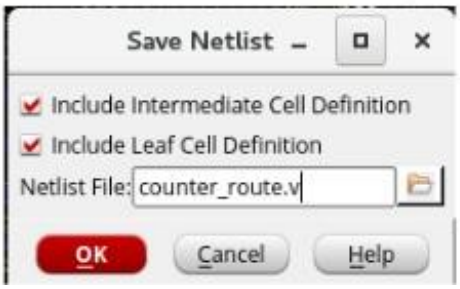
Enter the above shown command in the Terminal in order to run the Design Optimization first Post-Route.

- As an alternate to the setAnalysisMode command, you can use the GUI at Tools → Set Mode → Set Analysis Mode → Select On-Chip-Variation and CPPR.
- The Report generation is same as shown prior to DesignOptimization.

Saving Database :

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

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

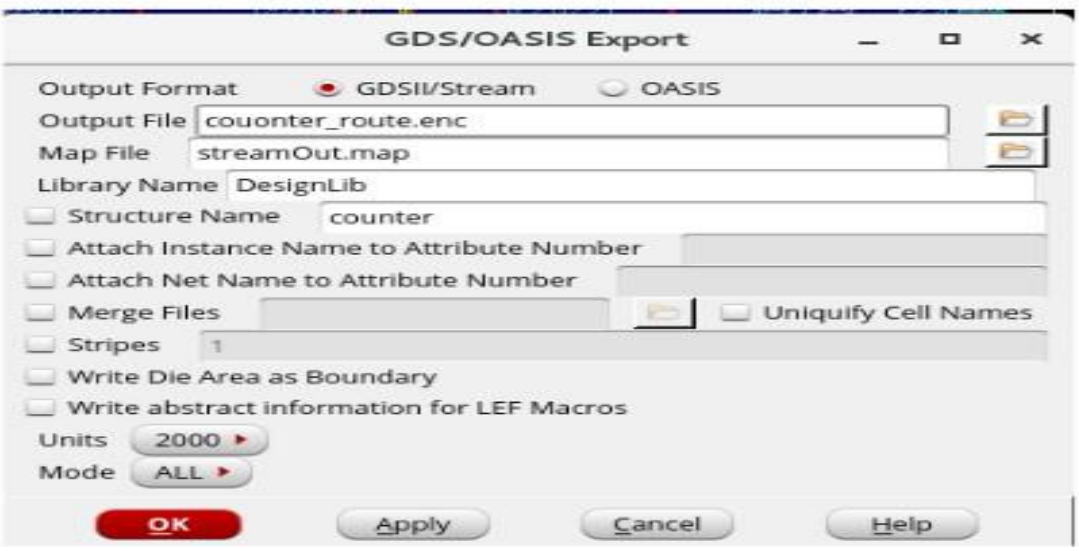


It is recommended to save Netlist and Design at every stage.

To restore a Design Data Base, type source <DesignName>.enc in theterminal.

GDS File Generation

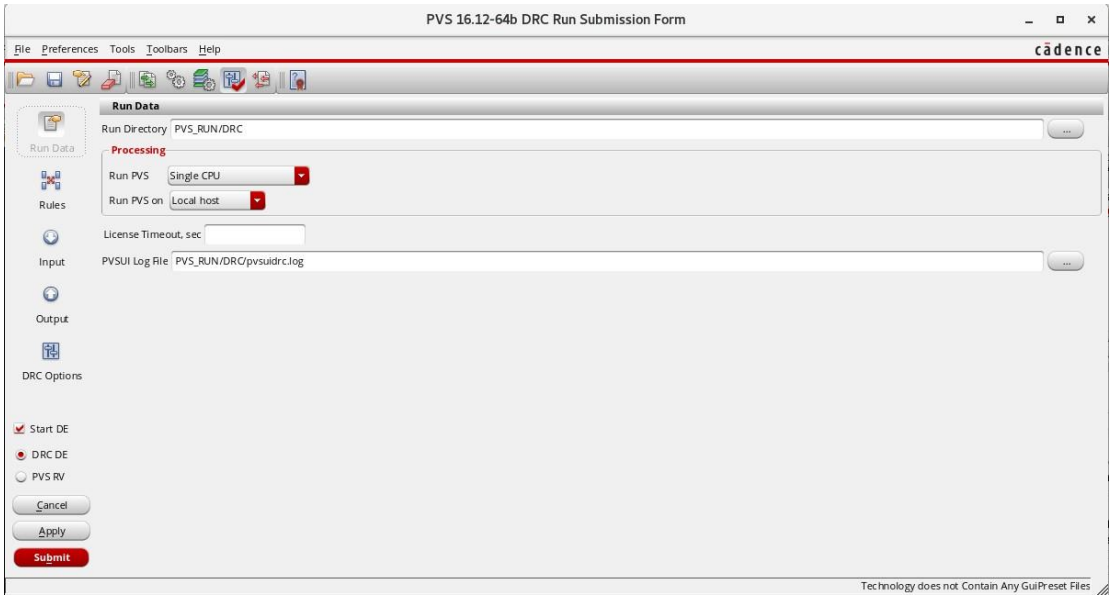
3. Saving GDS => File → Save → GDS/OASIS → <FileName>.gds → OK



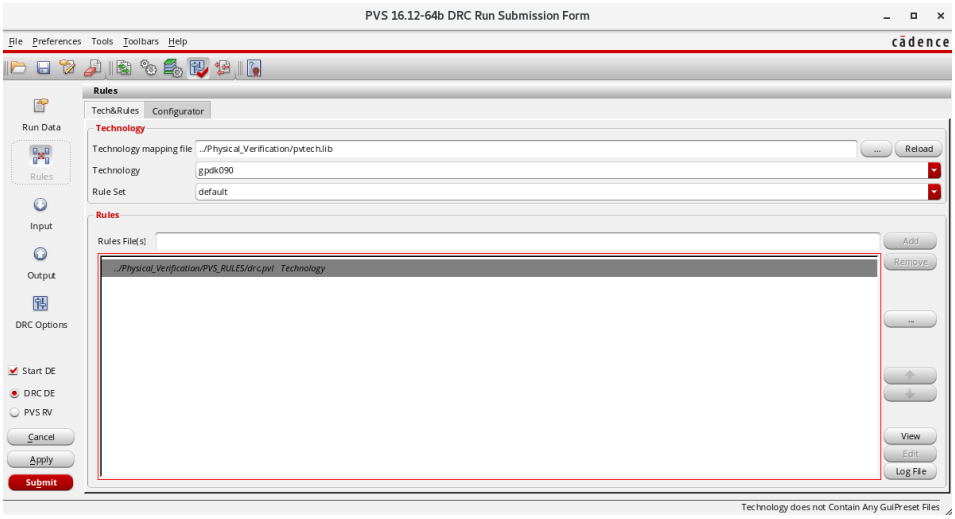
Physical Verification – Capturing DRC and LVS :

- After saving the routed Database, you can proceed for PhysicalVerification and capture the DRC and LVS reports.
- Inputs Required – DRC :
    - Technology Library and Rule Set
    - GDS format giles of all Standard Cells (Given by Cadence at /home/install/FOUNDRY/90nm/dig/gds for 90nm Tech node)
- Outputs – DRC :
    - DRC Violation Report
    - Physical Netlist (Optional)

From the Innovus GUI, select PVS → Run DRC to open the "DRCSubmission Form".

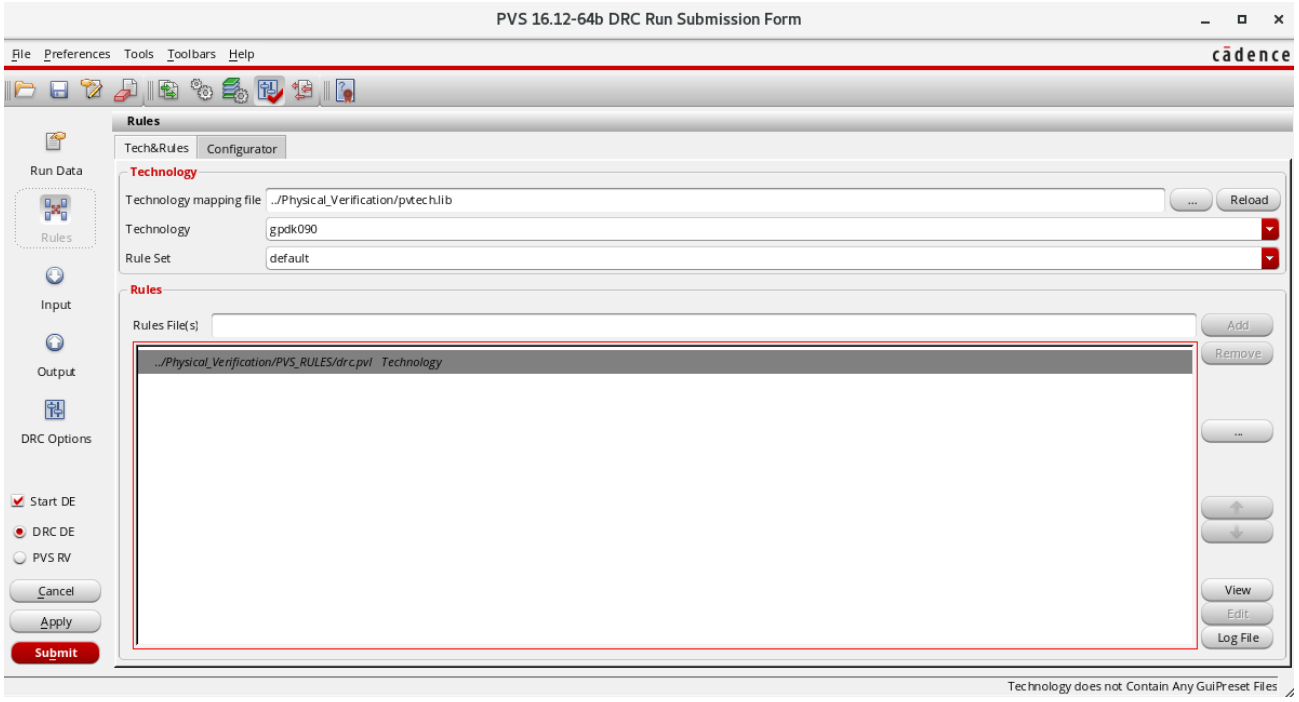


The DRC Run Submission Form begins with mentioning the Run Directory. The Run Directory is the location where all the logs, reports andother files concerned with PVS are saved.

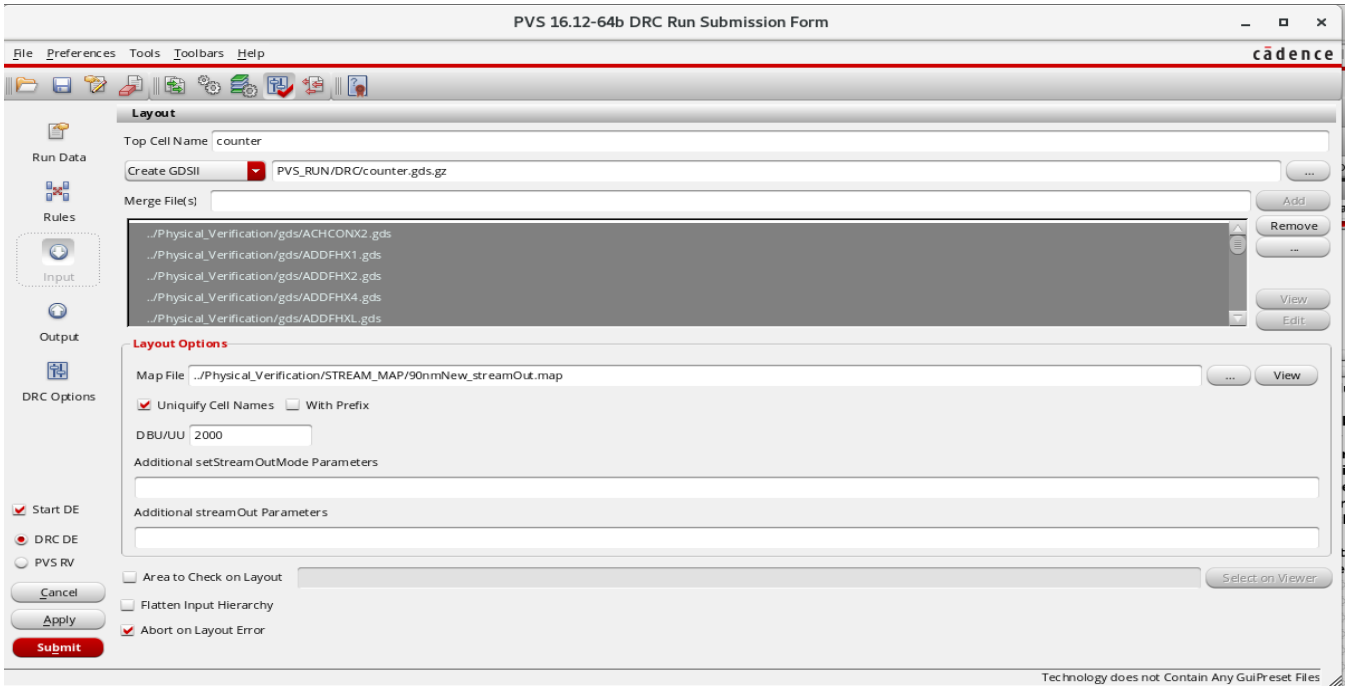


The technology Library is to be loaded under “Rules tab”.

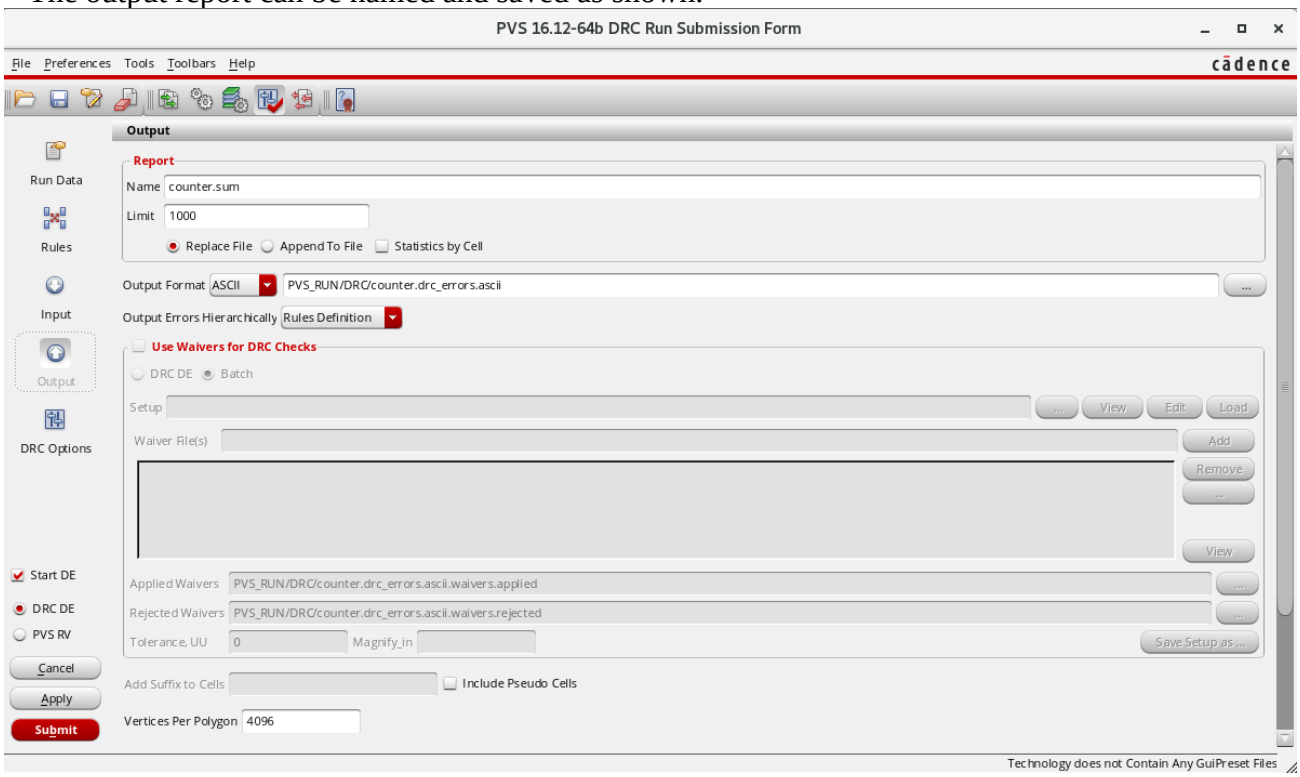
- The Technology Library is specific for PVS Tool and technologynode on which the design is created.
- On reading the tech lib, the rule set is loaded and the correspondingfabrication rules are read in to be checked against the design.



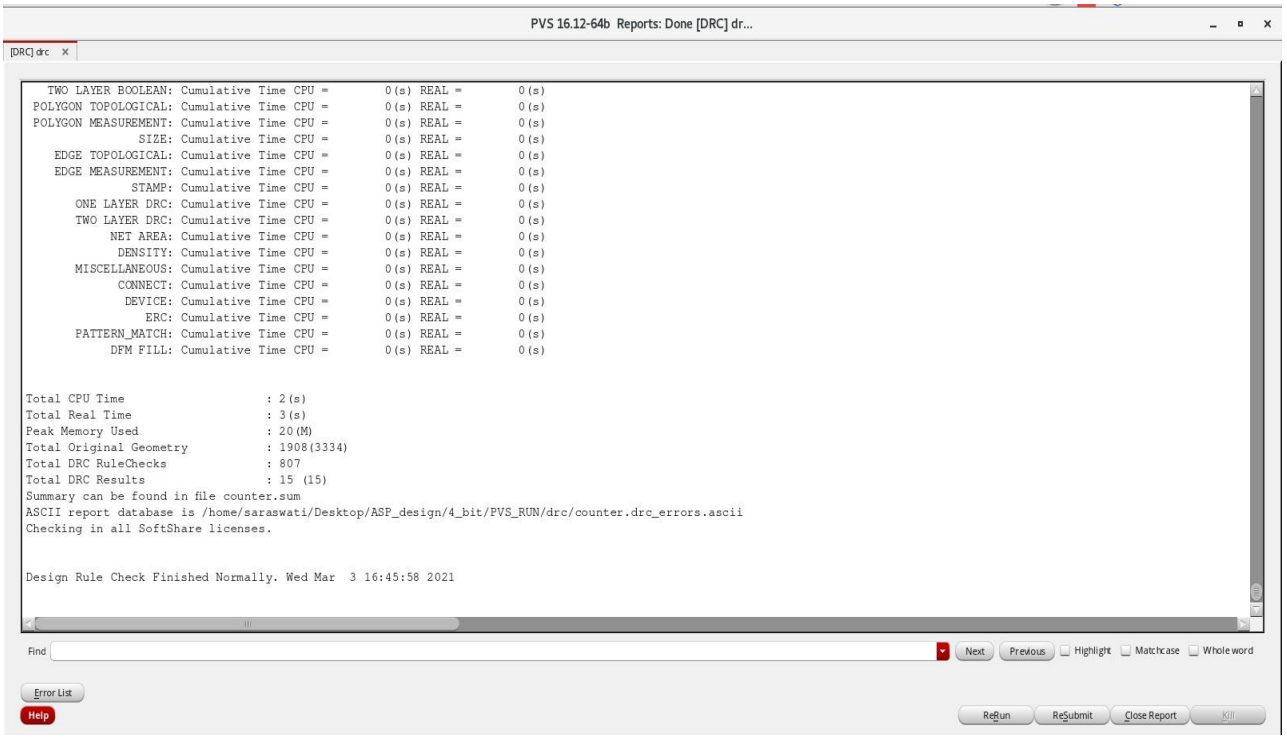
- The GDS format files of all standard cells available with the corresponding technology node are also provided by the vendor.Select all of them to add.

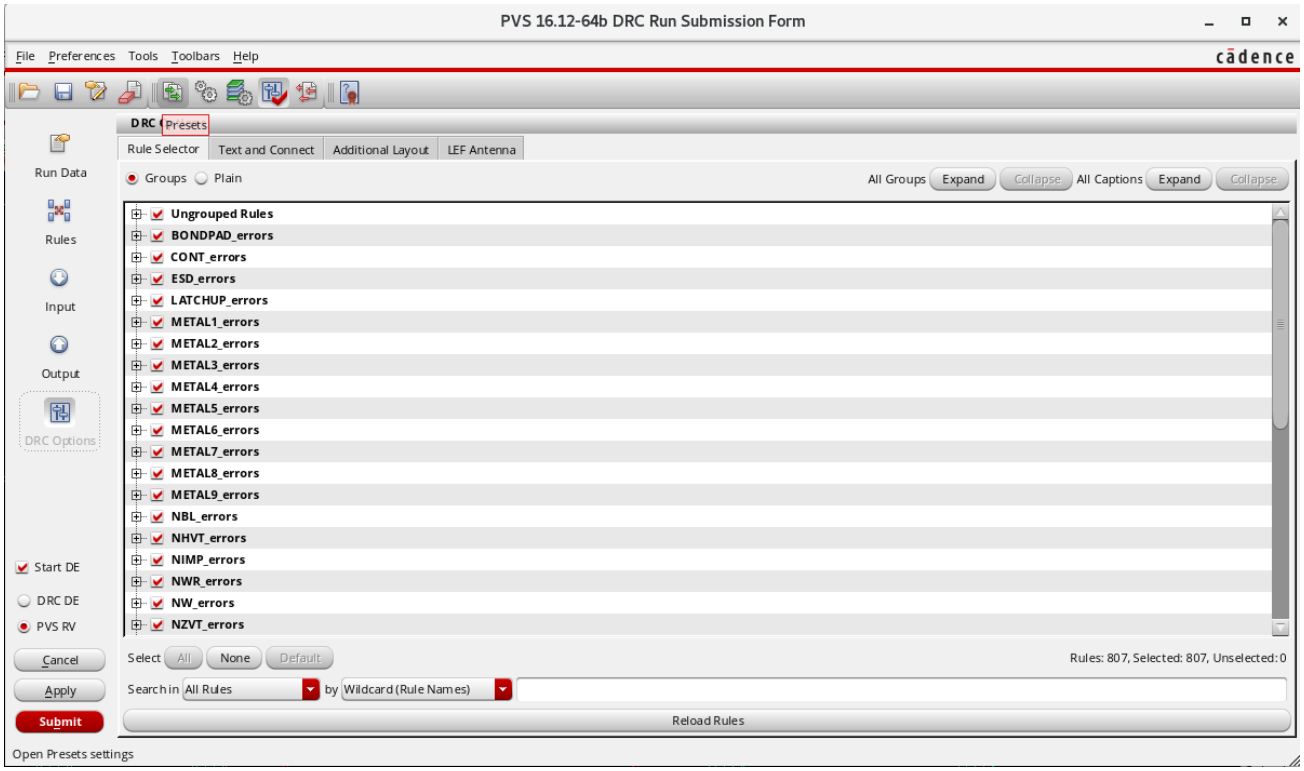


- The output report can be named and saved as shown.

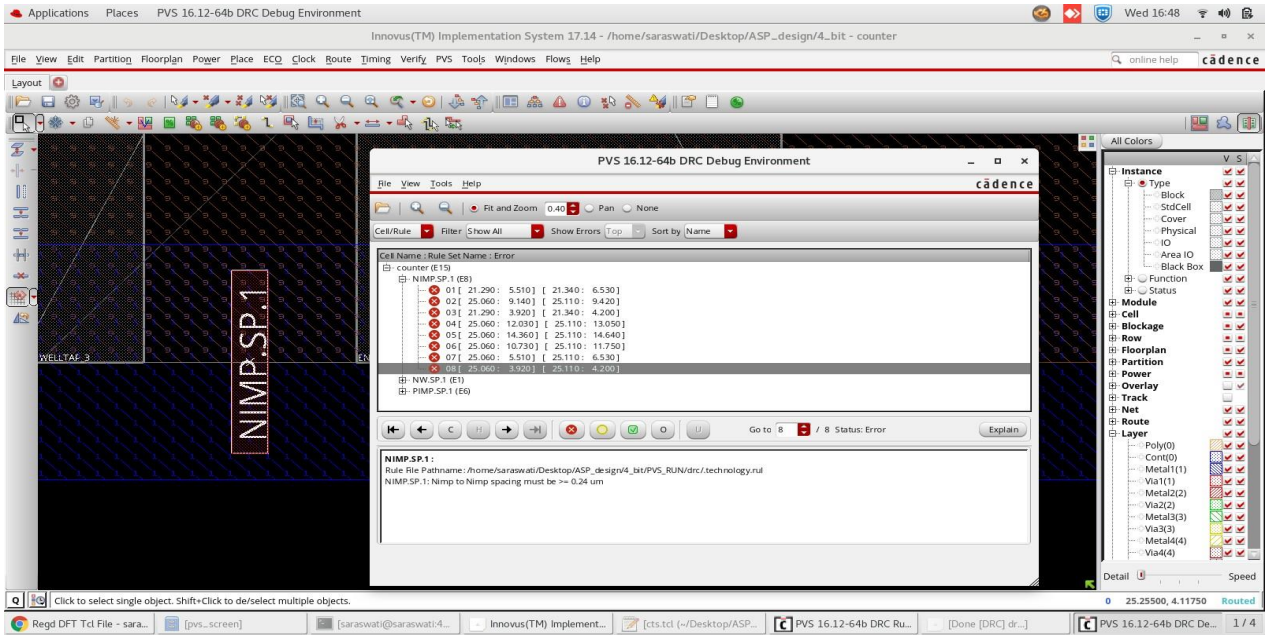


- Hit “Submit” to run the DRC and the following windows appear.

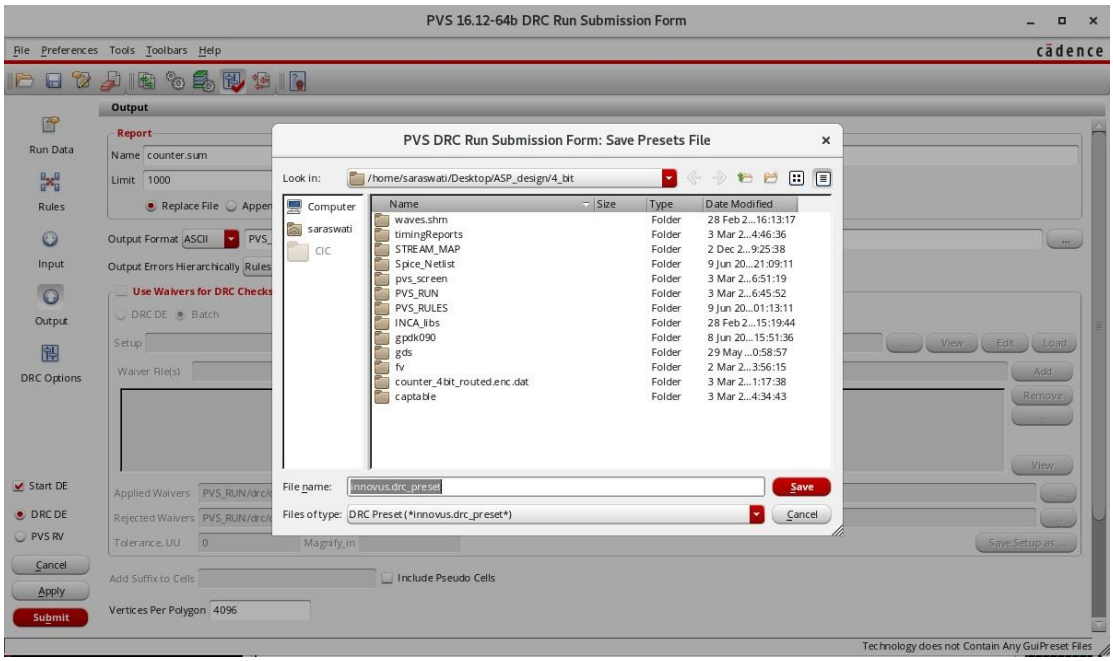




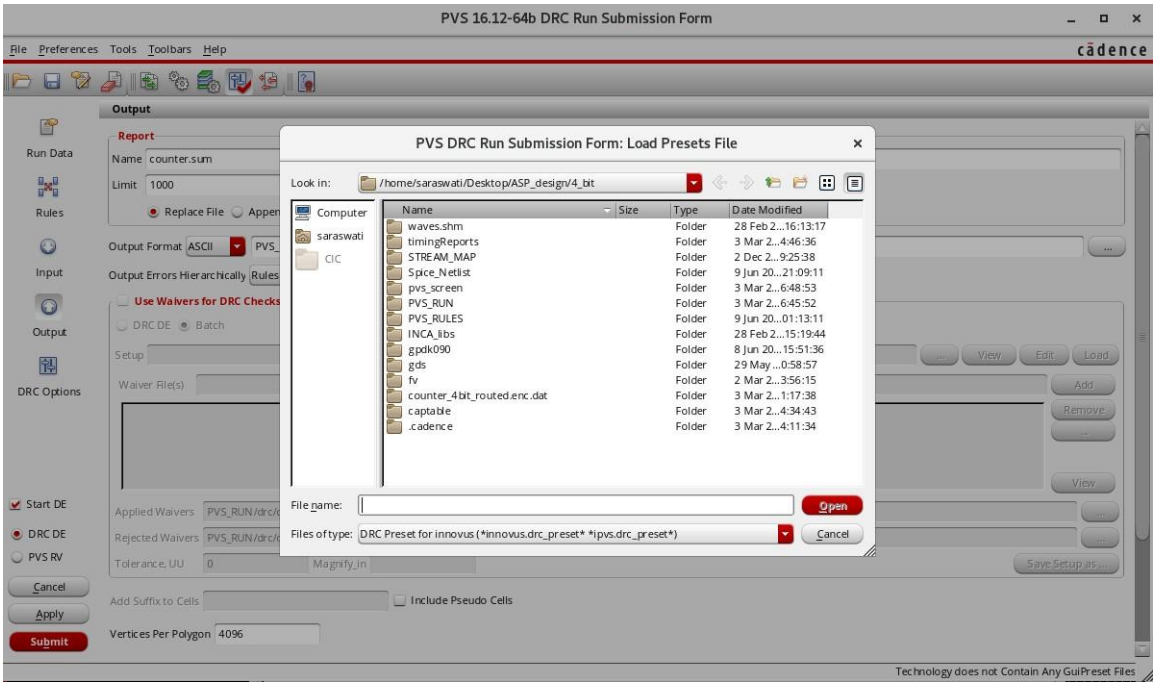
- All the list of DRC Errors can be seen in the above window of whichthe location of the DRC Violation occuring can be highlighted dealing one to one.



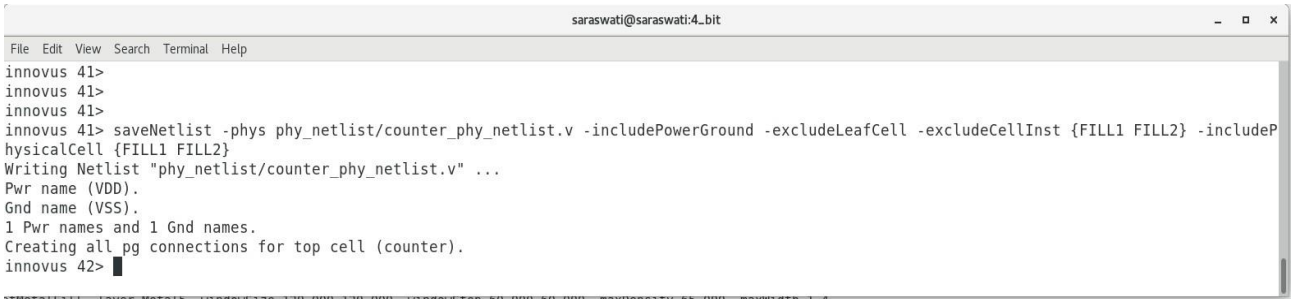
- For example, in the above shown snapshot, the errors associated withN-Implant can be seen. (Select a error occurrence and click on the right arrow below to highlight/zoom in the location.)
- You can save the DRC Run as a “Preset” file to rerun the DRC ifrequired at a later point of time.
- Saving/loading the Preset File is shown below.

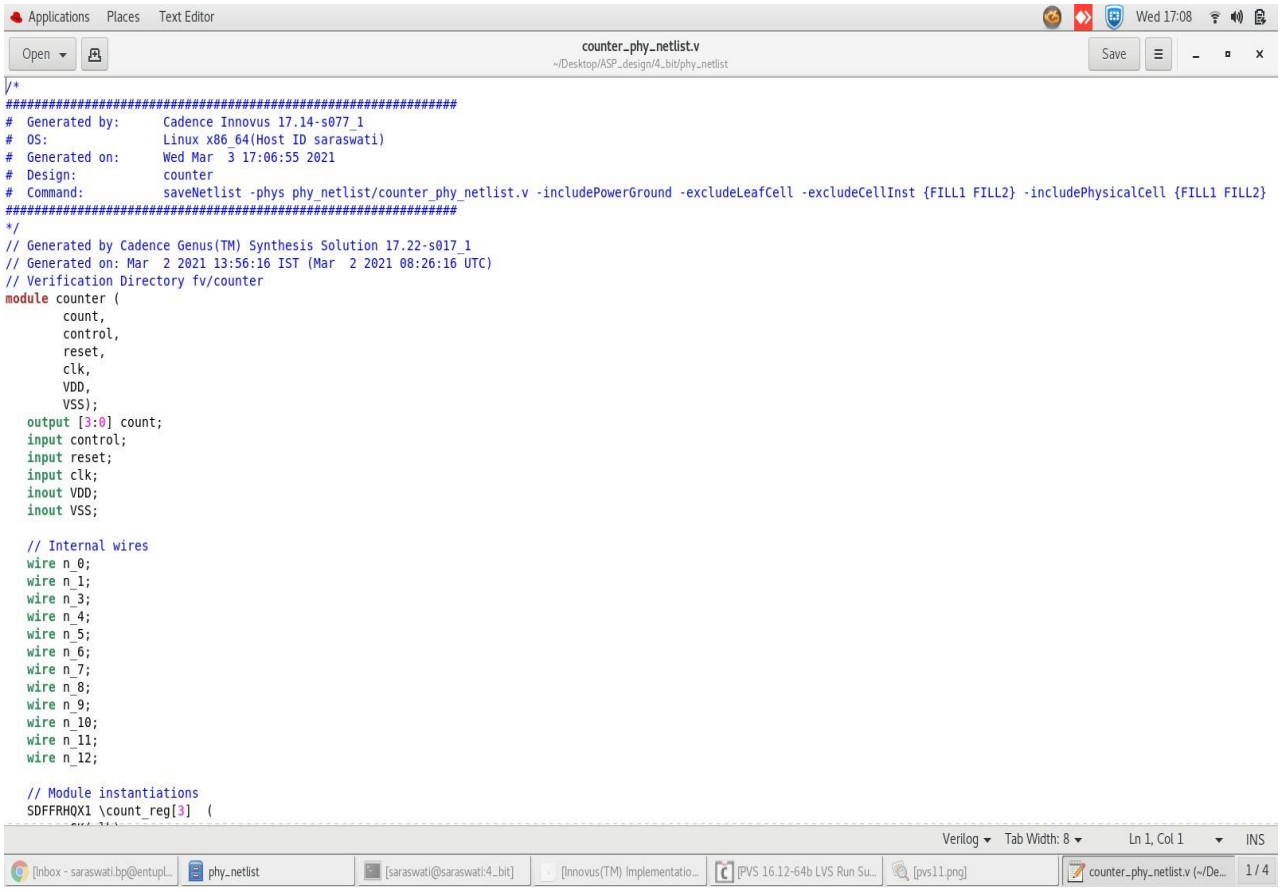


- Loading a Preset file is shown below.



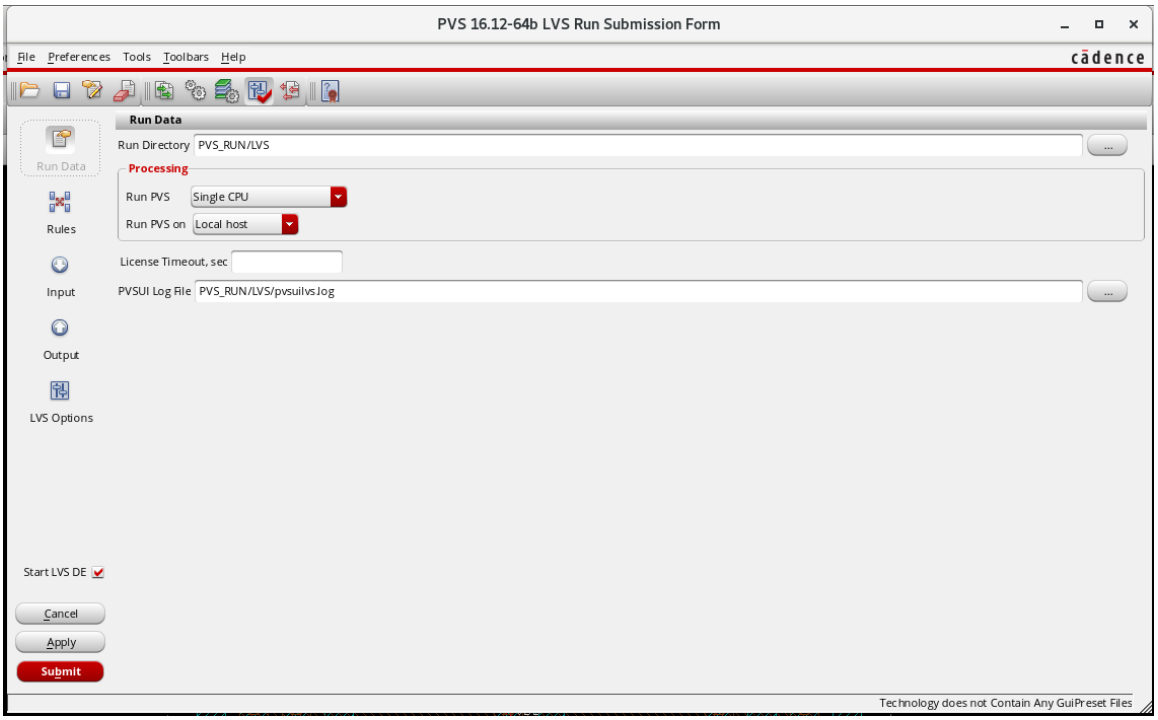
Note : A Physical Netlist can be saved after the DRC Run as shown below.





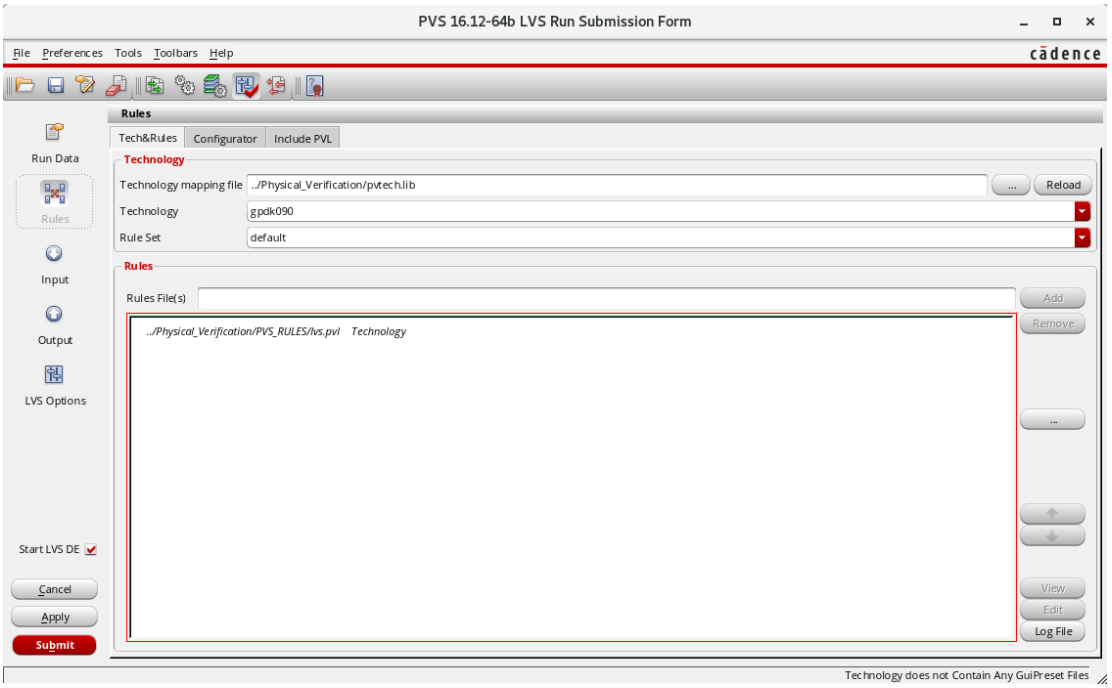
- Inputs Required – LVS :
  - Technology Library
  - Standard Cell GDS Files
  - Spice Netlist of all Standard Cells (Provided by Library Vendor)
- Outputs – LVS :
  - LVS Match/Mismatch Report

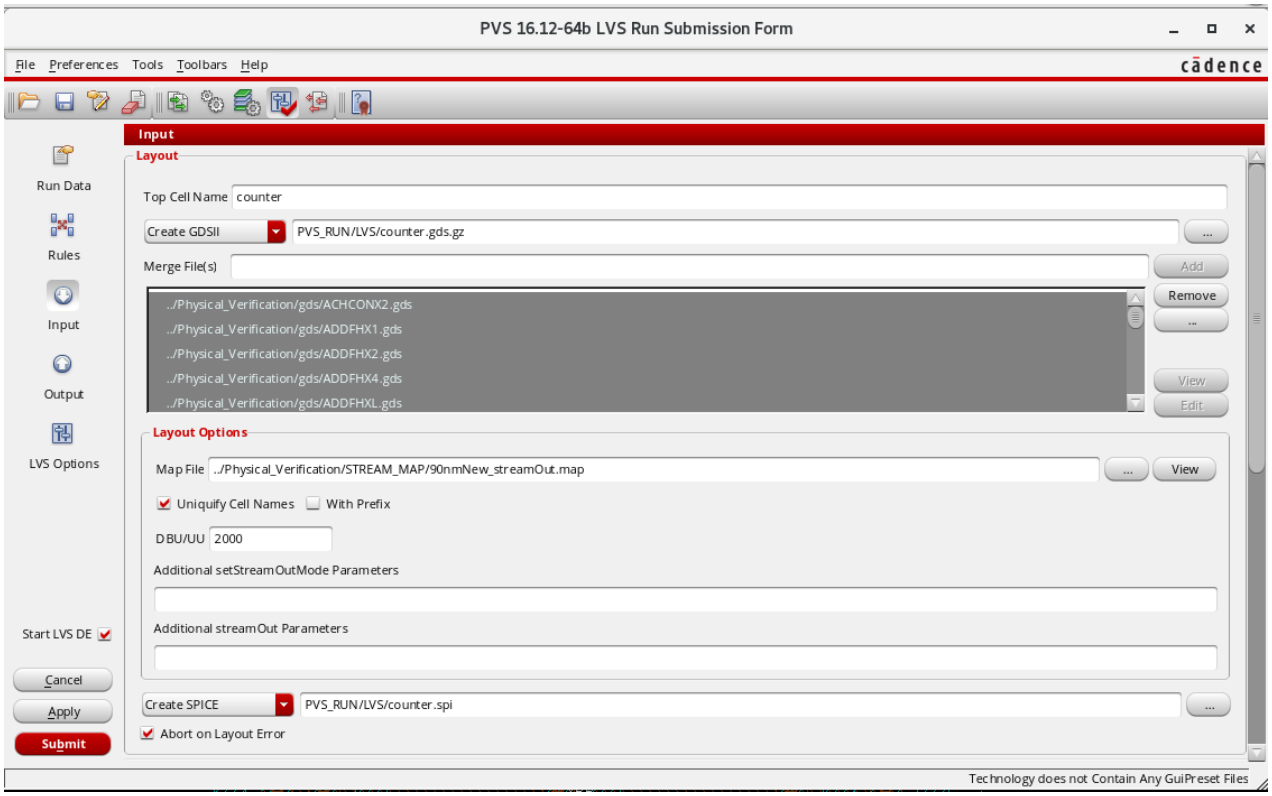
- From the Innovus GUI, Select PVS → Run LVS to open the LVS runsubmission

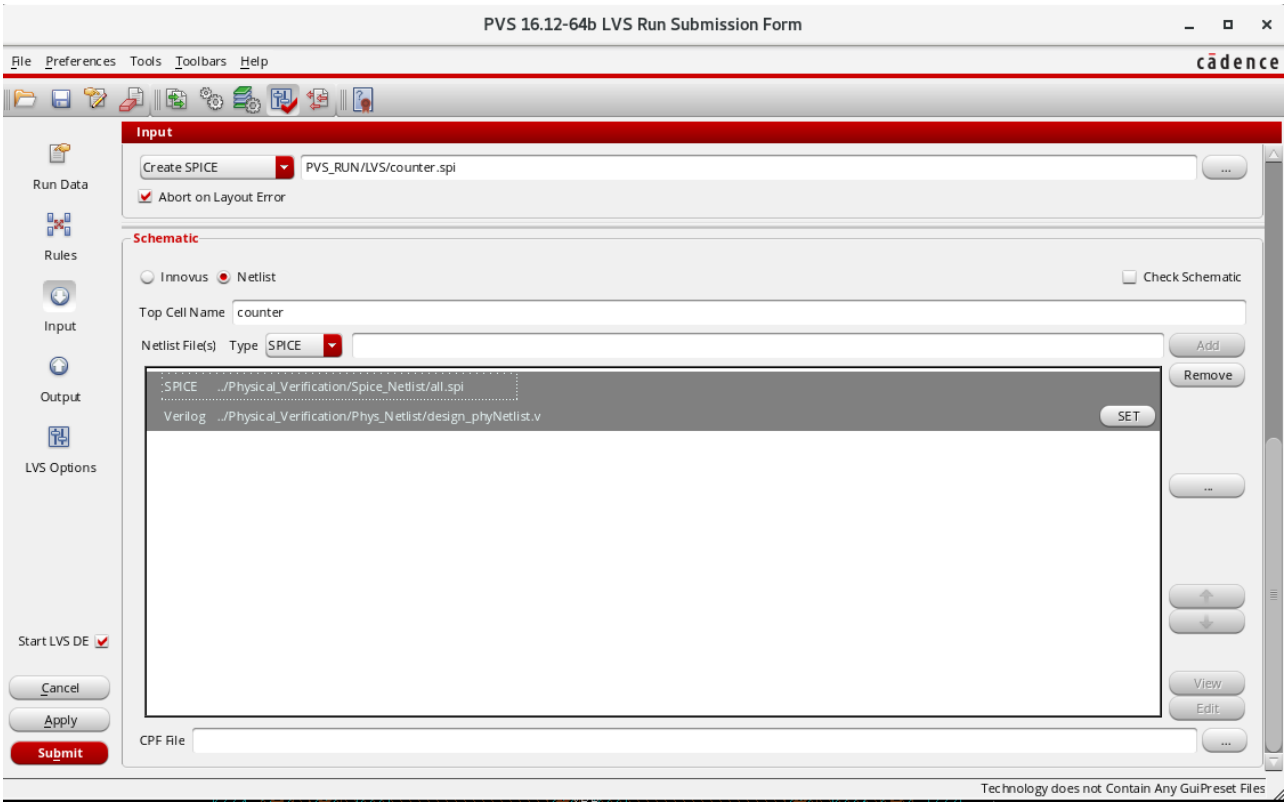


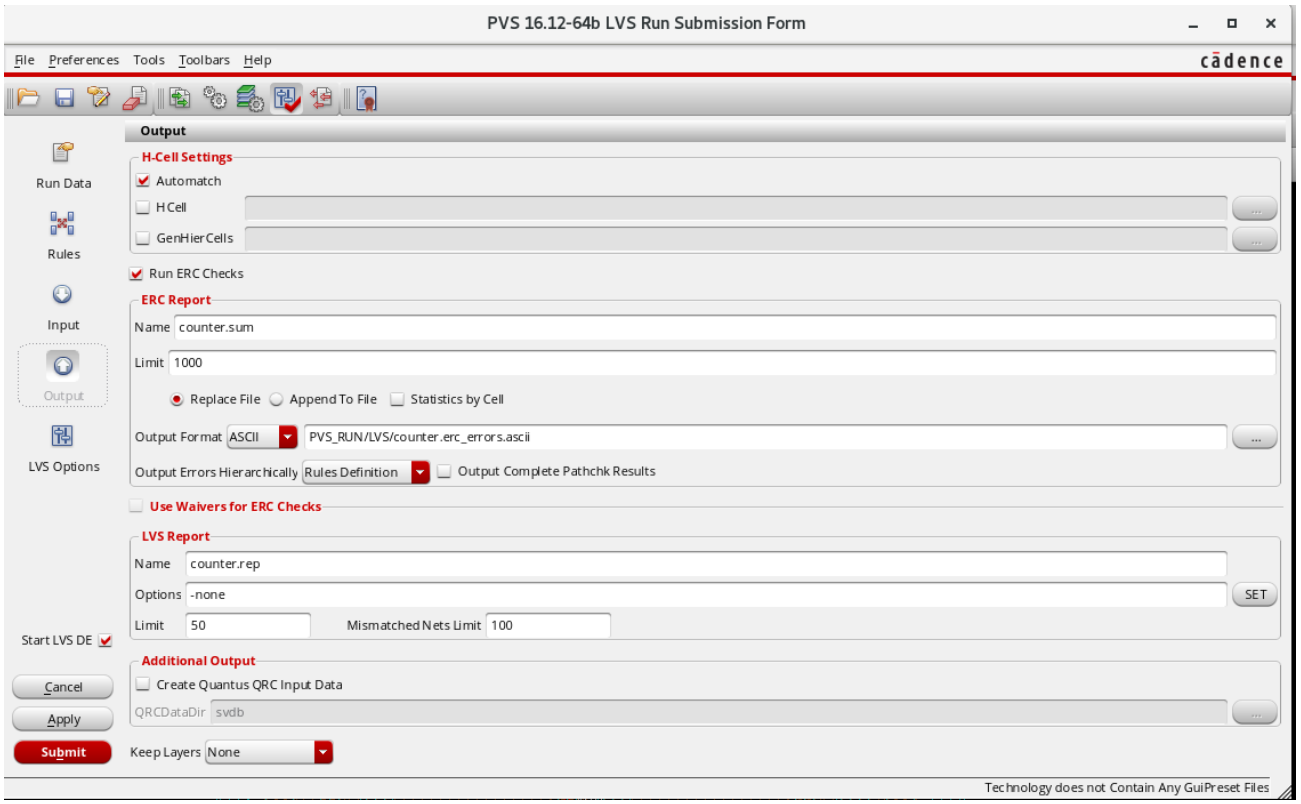
form.

- Provide the Run directory and log file name (Along with path – Optional)
- Load the Tech Lib, GDS Files and Spice Netlist of all Standard Cellsunder the corresponding technology node.

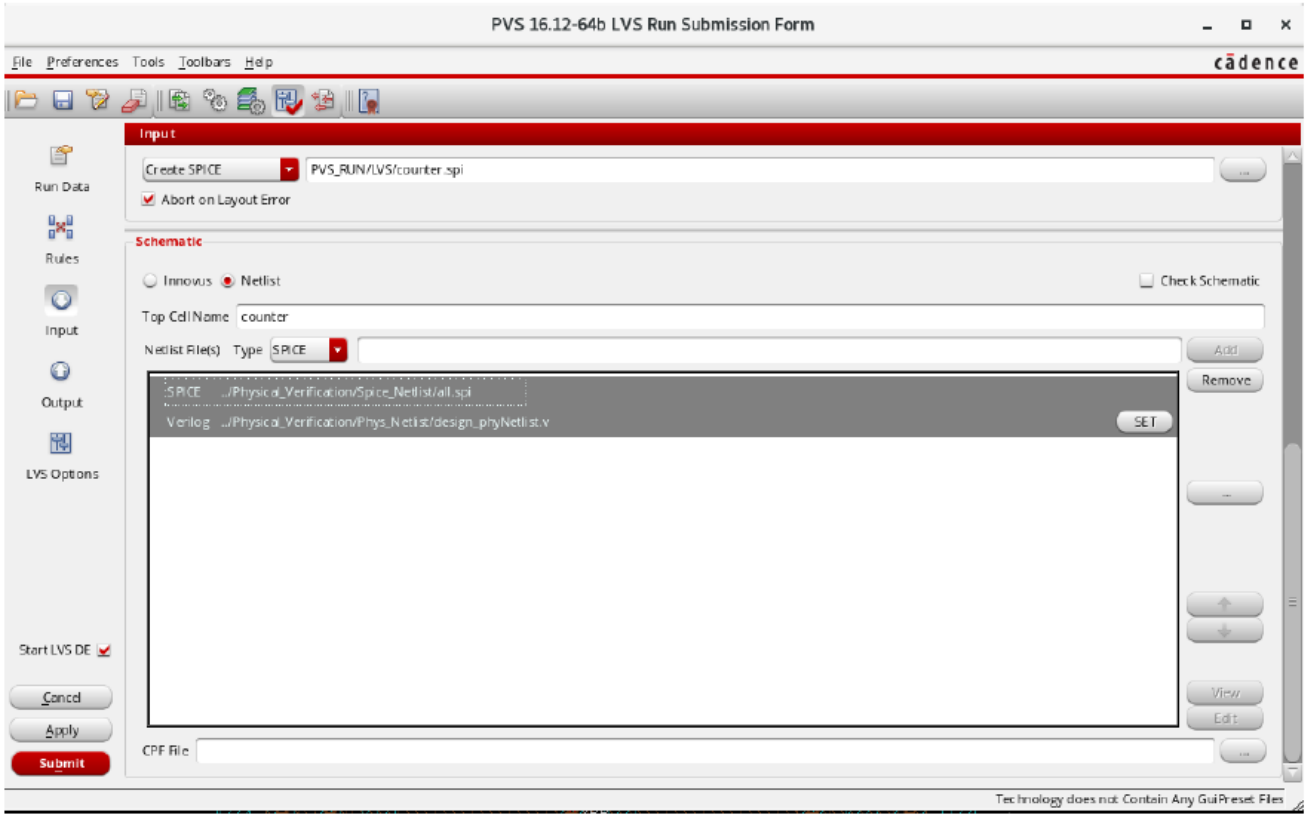

PVS 16.12-64b LVS Run Submission Form
File Preferences Tools Toolbars Help
cadence
Rules
Tech&Rules Configurator Include PVL
Run Data
Rules
Input
Output
LVS Options
Technology
Technology mapping file ../Physical_Verification/pvtech.lib
Reload
Technology gpdk090
Rule Set default
Rules
Rules File(s)
Add
Remove
../Physical_Verification/PVS_RULES/lvs.pvl Technology
View
Edit
Log File
Start LVS DE
Cancel
Apply
Submit
Technology does not Contain Any GuiPreset Files

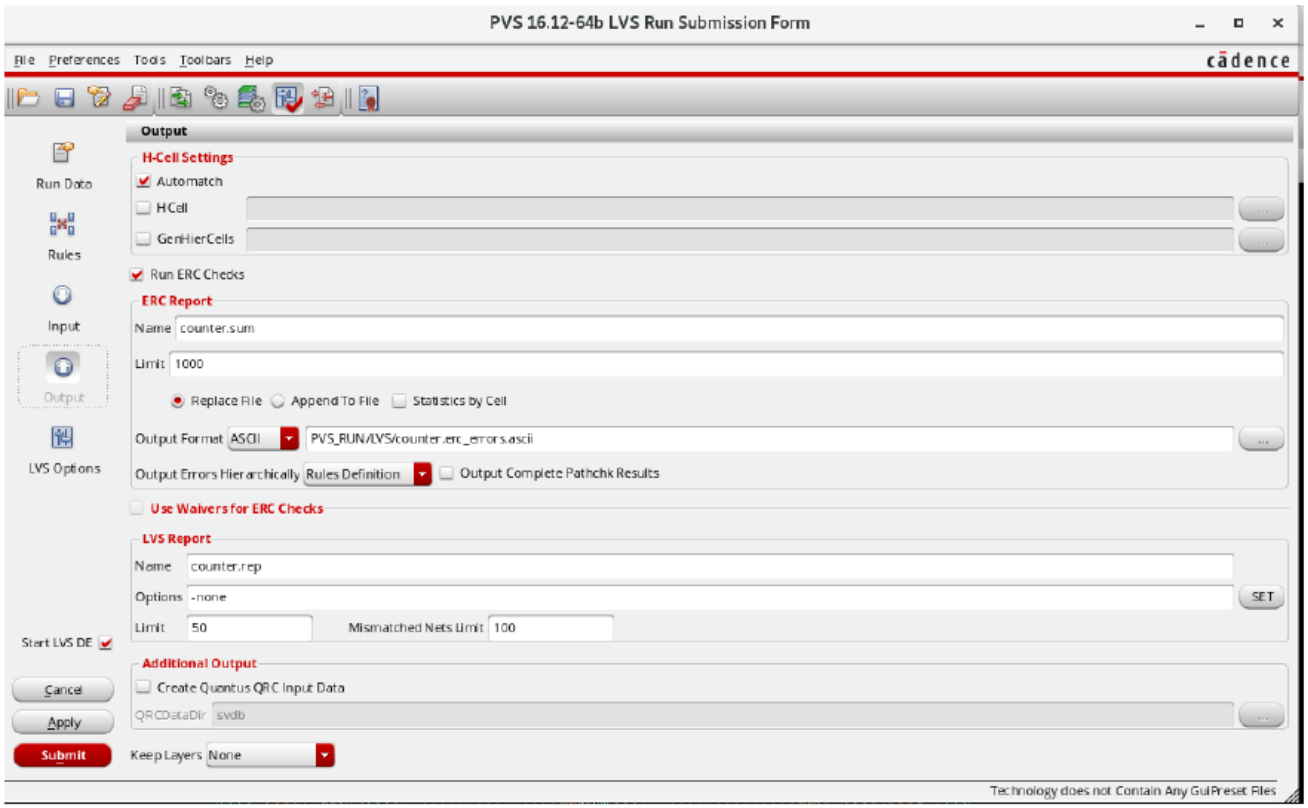

PVS 16.12-64b LVS Run Submission Form
File Preferences Tools Toolbars Help
cadence
Input
Layout
Run Data
Rules
Input
Output
LVS Options
Top Cell Name counter
Create GDSII
PVS_RUN/LVS/counter.gds.gz
Merge File(s)
Add
Remove
../Physical_Verification/gds/ACHCONX2.gds
../Physical_Verification/gds/ADDFHX1.gds
../Physical_Verification/gds/ADDFHX2.gds
../Physical_Verification/gds/ADDFHX4.gds
../Physical_Verification/gds/ADDFHXL.gds
View
Edit
Layout Options
Map File ../Physical_Verification/STREAM_MAP/90nmNew_streamOut.map
View
Uniquify Cell Names
With Prefix
DBU/UU 2000
Additional setStreamOutMode Parameters
Additional streamOut Parameters
Start LVS DE
Cancel
Apply
Submit
Create SPICE
PVS_RUN/LVS/counter.spi
Abort on Layout Error
Technology does not Contain Any GuiPreset Files


PVS 16.12-64b LVS Run Submission Form
File Preferences Tools Toolbars Help
cadence
Input
Run Data
Rules
Input
Output
LVS Options
Create SPICE
PVS_RUN/LVS/counter.spi
Abort on Layout Error
Schematic
Innovus
Netlist
Check Schematic
Top Cell Name counter
Netlist File(s)
Type SPICE
Add
Remove
SPICE ../Physical_Verification/Spice_Netlist/all.spi
Verilog ../Physical_Verification/Phys_Netlist/design_phyNetlist.v
SET
View
Edit
CPF File
Start LVS DE
Cancel
Apply
Submit
Technology does not Contain Any GuiPreset Files

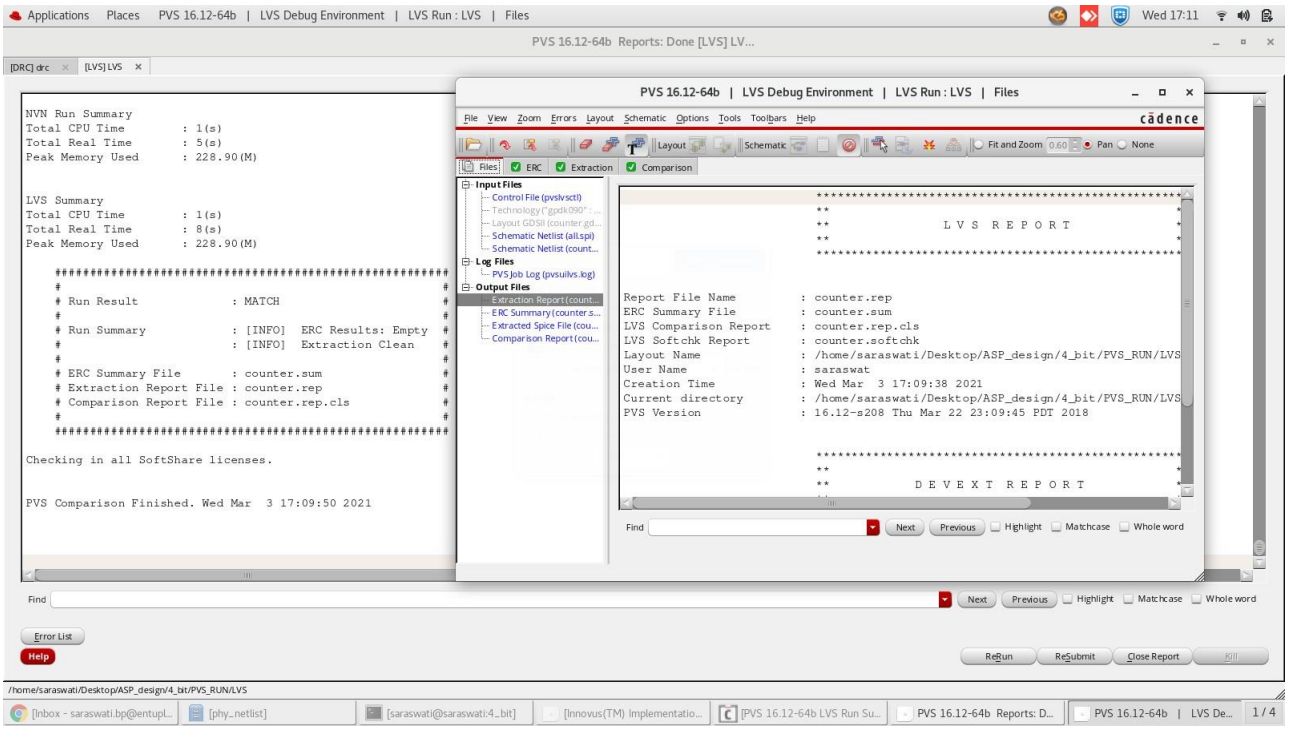


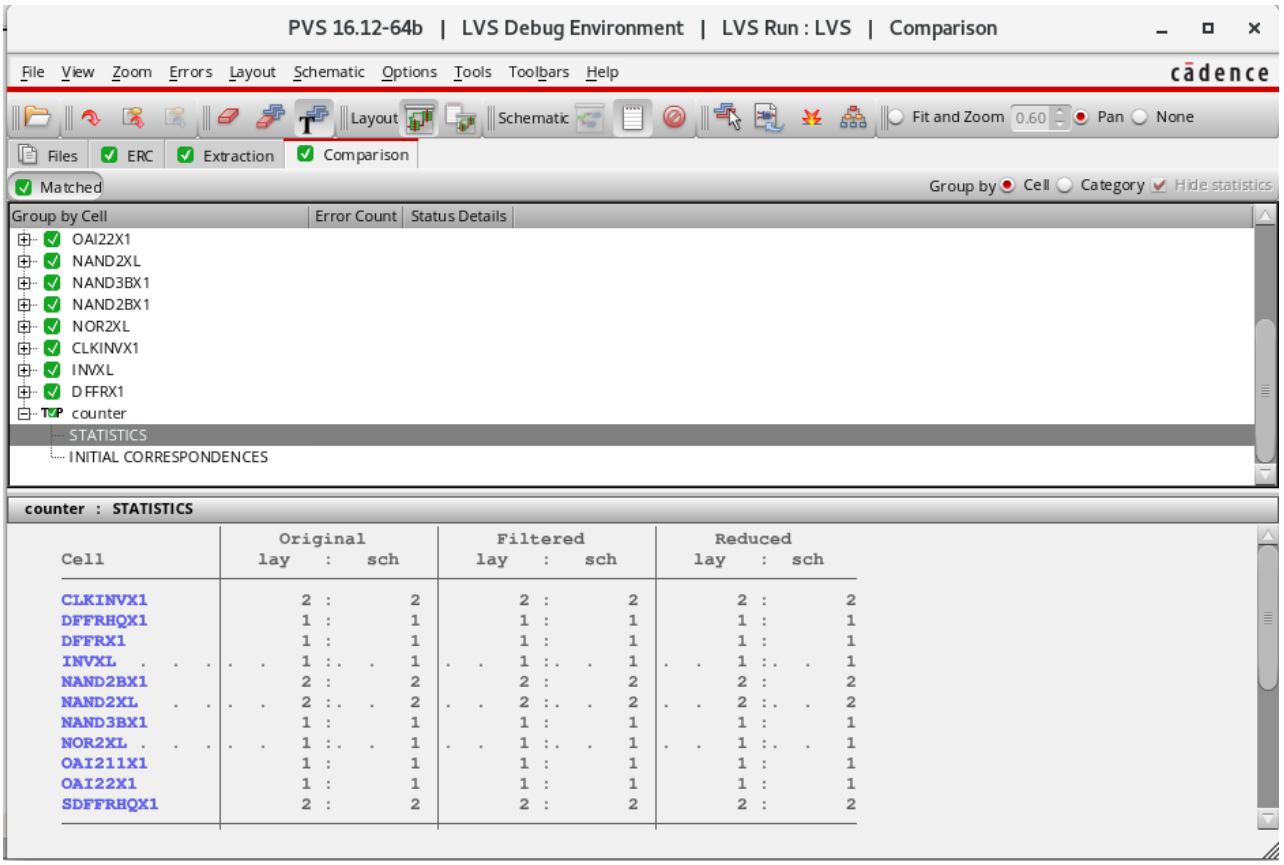
- On successful completion of LVS Run, the following windowsappear.

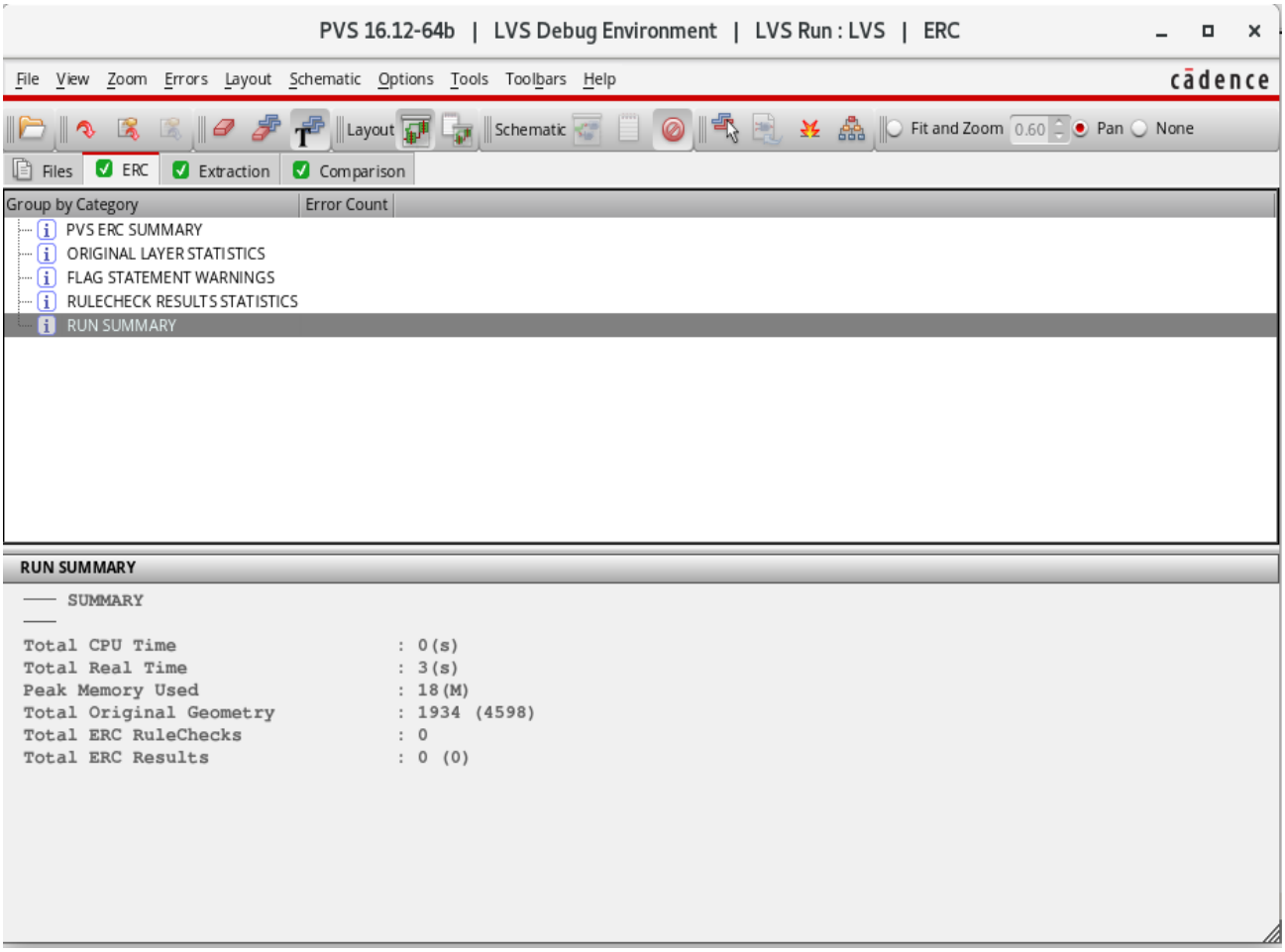




- On successful completion of LVS Run, the following windows appear.


PVS 16.12-64b | LVS Debug Environment | LVS Run : LVS | Files
NVN Run Summary
Total CPU Time : 1(s)
Total Real Time : 5(s)
Peak Memory Used : 228.90(M)
LVS Summary
Total CPU Time : 1(s)
Total Real Time : 8(s)
Peak Memory Used : 228.90(M)
Run Result : MATCH
Run Summary : [INFO] ERC Results: Empty
: [INFO] Extraction Clean
ERC Summary File : counter.sum
Extraction Report File : counter.rep
Comparison Report File : counter.rep.cls
Checking in all SoftShare licenses.
PVS Comparison Finished. Wed Mar 3 17:09:50 2021
L V S R E P O R T
Report File Name : counter.rep
ERC Summary File : counter.sum
LVS Comparison Report : counter.rep.cls
LVS Softchk Report : counter.softchk
Layout Name : /home/saraswati/Desktop/ASP_design/4_bit/PVS_RUN/LVS
User Name : saraswat
Creation Time : Wed Mar 3 17:09:38 2021
Current directory : /home/saraswati/Desktop/ASP_design/4_bit/PVS_RUN/LVS
PVS Version : 16.12-s208 Thu Mar 22 23:09:45 PDT 2018
D E V E X T R E P O R T


PVS 16.12-64b | LVS Debug Environment | LVS Run : LVS | Comparison
Files ERC Extraction Comparison
Matched
Group by Cell Category Hide statistics
Group by Cell
Error Count
Status Details
OAI22X1
NAND2XL
NAND3BX1
NAND2BX1
NOR2XL
CLKINVX1
INVXL
DFFRX1
counter
STATISTICS
INITIAL CORRESPONDENCES
counter : STATISTICS
Cell
Original lay : sch
Filtered lay : sch
Reduced lay : sch
CLKINVX1 2 : 2 2 : 2 2 : 2
DFFRHQX1 1 : 1 1 : 1 1 : 1
DFFRX1 1 : 1 1 : 1 1 : 1
INVXL 1 : 1 1 : 1 1 : 1
NAND2BX1 2 : 2 2 : 2 2 : 2
NAND2XL 2 : 2 2 : 2 2 : 2
NAND3BX1 1 : 1 1 : 1 1 : 1
NOR2XL 1 : 1 1 : 1 1 : 1
OAI211X1 1 : 1 1 : 1 1 : 1
OAI22X1 1 : 1 1 : 1 1 : 1
SDFFRHQX1 2 : 2 2 : 2 2 : 2



- You can create a GDS file along with Stream out file either using theGUI as File → Save → GDS/Oasis or use the following command.
- **Cmd :** streamOut <GDSFileName>.gds -streamOut <streamOut>.map