|| **Jai Sri Gurudev** ||

Sri Adichunchanagiri Shikshana Trust[R]

# S.J.C. INSTITUTE OF TECHNOLOGY

## CHICKBALLAPUR – 562101

## DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

# VLSI LAB MANUAL
# (18ECL77)
# VII SEMESTER

**PREPARED BY**

**Dr. B.N. Shobha,** Professor & Head

**Veena S,** Assistant Professor

# S.J.C. INSTITUTE OF TECHNOLOGY

## VISION OF THE INSTITUTE

S.J.C.I.T is committed to Quality Education, Training and Research.

## MISSION OF THE INSTITUTE

- Augmenting the supply of competent Engineers and Managers
- Building Engineers and Managers with Value, Vision and Versatility
- Developing and dissemination of new Knowledge and Insights

## DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

## VISION OF DEPARTMENT

To Achieve Academic Excellence in Electronics and Communication Engineering by Imparting quality Technical Education and facilitating Research Activities

## MISSION OF DEPARTMENT

- Establishing State of the Art Laboratory facilities and Infrastructure to develop the spirit of Innovation and Entrepreneurship
- Nurturing the Students with Technical Expertise along with Professional Ethics to provide solutions for Societal Needs
- Encourage Life Long Learning And Research among the Students and Faculty

## PROGRAM EDUCATIONAL OBJECTIVES

After successful completion of the program, the Graduates of the program will :

**PEO1:** Have successful technical and professional career in Engineering, Technology and multidisciplinary environments.

**PEO2:** Utilize their knowledge, technical and communication Skills to propose optimal solutions to problems related to society in the field of Electronics and Communication.

**PEO3:** Exhibit good interpersonal skills, leadership qualities and adapt themselves for lifelong learning.

## PROGRAM SPECIFIC OUTCOMES

**PSO1: Professional Skills:** Ability to absorb and apply fundamental knowledge of core Electronics and Communication Engineering in the analysis, design and development of Electronics Systems as well as to interpret and synthesize experimental data leading to valid conclusions

**PSO2: Problem-solving skills:** Ability to solve complex Electronics and Communication Engineering problems, using latest hardware and software tools, along with analytical and managerial skills to arrive at appropriate solutions, either independently or in team

## TABLE OF CONTENTS

# Course syllabus

## PART – A

### Analog Design

**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.

**3. 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

**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:

| | |
|---|---|
| **C407.1** | Understand the Synthesis process of digital circuits using EDA tool. |
| **C407.2** | Design and simulate combinational and sequential digital circuits using Verilog HDL |
| **C407.3** | Perform ASIC design flow and understand the process of synthesis, synthesis constraints and evaluating the synthesis reports to obtain optimum gate level net list |
| **C407.4** | Design and simulate basic CMOS circuits like inverter, common source amplifier and differential amplifiers. |
| **C407.5** | Perform RTL-GDSII flow and understand the stages in ASIC design. |

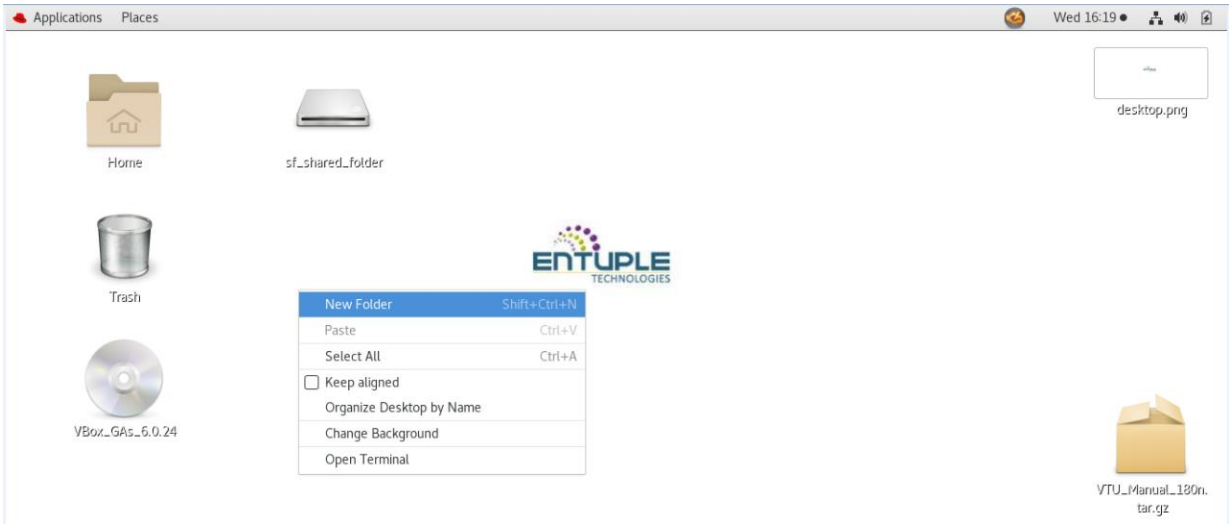
| CO/POs | PO1 | PO2 | PO3 | PO4 | PO5 | PO6 | PO7 | PO8 | PO9 | PO10 | PO11 | PO12 | PSO1 | PSO2 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **C407.1** | 3 | 3 | 3 | - | - | - | - | - | 2 | 1 | - | 1 | 2 | 3 |
| **C407.2** | 3 | 3 | 3 | 3 | 3 | - | - | - | 2 | 1 | - | - | 3 | 2 |
| **C407.3** | 3 | 3 | 3 | 2 | 3 | - | - | - | 2 | 1 | - | - | 3 | 3 |
| **C407.4** | 3 | 3 | 3 | 2 | 3 | - | - | - | 1 | 1 | - | - | 3 | 3 |
| **C407.5** | 3 | 3 | - | - | 3 | - | - | - | - | - | - | - | 3 | 3 |

# GENERAL NOTES

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.

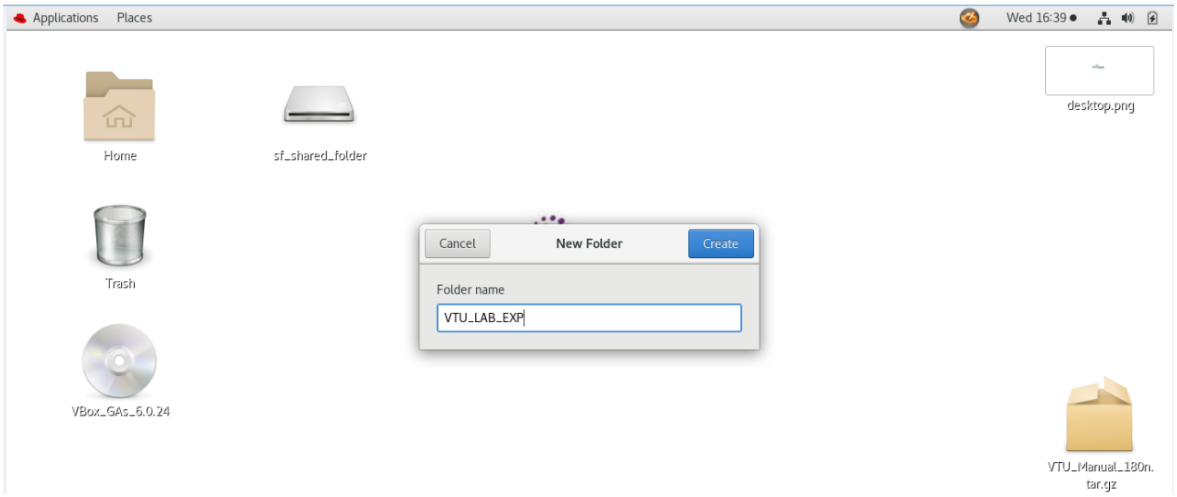



***Figure – 2: Name the Folder***

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

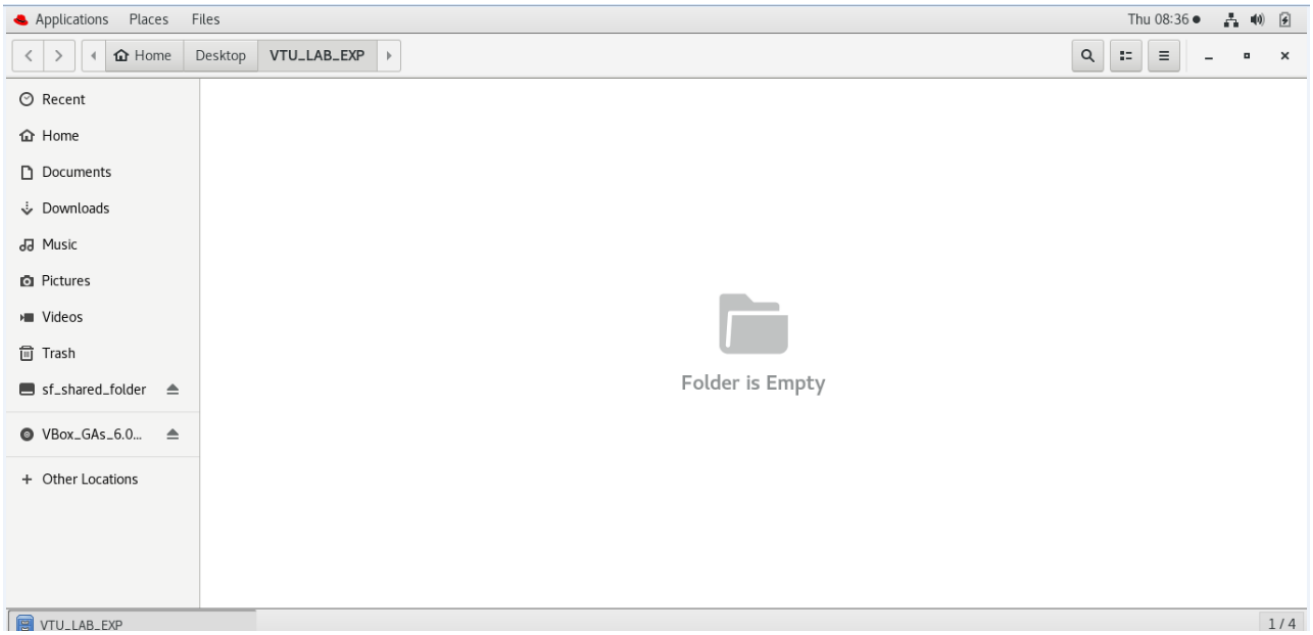



*Figure – 3: Open the folder*

**INITIALISING csh& SOURCING cshrc:**

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

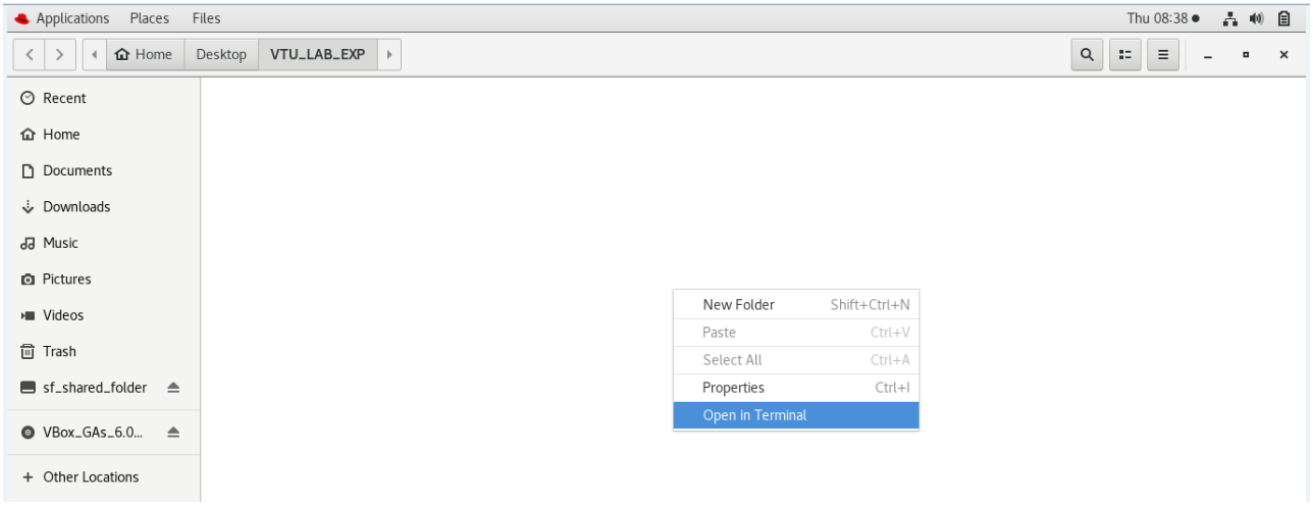



*Figure – 4: Open in Terminal*

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.

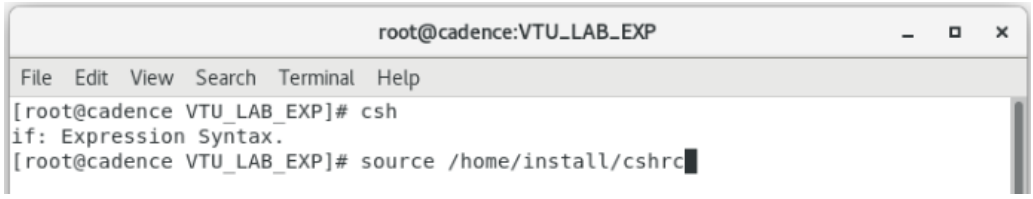



*Figure – 5: "csh" and "source /home/install/cshrc" commands*

**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.

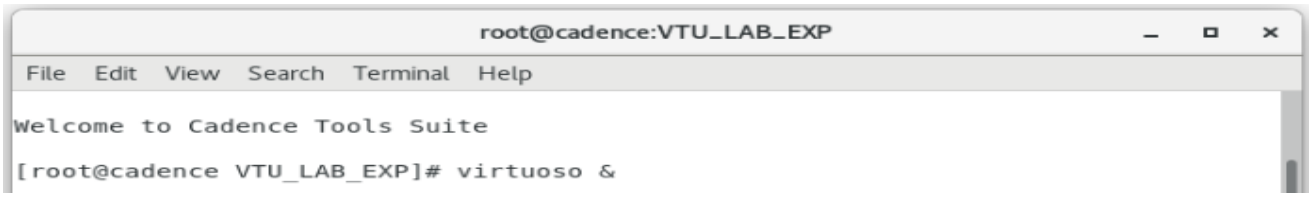



*Figure - 7: Command to invoke "virtuoso"*

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

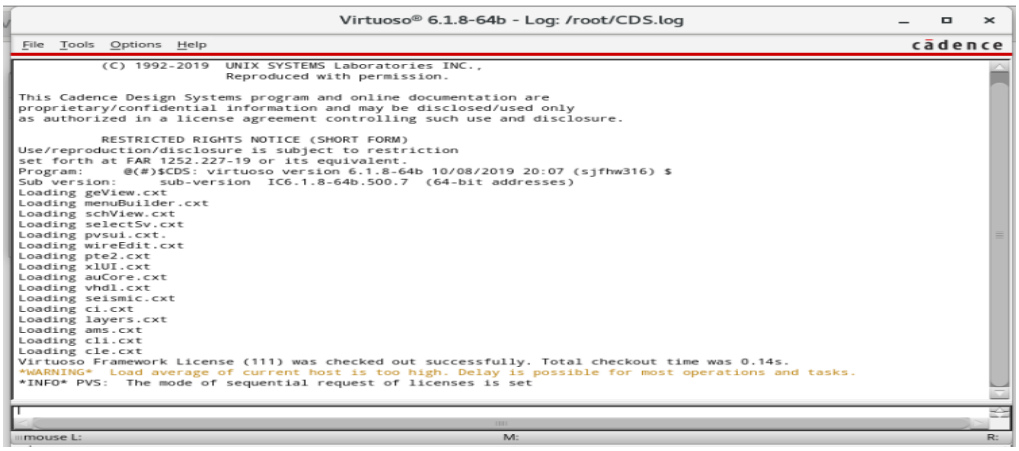



*Figure – 8: Command Interpreter Window (CIW)*

## 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) $W_N = 2W_P$
(iii) $W_N = W_P / 2$
and Length at selected Technology. Carry out the following:

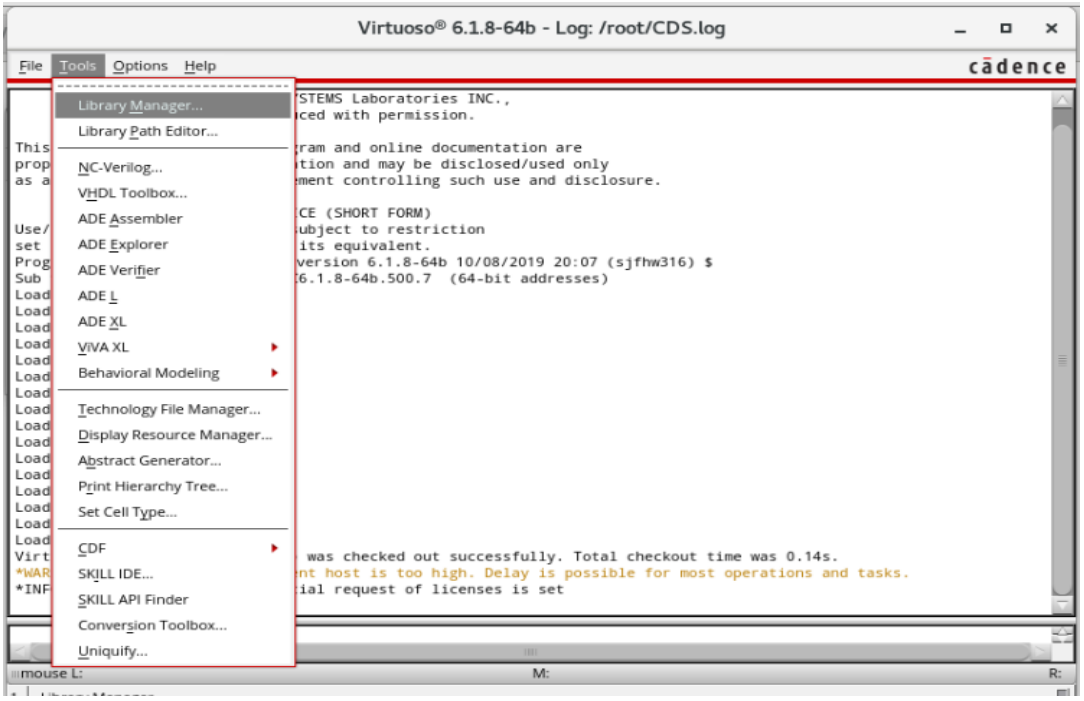
1. Set the Input Signal to a pulse with Rise Time, Fall Time of 1 ps and Pulse Width of 10 ns, Time Period of 20 ns and plot the input voltage and output voltage of the designed Inverter
2. From the Simulation Results, compute $tp_{HL}$, $tp_{LH}$and $t_{PD}$ for all the three geometrical settings of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

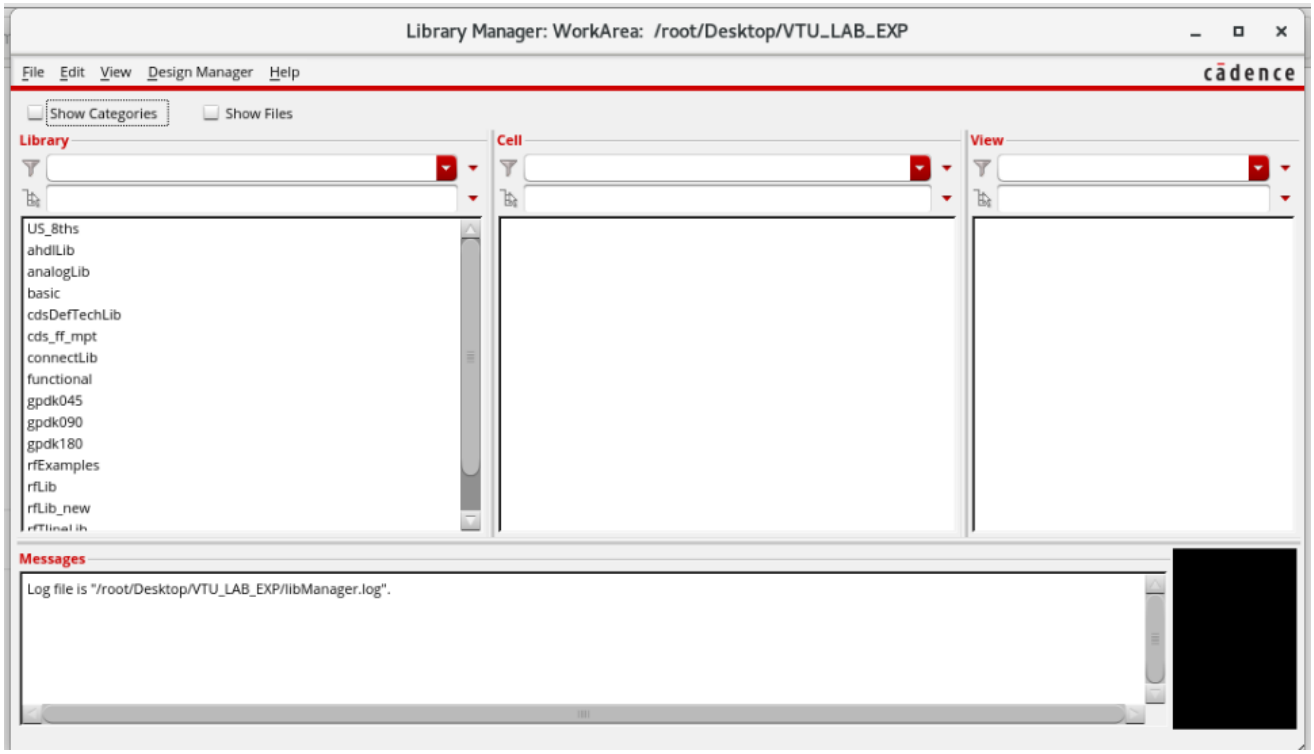
## CREATE A LIBRARY:

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.




***Figure–1.2: Library Manager***

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

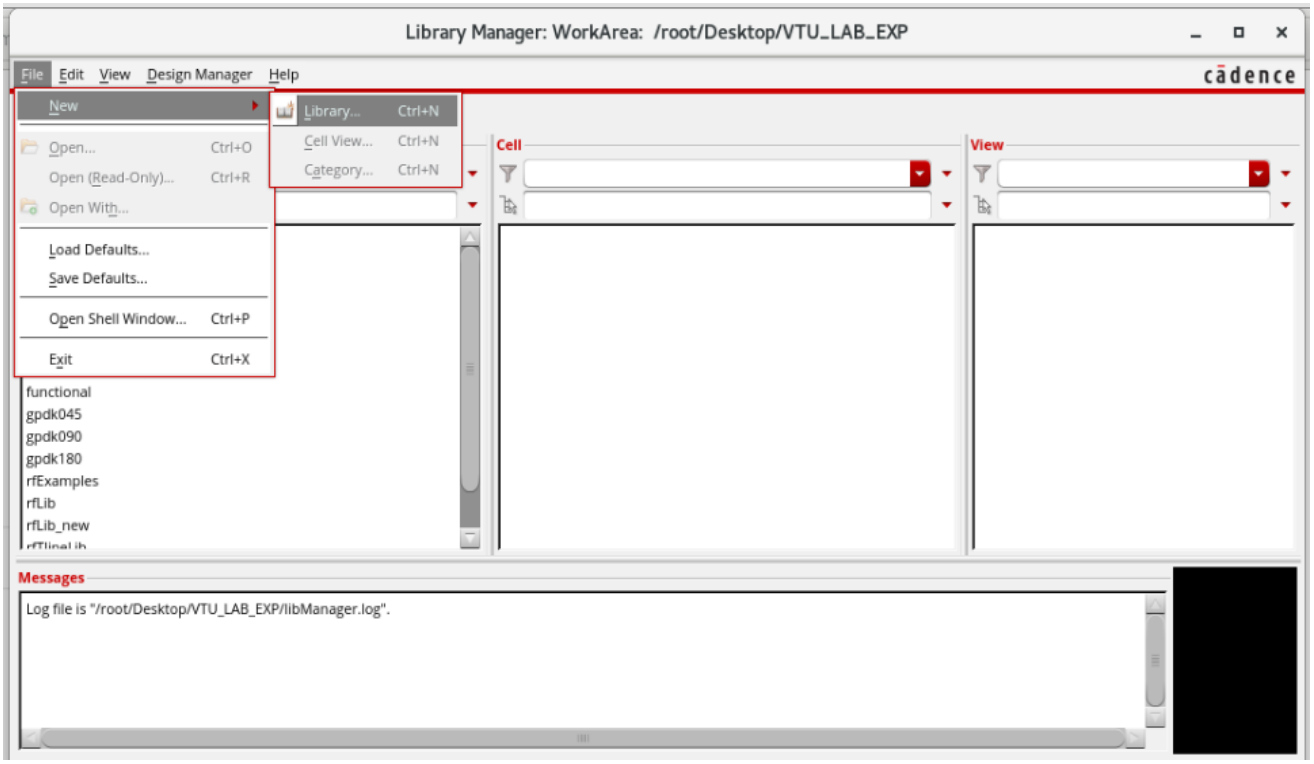



***Figure – 1.3: File → New → Library***

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**".

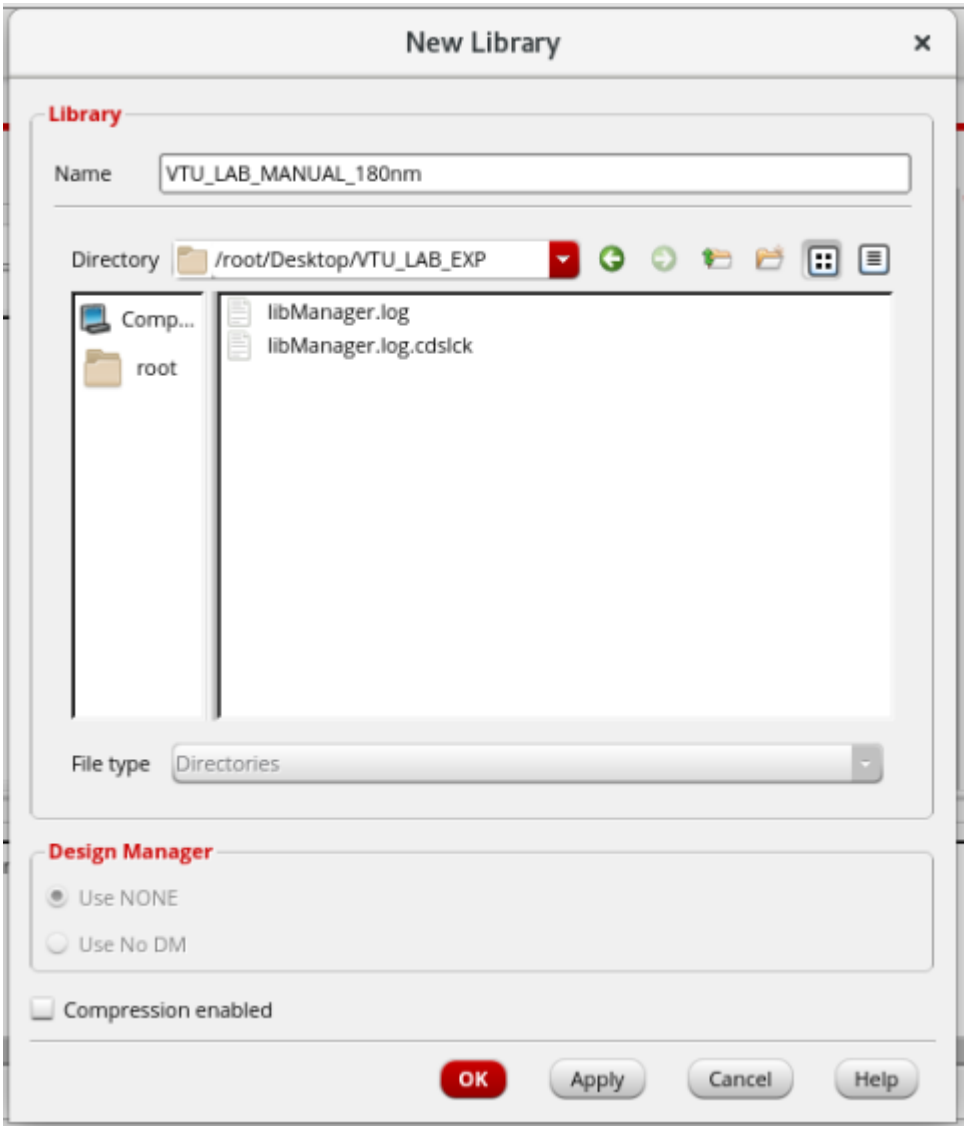



***Figure – 1.4: Name the Library***

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.

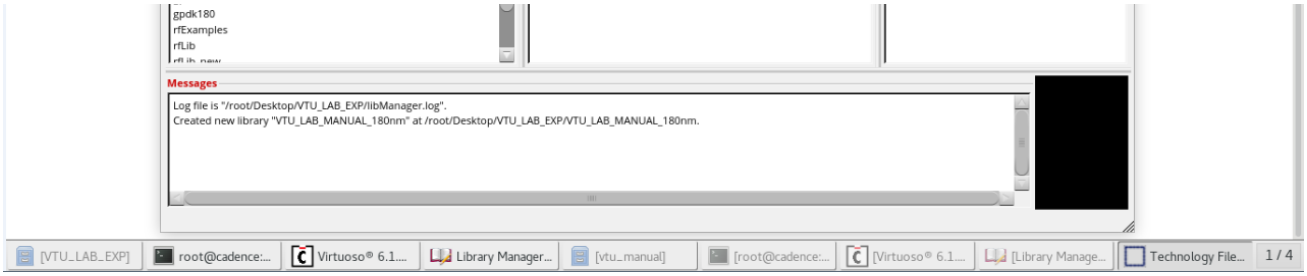
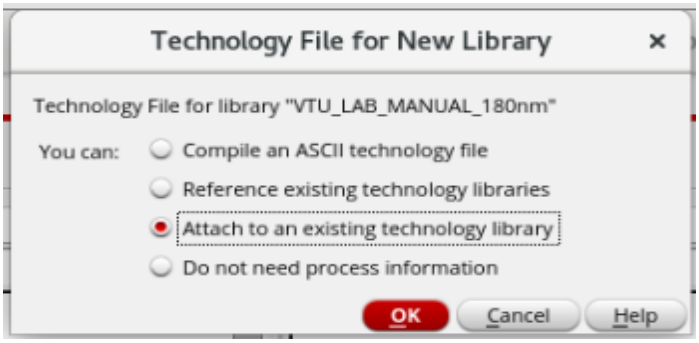



***Figure – 1.5: "Technology File.." Tab***

Click on the tab and "**Technology File for New Library**"window can be seen as in Figure–1.6. 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**".

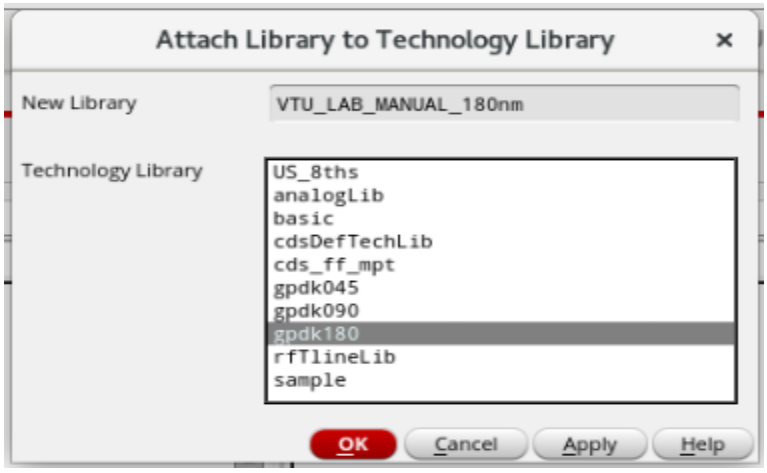



***Figure – 1.7: Technology Node Selection***

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

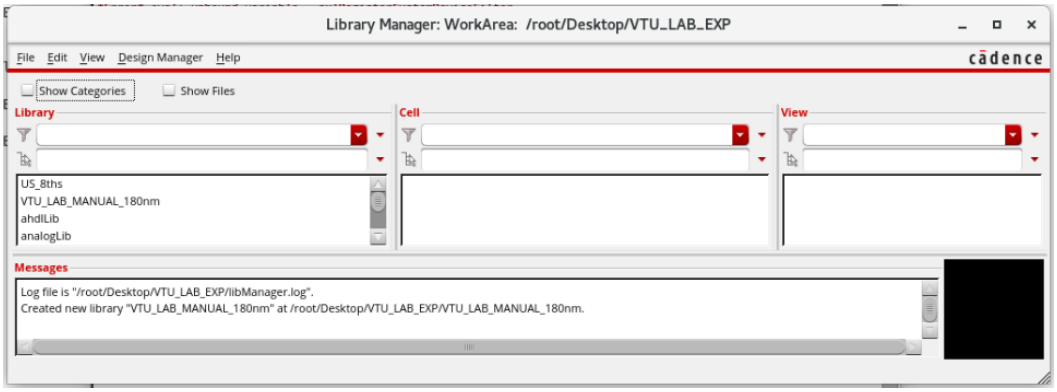



***Figure – 1.8: New Library included to Library Manager***

## CREATE A CELLVIEW:

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

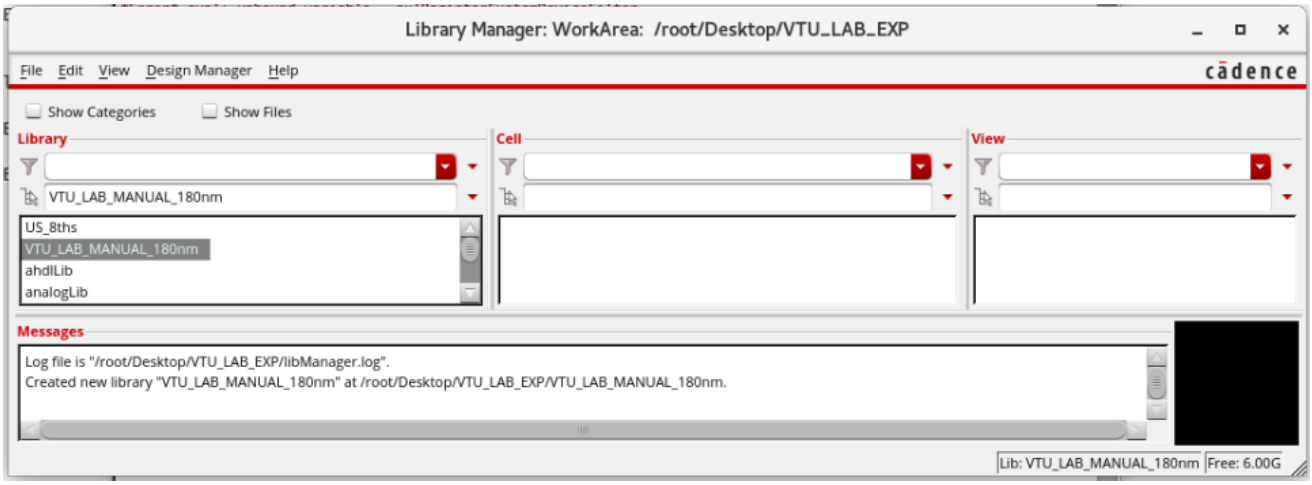



***Figure – 1.9: Select the Library***

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

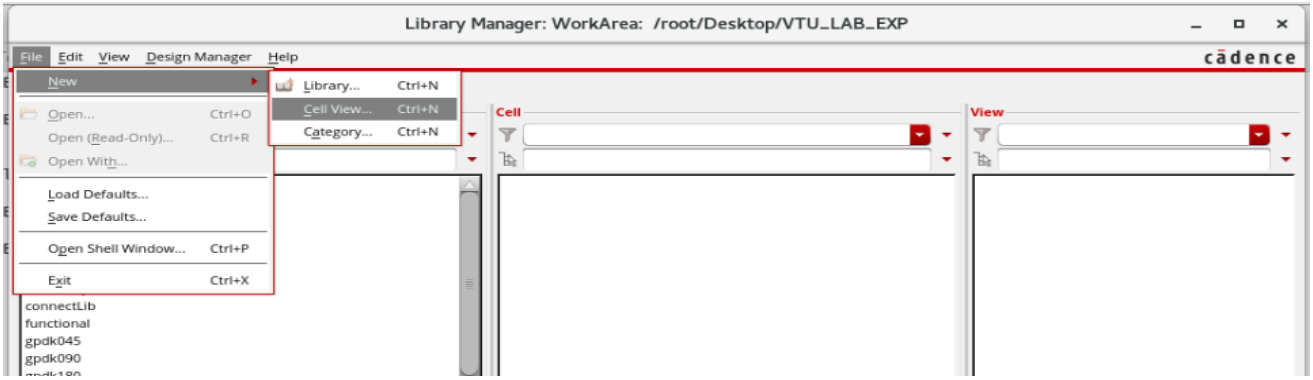



***Figure – 1.10: File → New → Cell View***

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

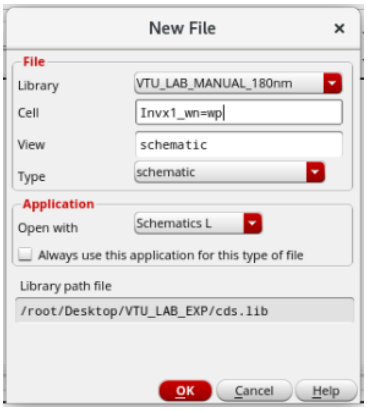



***Figure – 1.11: "New File" Window***

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

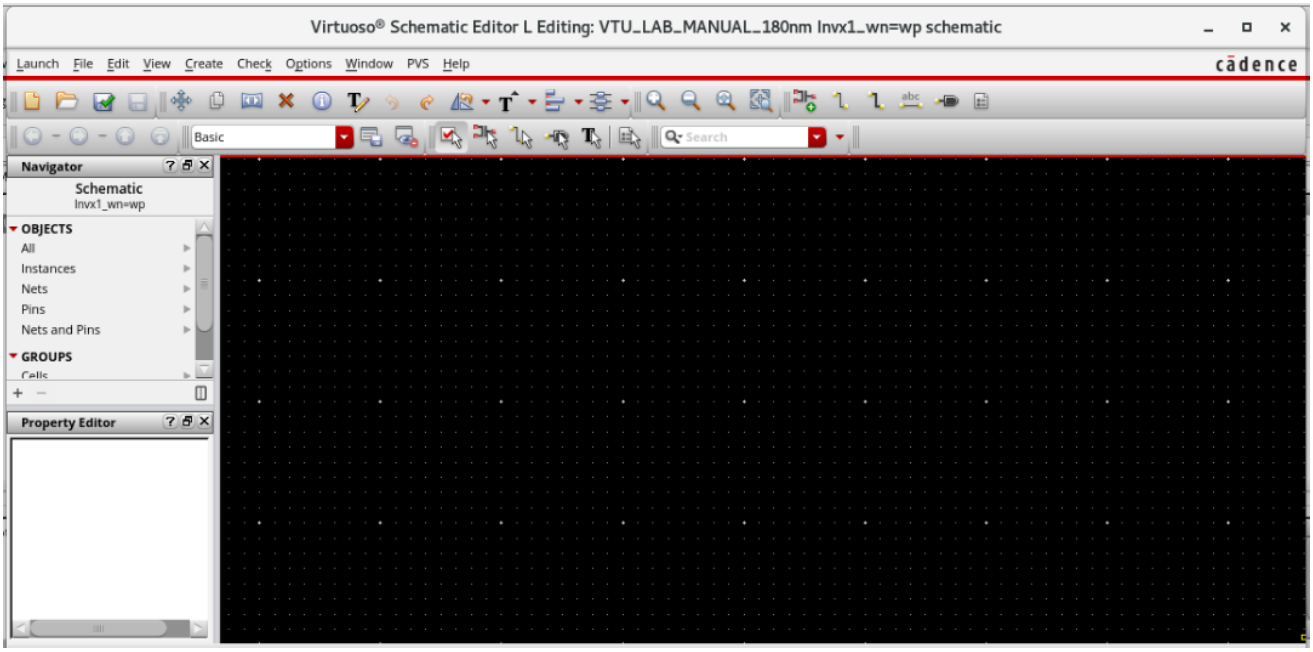



***Figure – 1.12: Virtuoso Schematic Editor***

### (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 Figure1.13

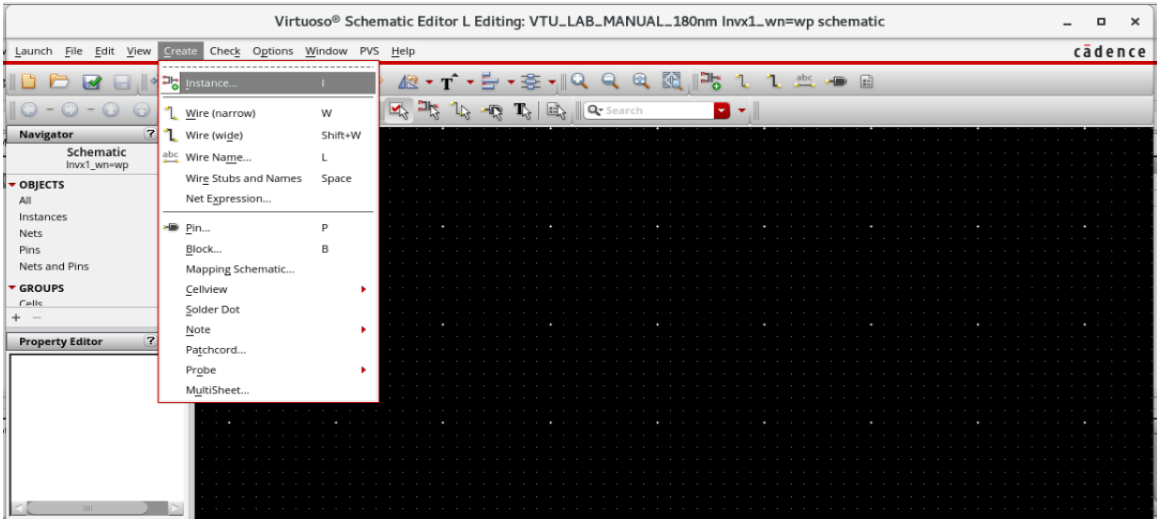



***Figure – 1.13: Create → Instance***

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

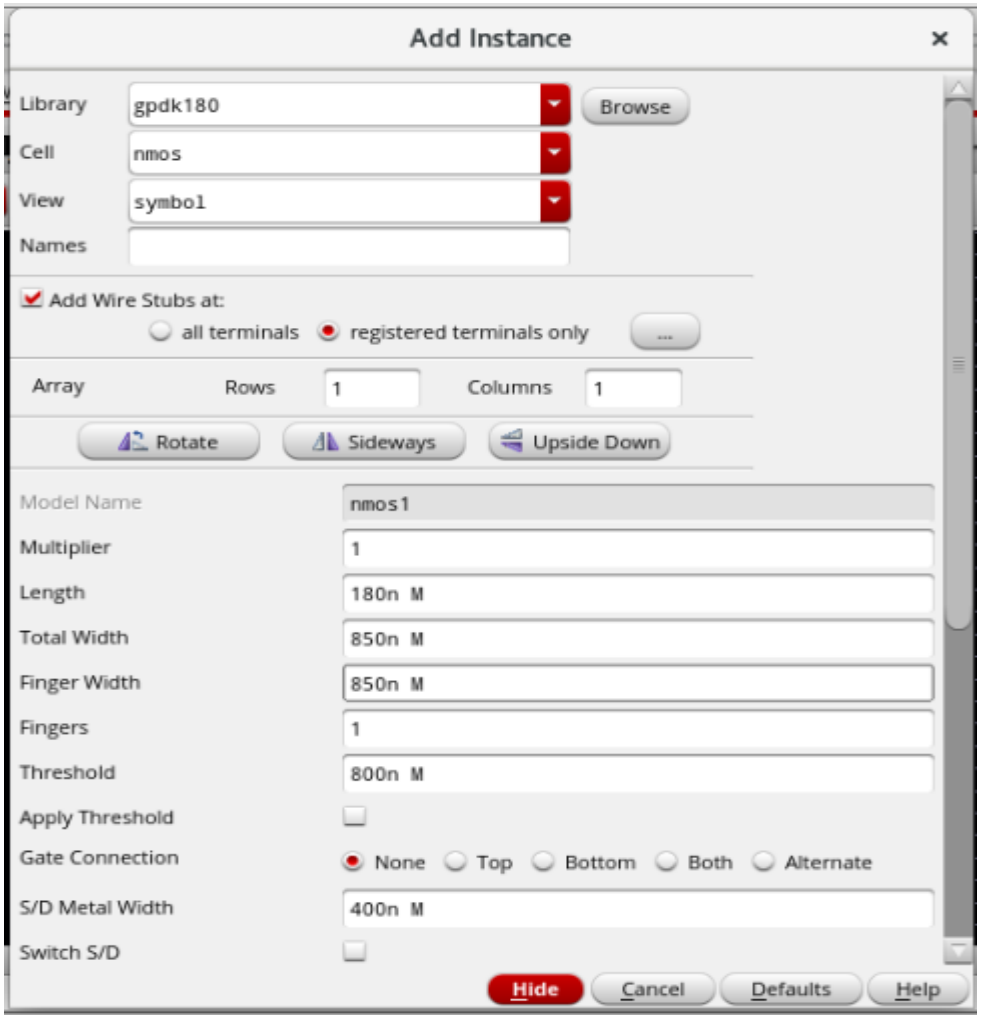



***Figure – 1.14: "Add Instance" Window***

Click onthe 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 devicefrom the list. For the CMOS Inverter circuit, PMOS and NMOS transistorsare 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<br>Length, L = 180 n |
| gpdk180 | Pmos | Width, $W_P$ = 850 n<br>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<br>Length, L = 180 n |
| gpdk180 | Pmos | Width, $W_P$ = 1.7 u<br>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<br>Length, L = 180 n |
| gpdk180 | Pmos | Width, $W_P$ = 425 n<br>Length, L = 180 n |

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

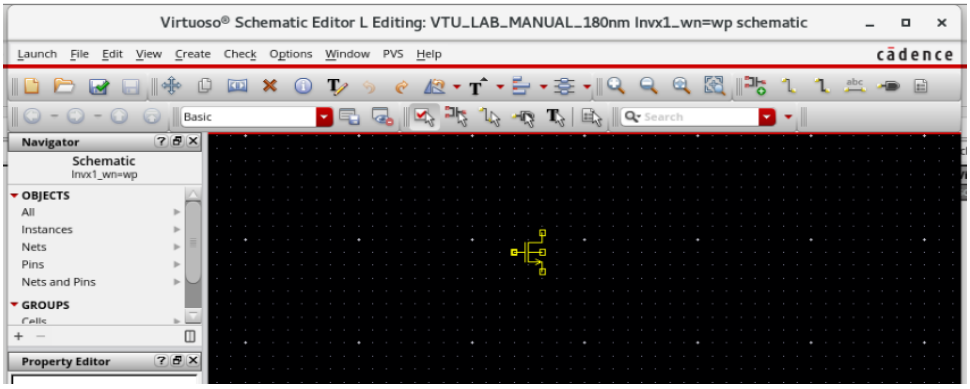



***Figure – 1.15: Instance after Selection***

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.

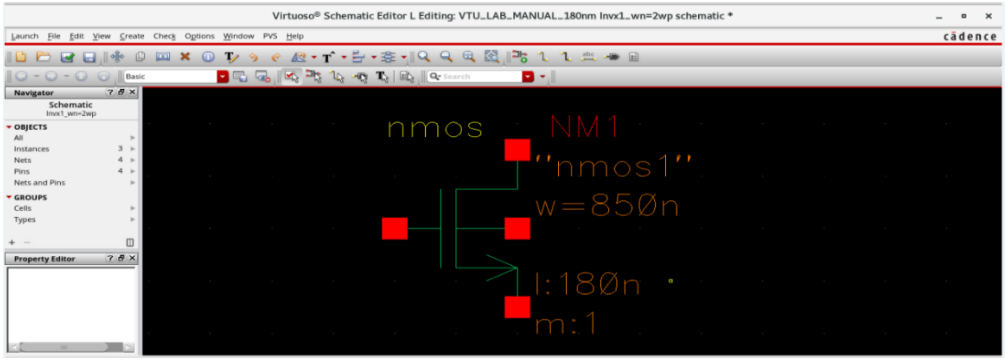



***Figure – 1.16: Instance after left mouse click***

**ADD PIN:**

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

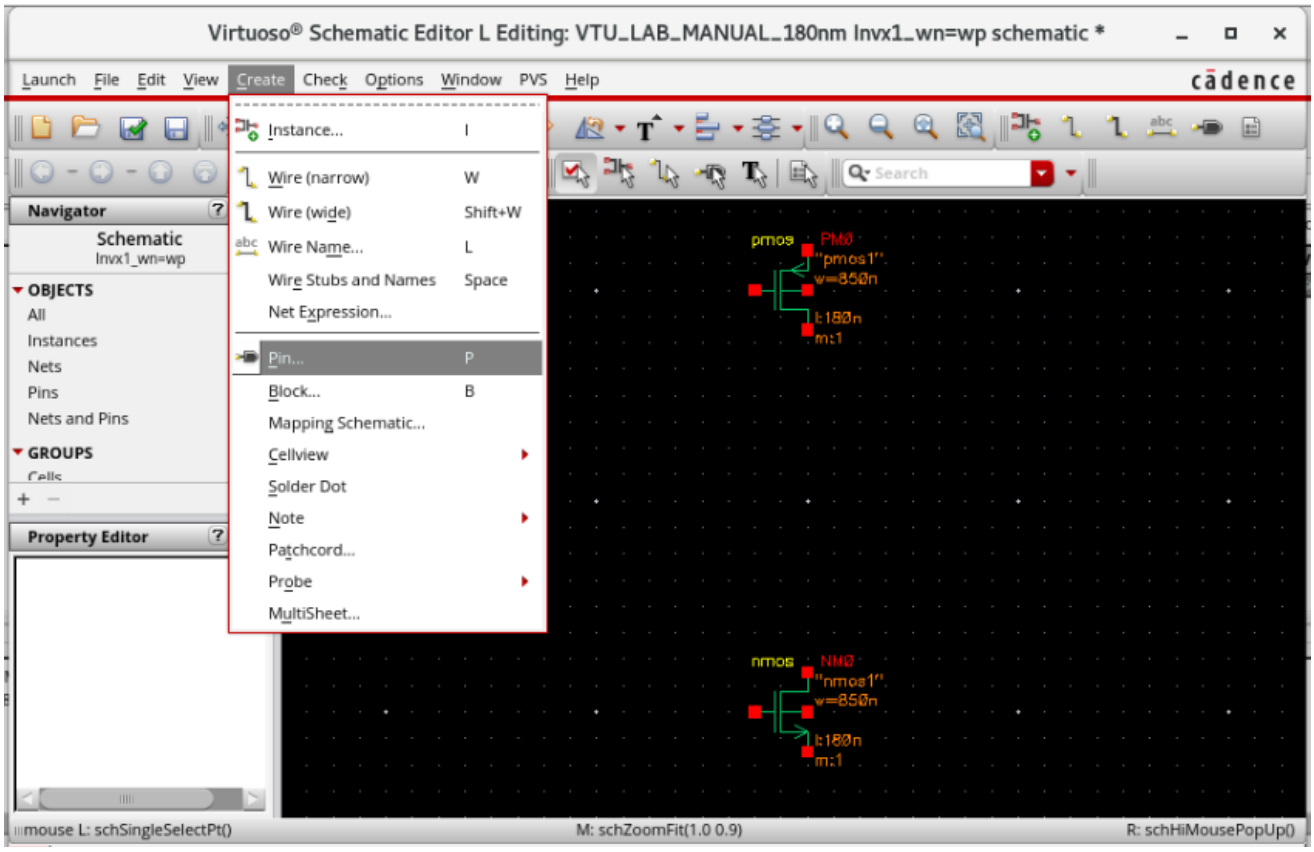
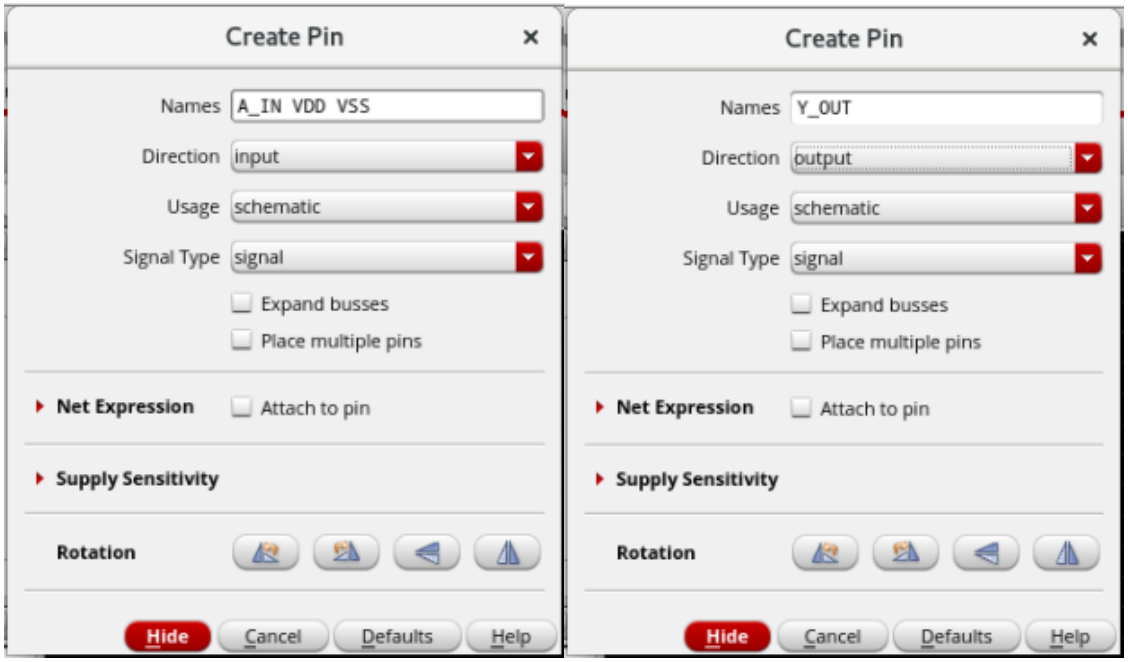



***Figure – 1.17: Create → Pin***

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–1.19(a): Naming the Input Pins*** ***Figure–1.19(b): Naming the Output Pins***

The pins are visualized on the Schematic Editor as shown in Figure – 1.20.

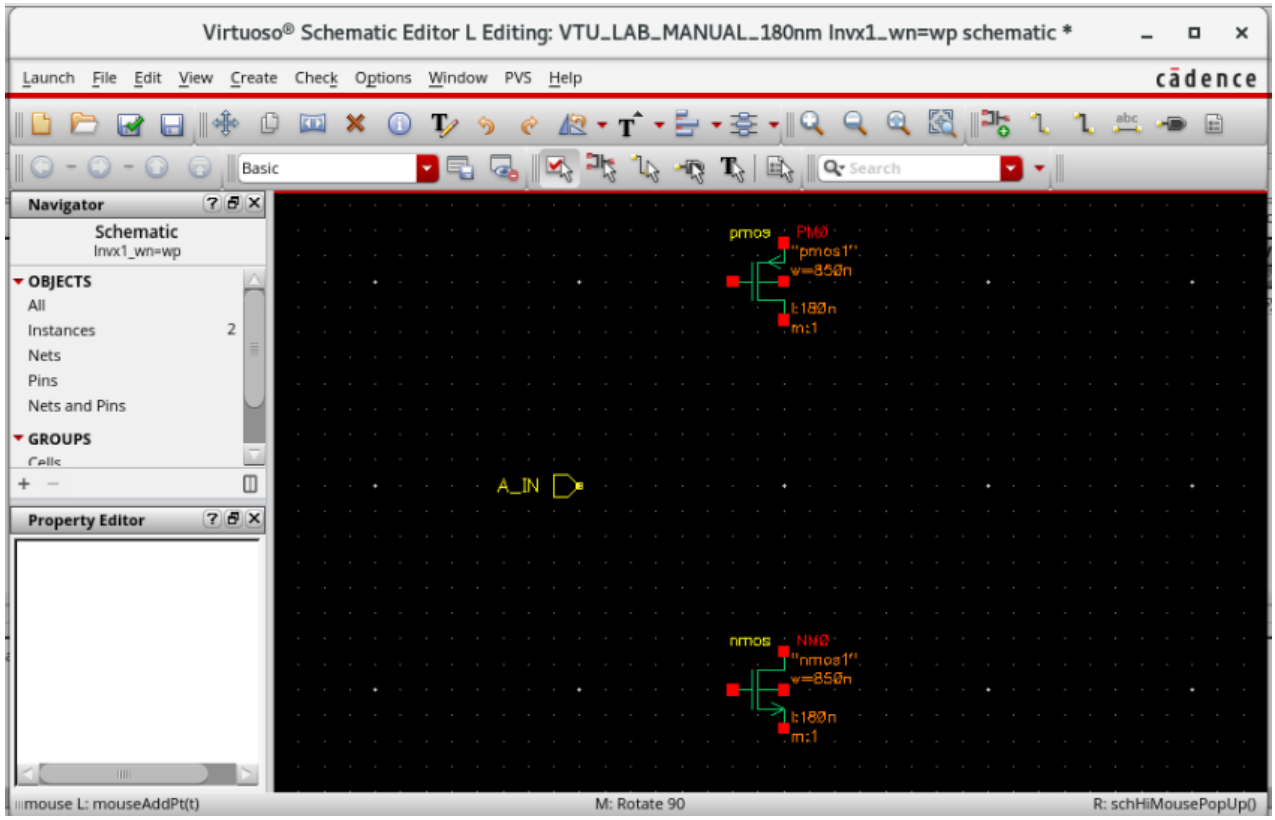



***Figure – 1.20: Pins after left mouse click on "Hide"***

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.

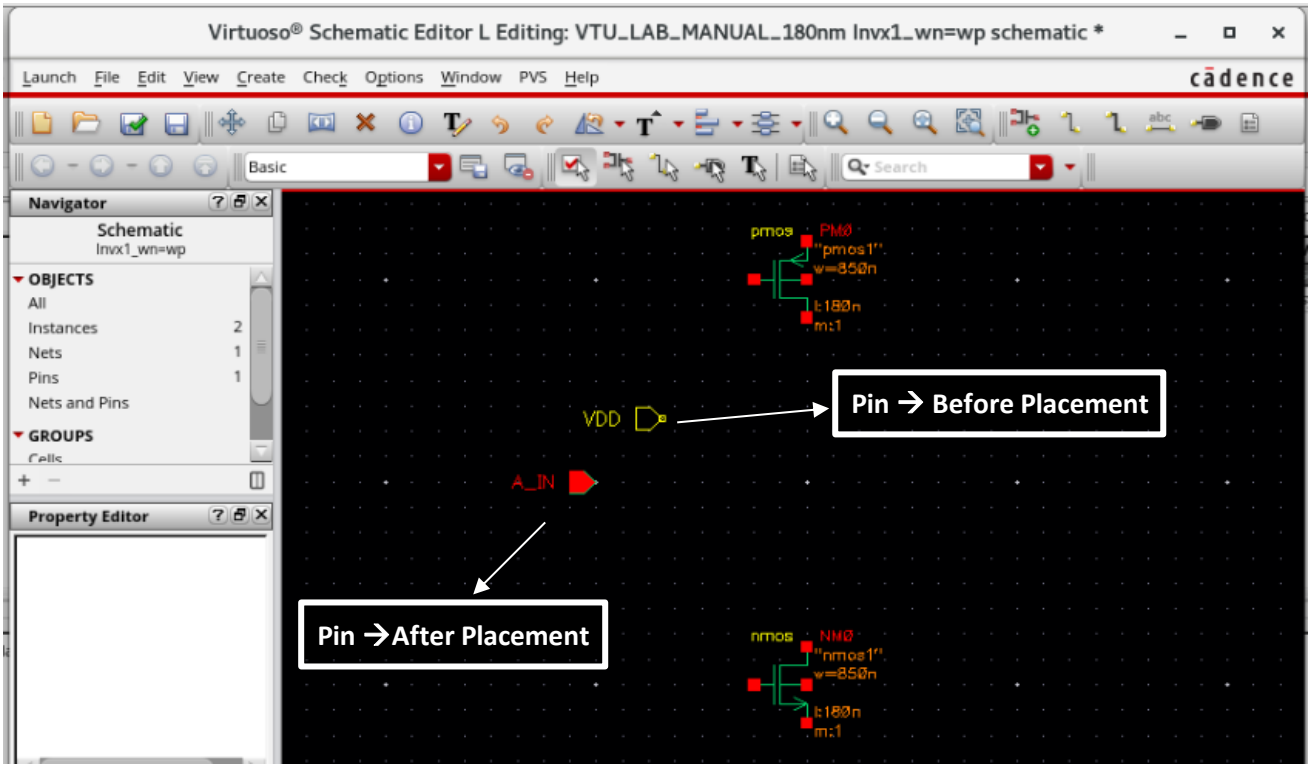



*Figure – 1.21: Pins Before and After Placement*

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.

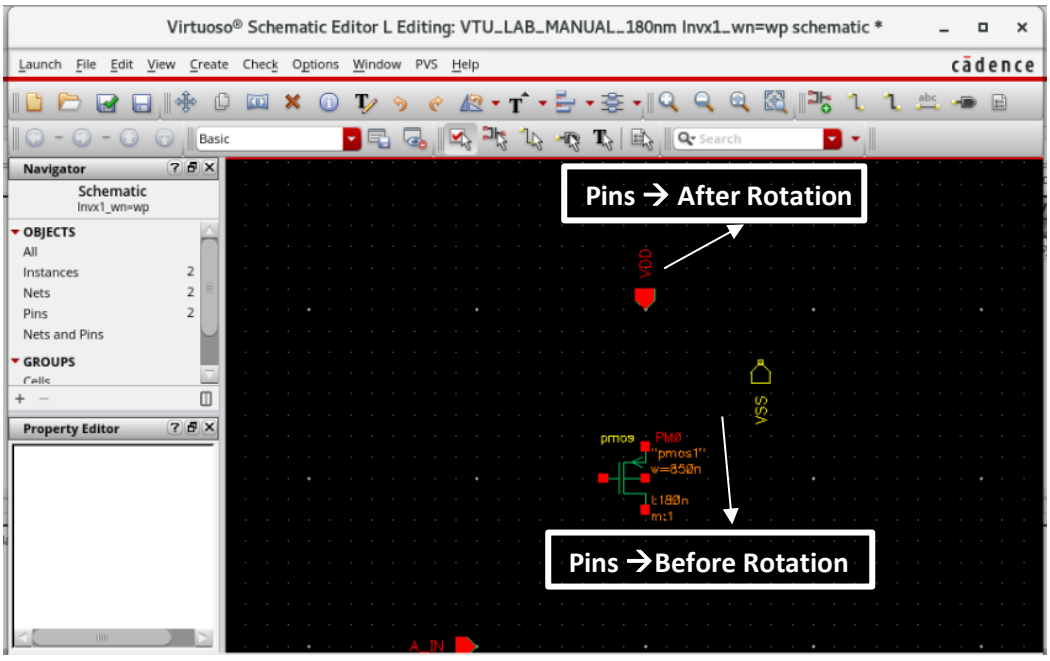



*Figure – 1.22: Pins Before and After Rotation*

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

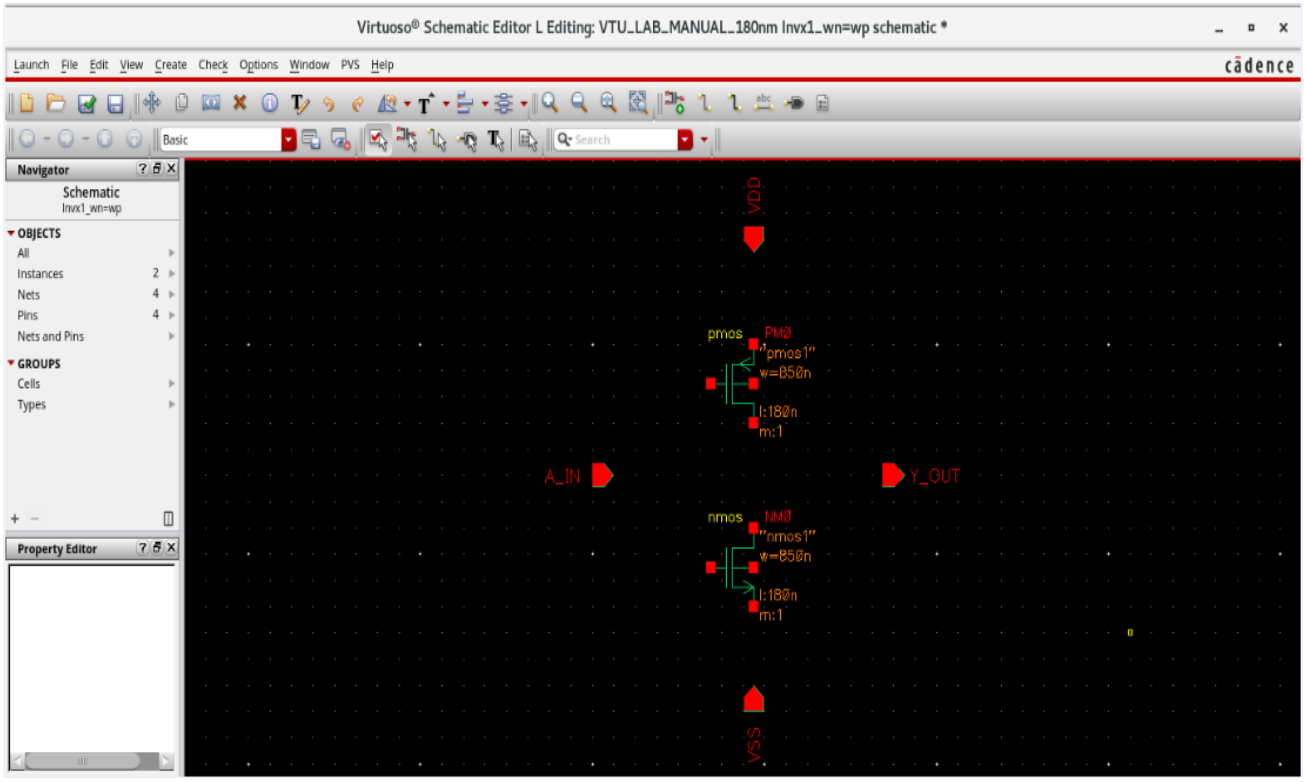



***Figure – 1.23: Schematic Editor after pin placement***

**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.

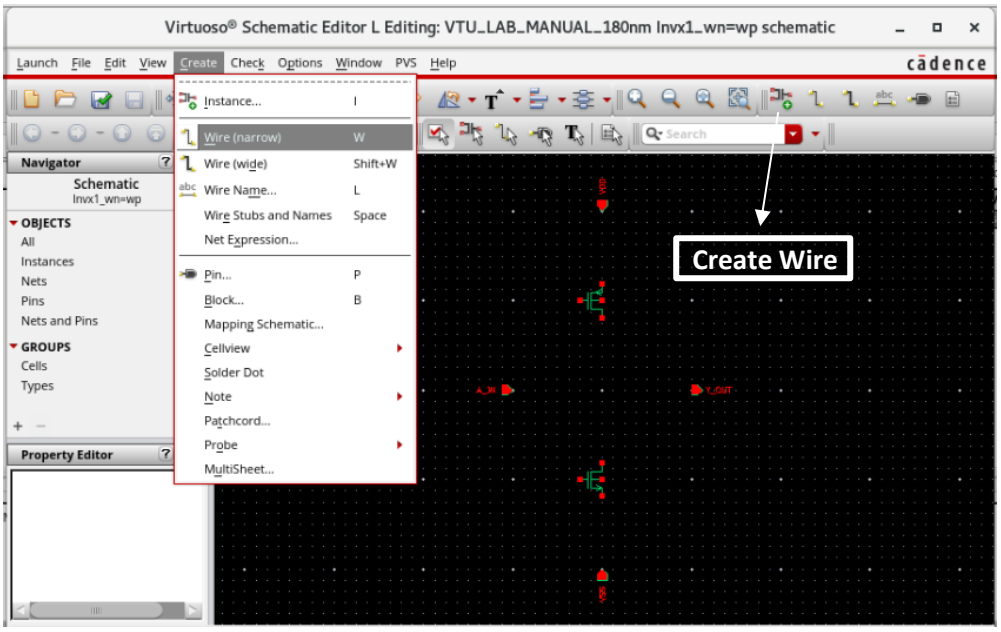



***Figure – 1.24: Create → Wire***

Use the left mouse click to start / complete the wire from one terminal / pin to another. The complete Schematic after connecting the pins and terminalsfor 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.

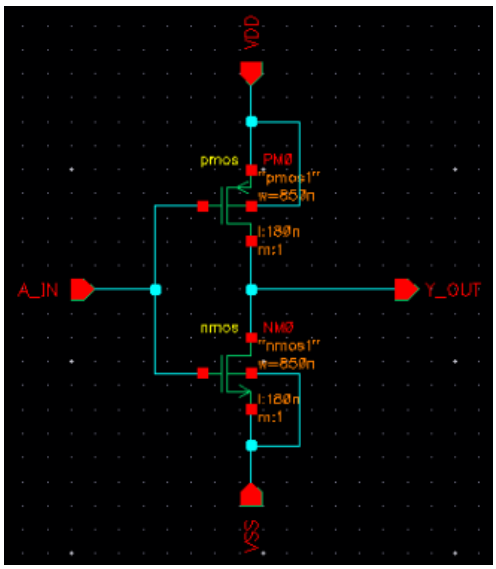



***Figure – 1.25(a):$W_N = W_P$***

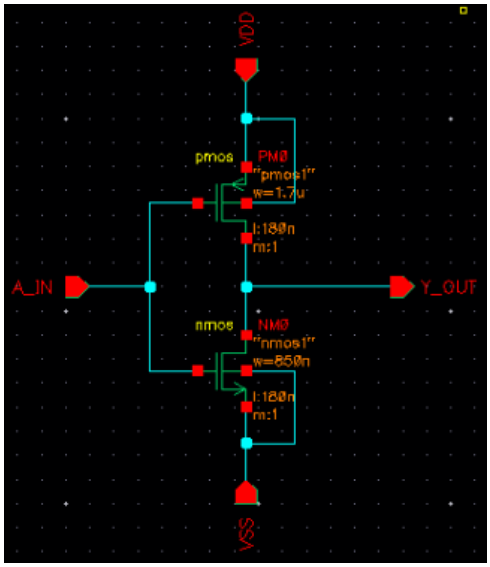



***Figure – 1.25(b): $W_N = 2 * W_P$***

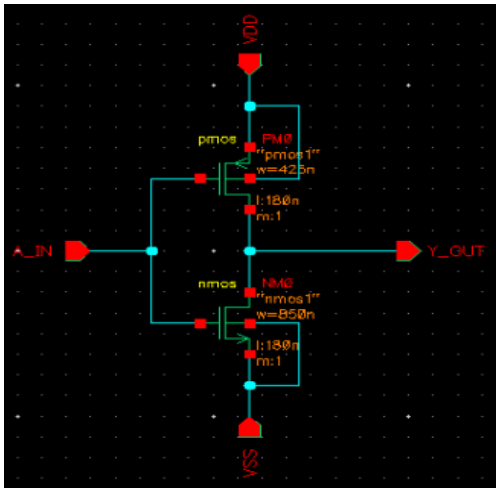



***Figure – 1.25(c): $W_N = W_P / 2$***

## 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.

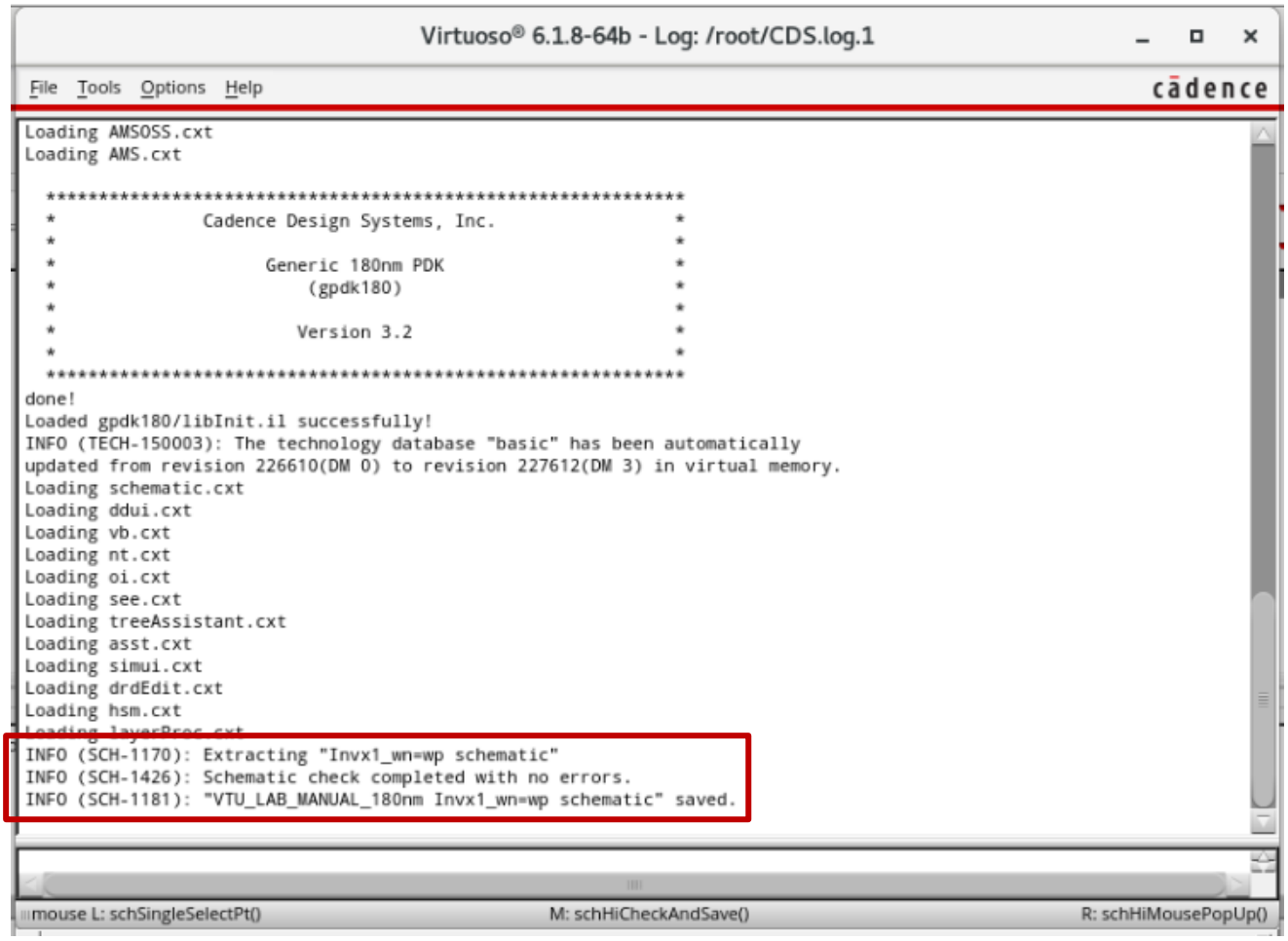



***Figure – 1.27: Message after selecting "Check and Save"***

## 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.

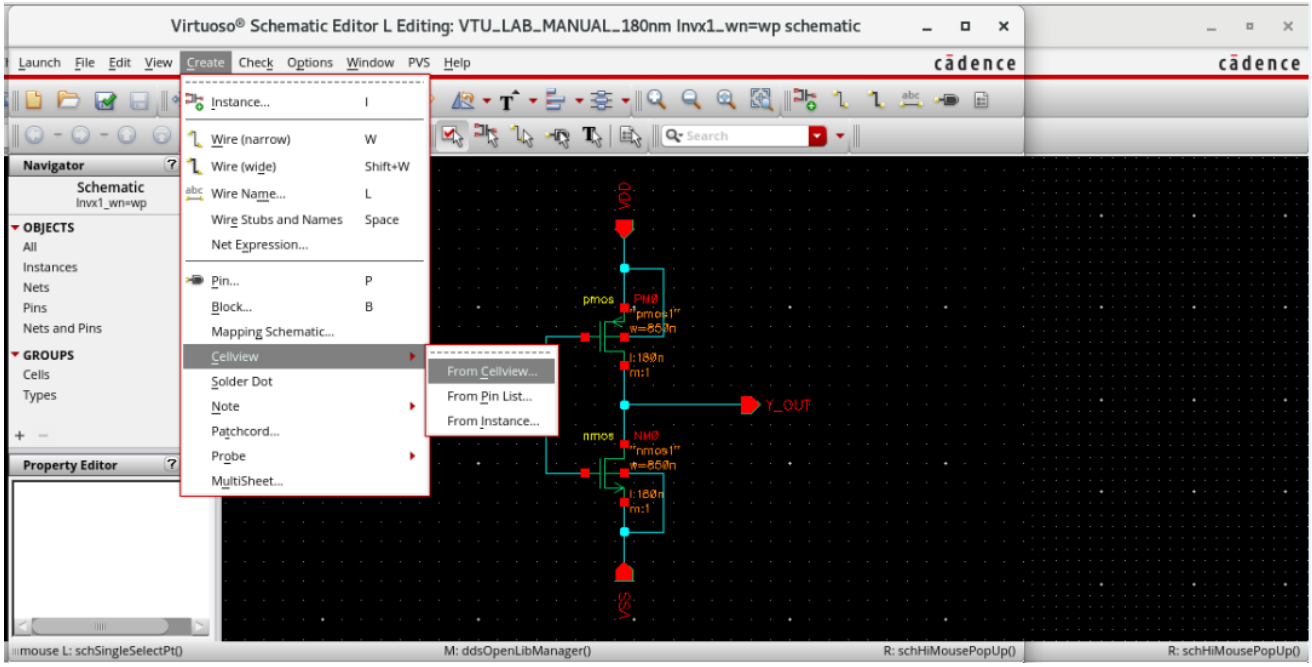



***Figure – 1.28: Create → Cellview → From Cellview***

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

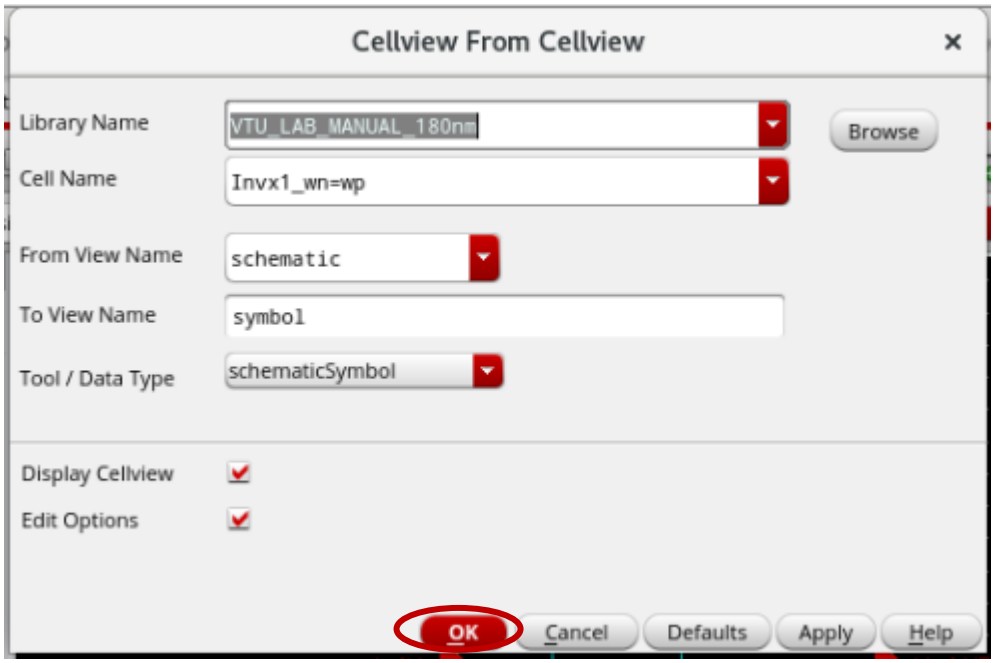



***Figure – 1.29: "Cellview From Cellview" window***

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

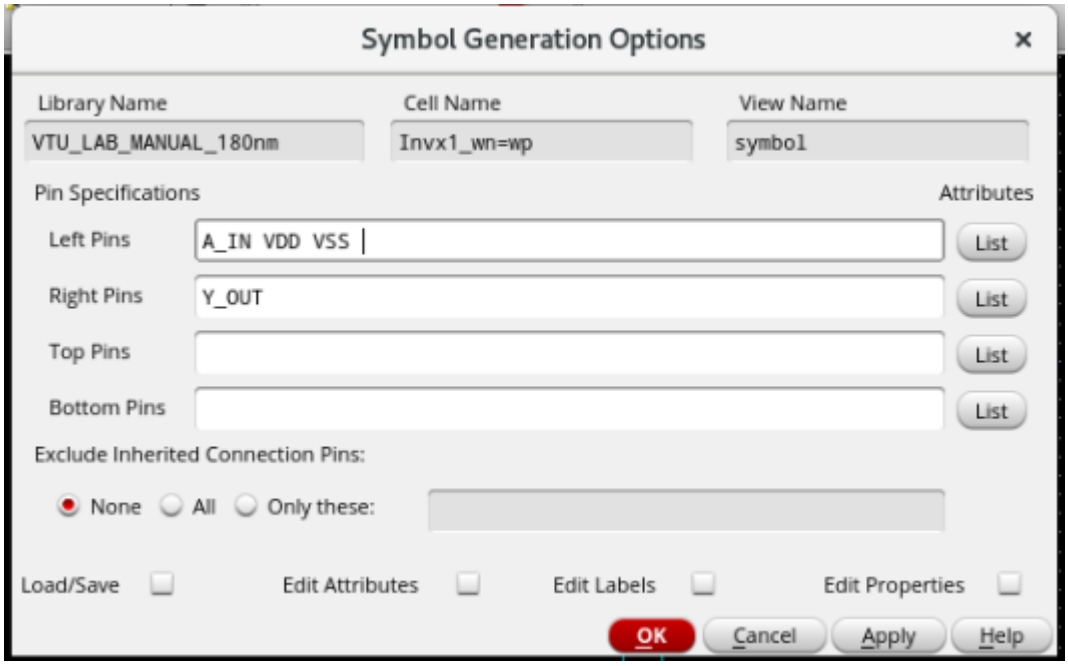



***Figure – 1.30: "Symbol Generation Options" window***

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.

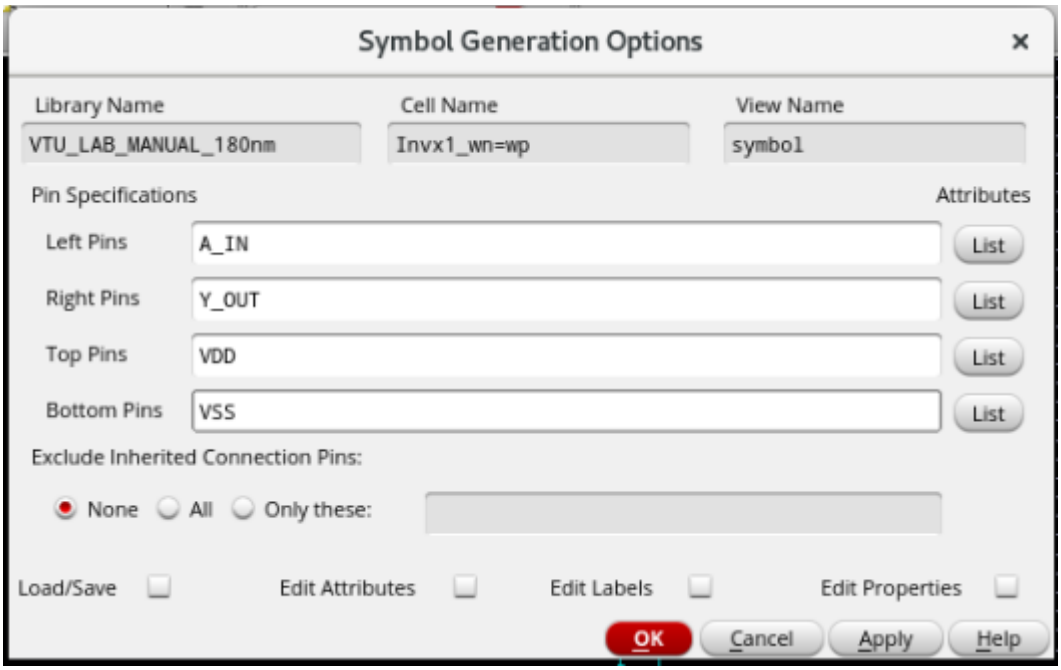



***Figure – 1.31: "Symbol Generation Options" window***

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

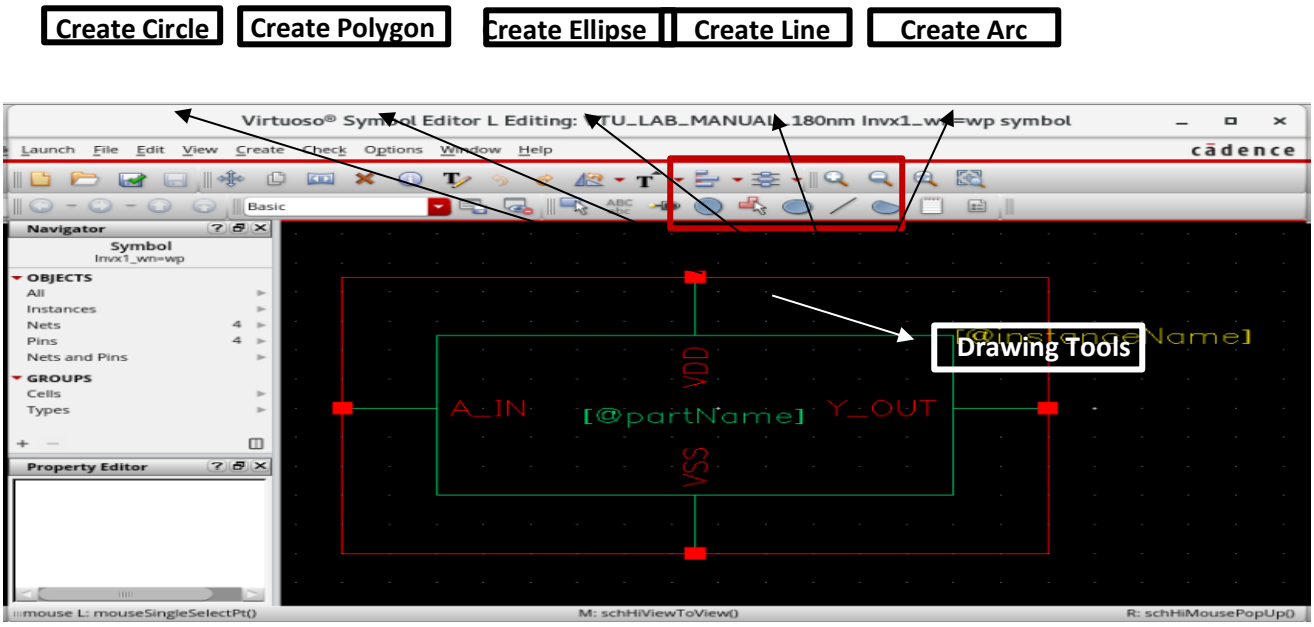
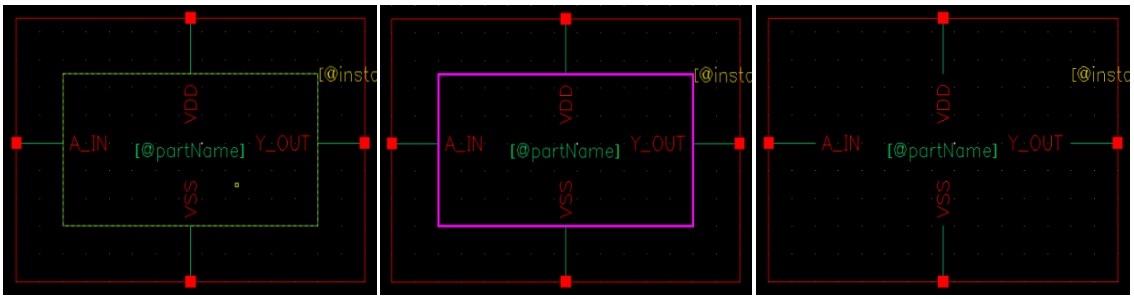



*Figure – 1.32: "Virtuoso Symbol Editor"window with default symbol*

**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 clickto 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.

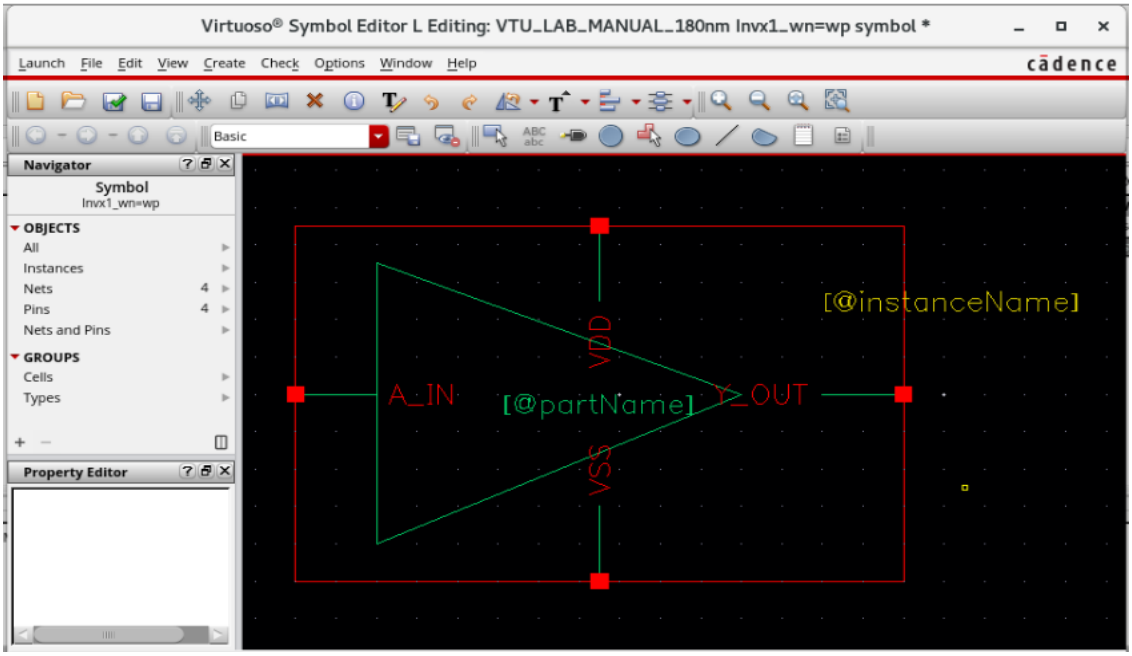
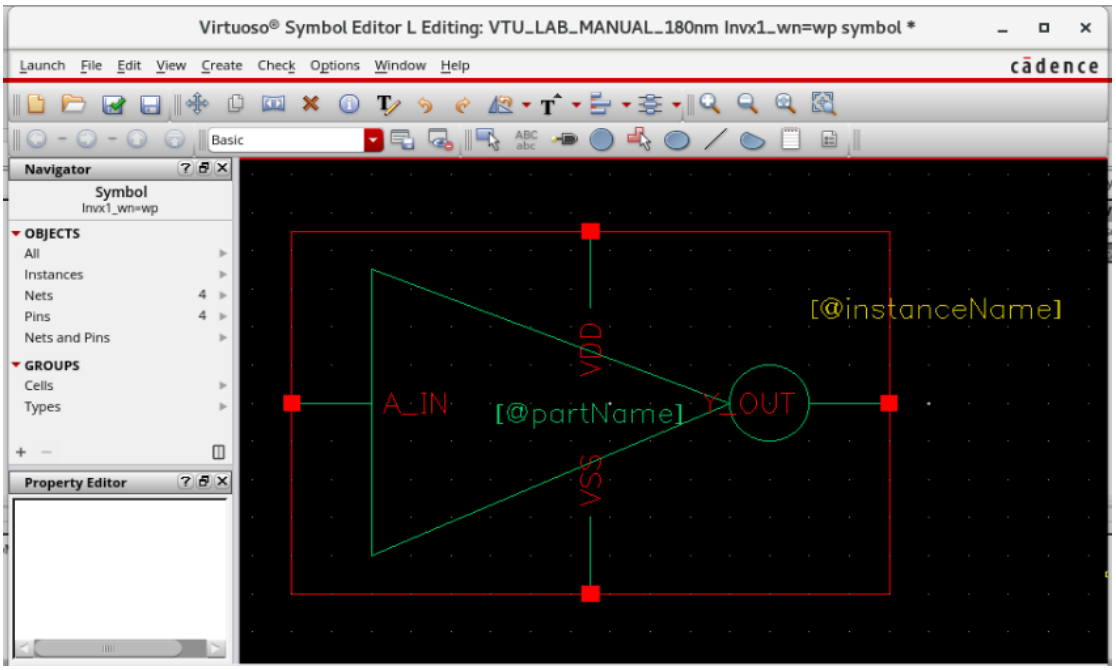



***Figure – 1.34: Symbol after creating the Triangle***

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 aleft 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.

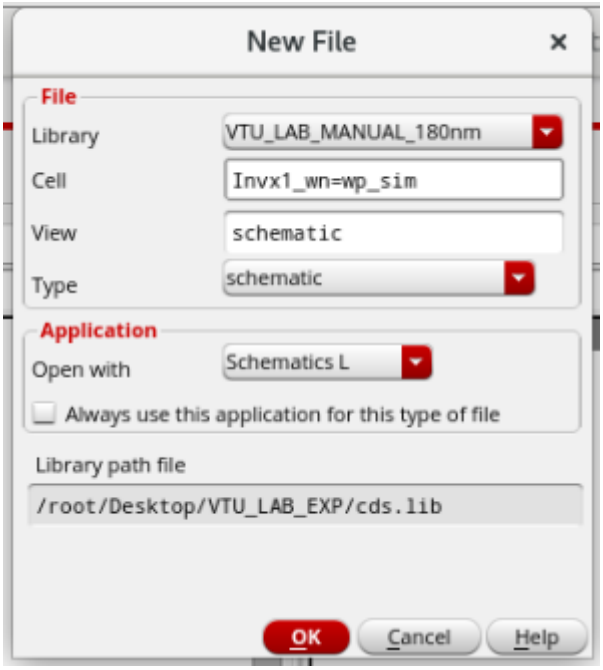



***Figure – 1.36: New Cellview for Test Circuit***

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

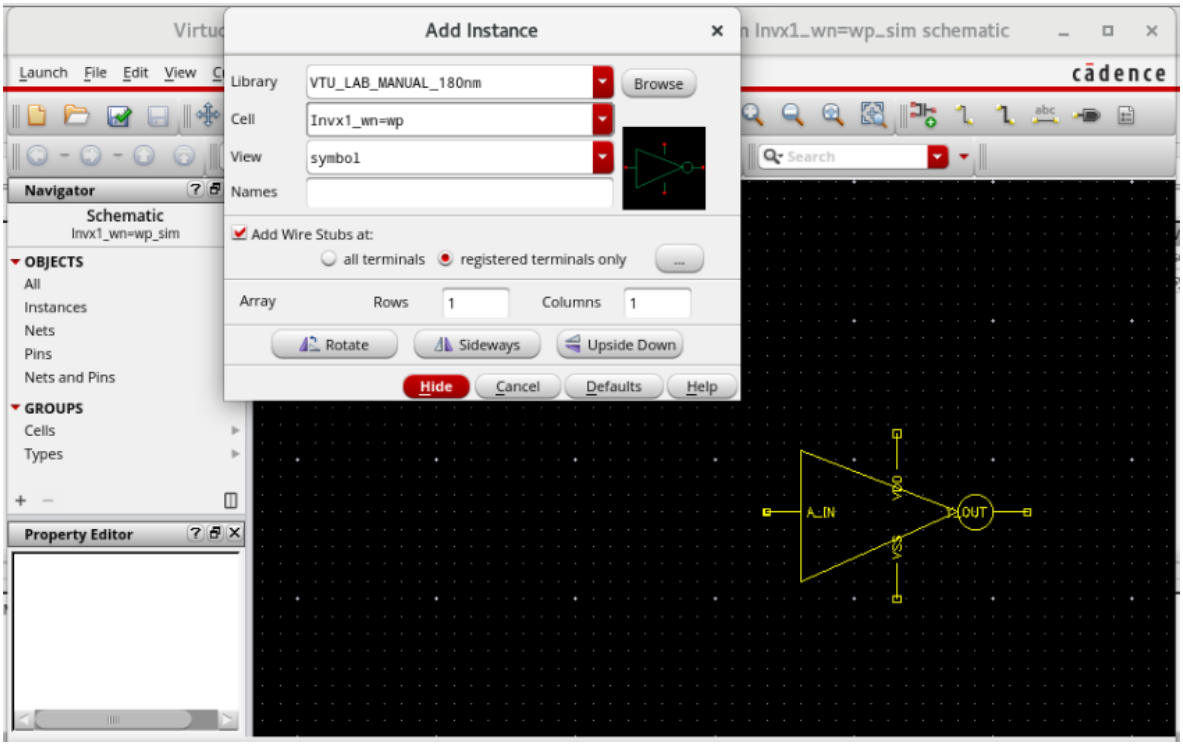



***Figure – 1.37: Symbol Instantiation for the Test Circuit***

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.

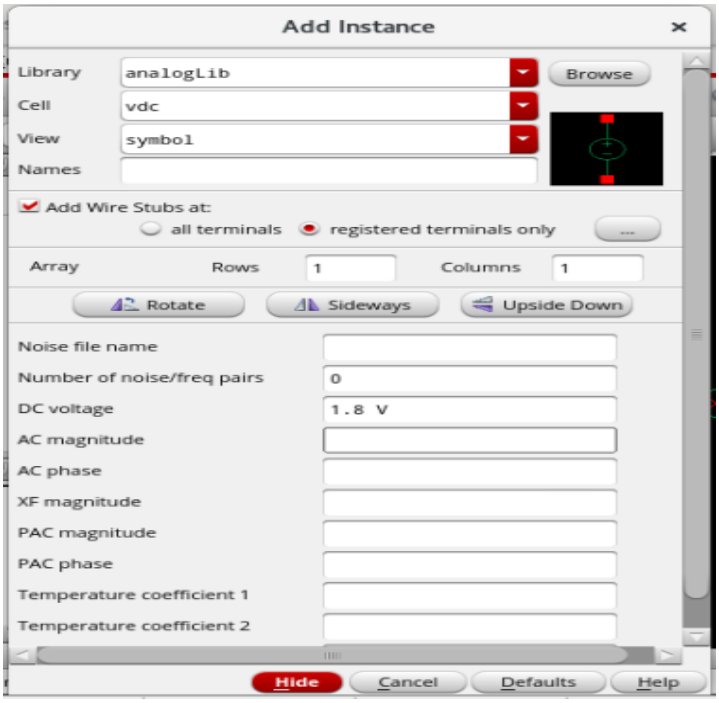



***Figure – 1.38: Instantiating "vdc"***

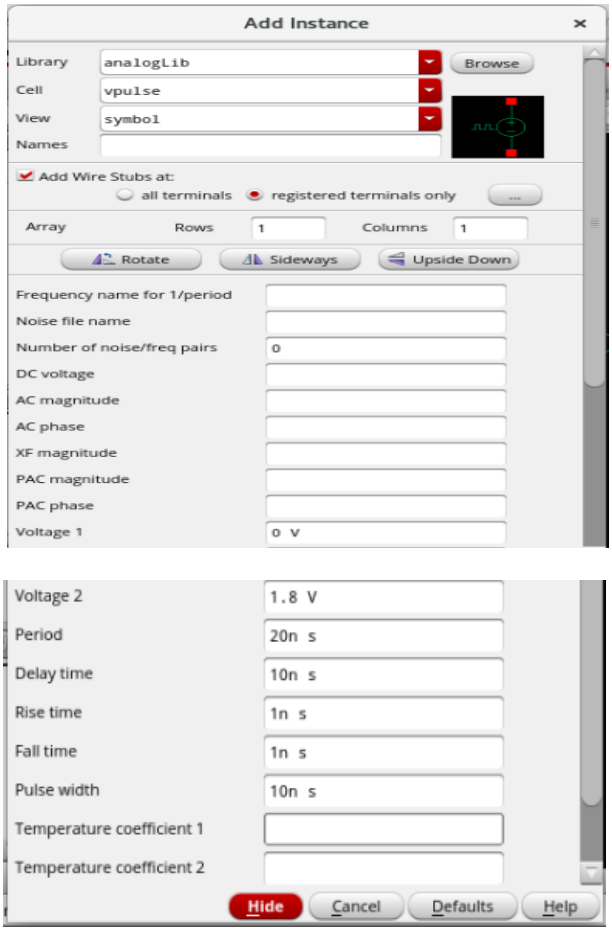



*Figure – 1.39: Instantiating "vpulse"*

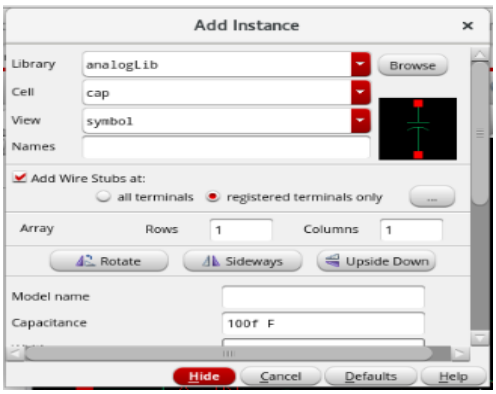



*Figure – 1.40: Instantiating "cap"*


*Figure – 1.41: Instantiating "gnd"*

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

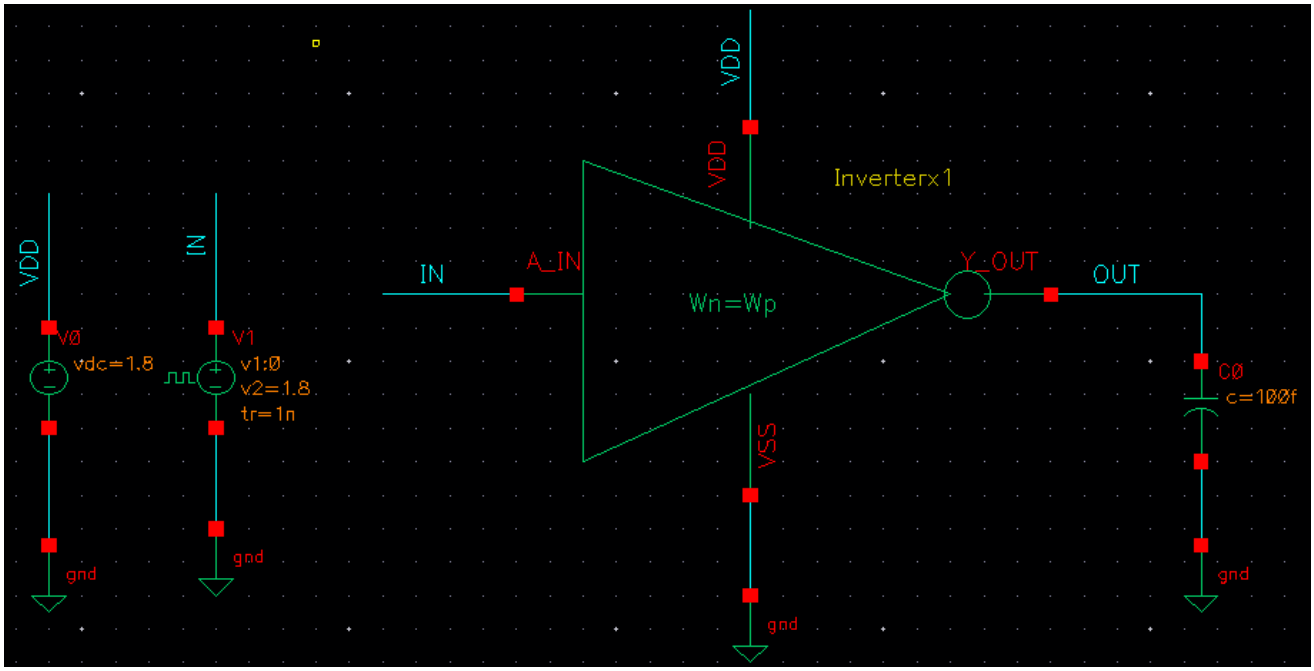



*Figure – 1.42: Complete Test Schematic*

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.

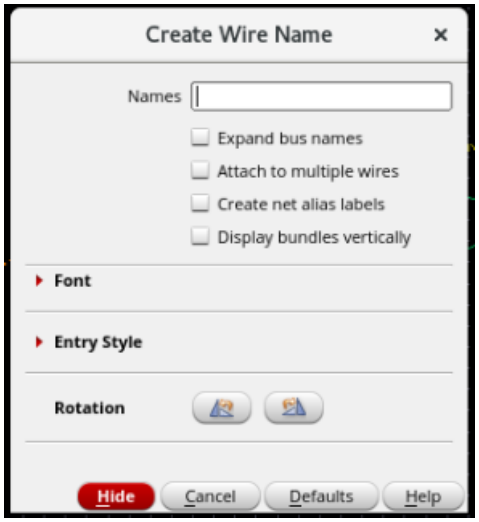




***Figure – 1.43: Create Wire Name Window*** ***Figure – 1.44: Wire Names***

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.

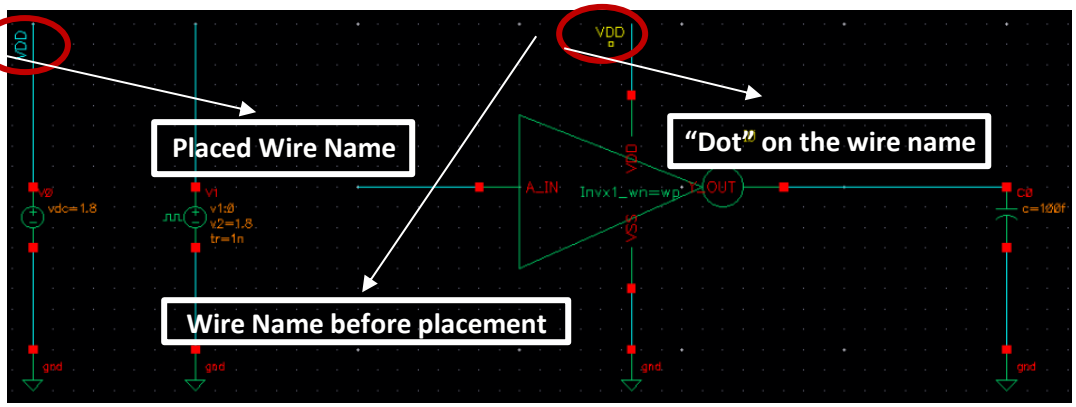



***Figure – 1.45: Wire Name before and after its Placement***

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

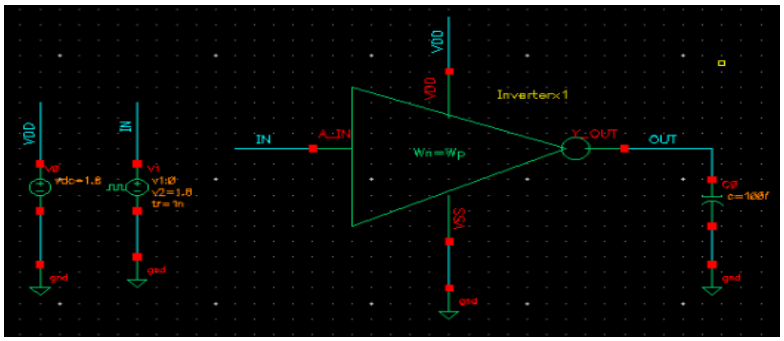



***Figure – 1.46: Complete Test Schematic***

## 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.

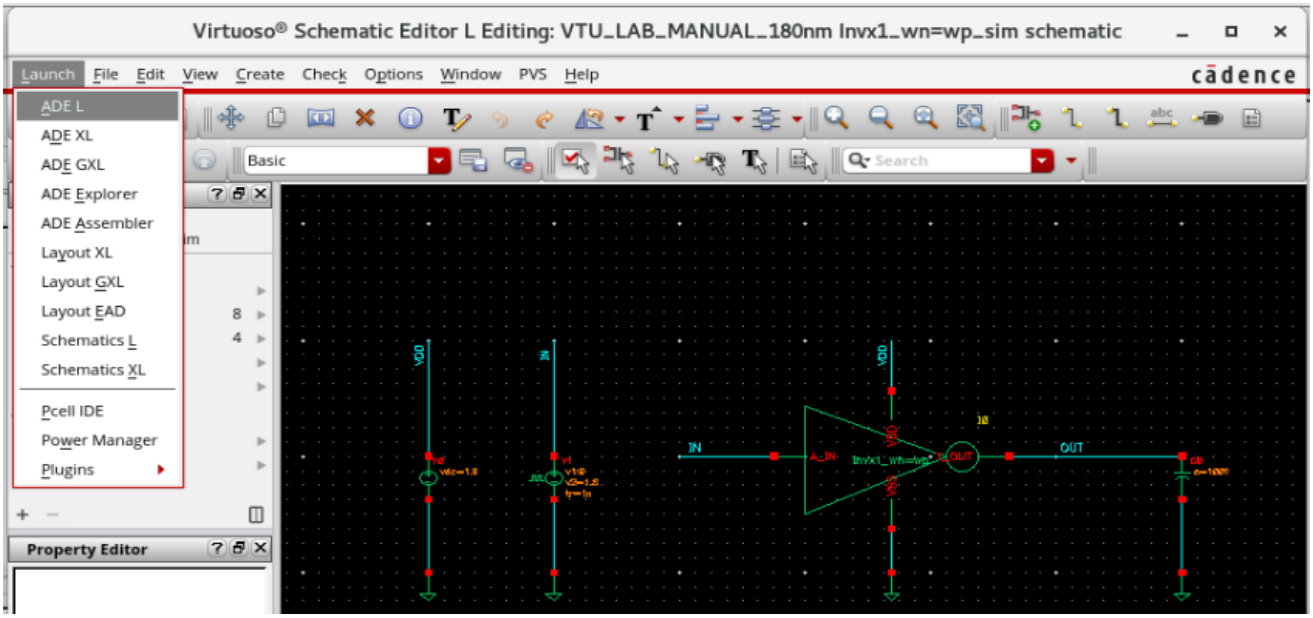



***Figure – 1.47: Launch →ADE L***

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

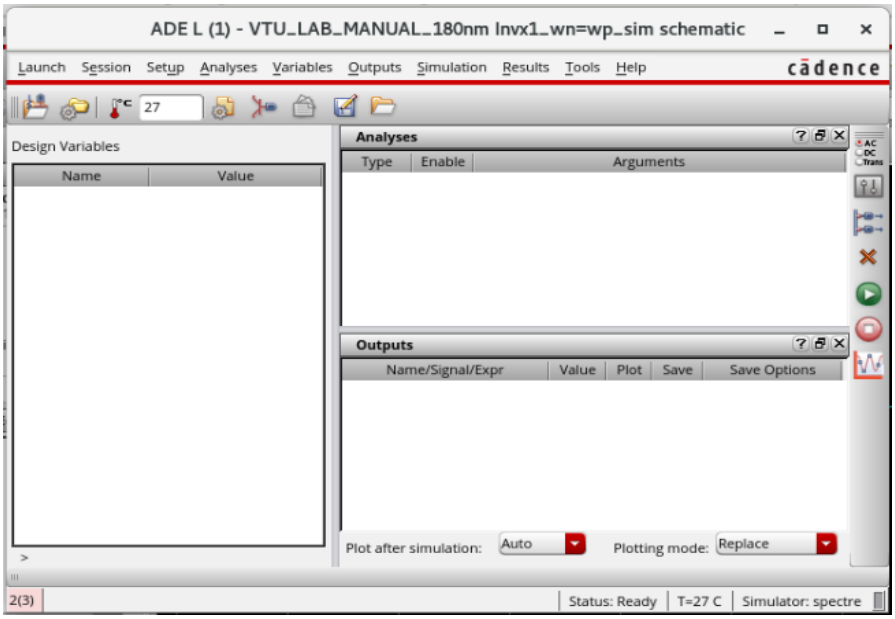



***Figure – 1.48: ADE L Window***

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.

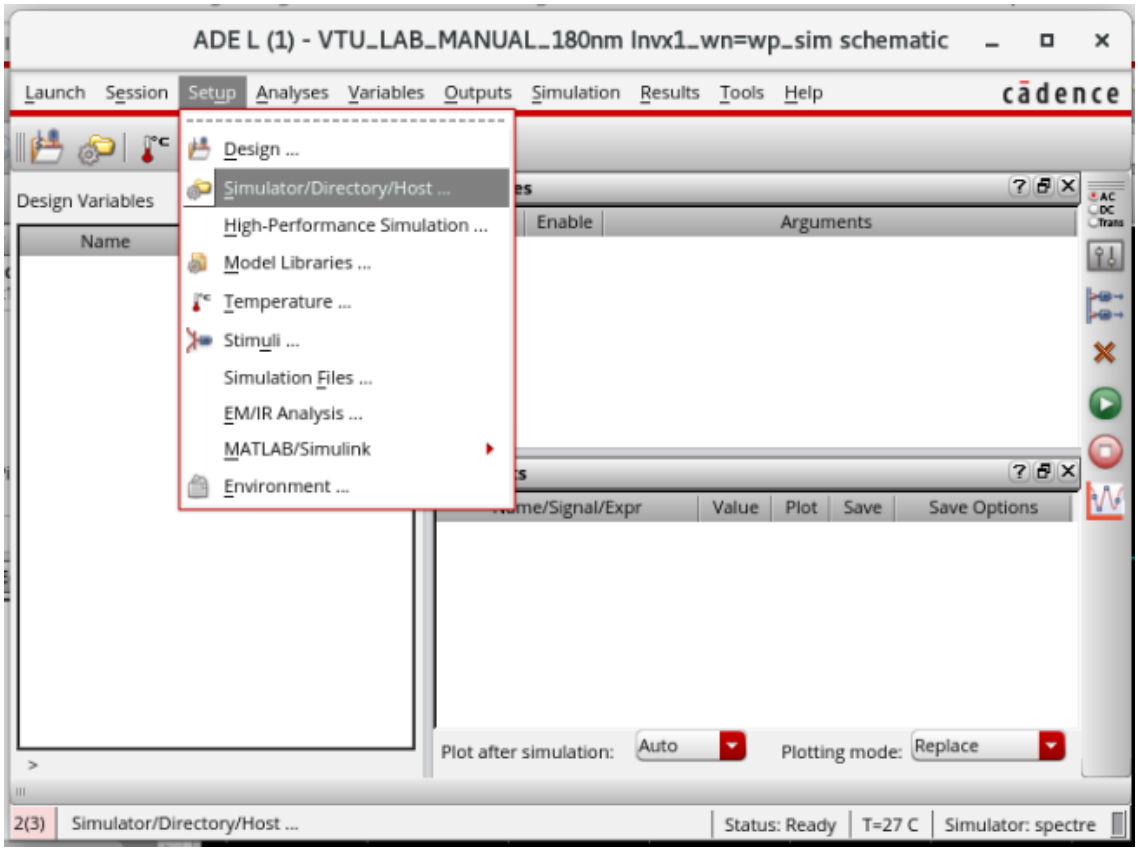



***Figure – 1.49: Setup → Simulator/Directory/Host..***

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

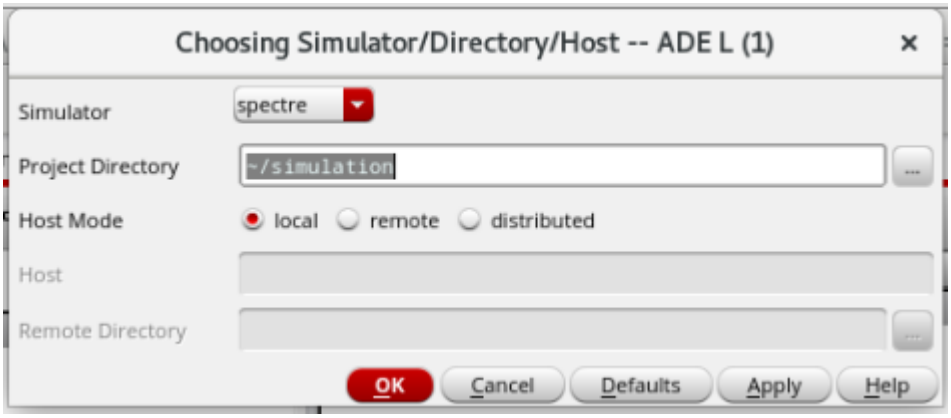



***Figure – 1.50: Simulator → Spectre***

## SELECTING THE MODEL LIBRARIES AND PROCESS CORNERS:

The Model Libraries and Process Cornersare important to run the simulation.
To select the "**.scs**" file with respect to the technology node, select "**Setup → Model Libraries**"as shown in Figure – 1.51.

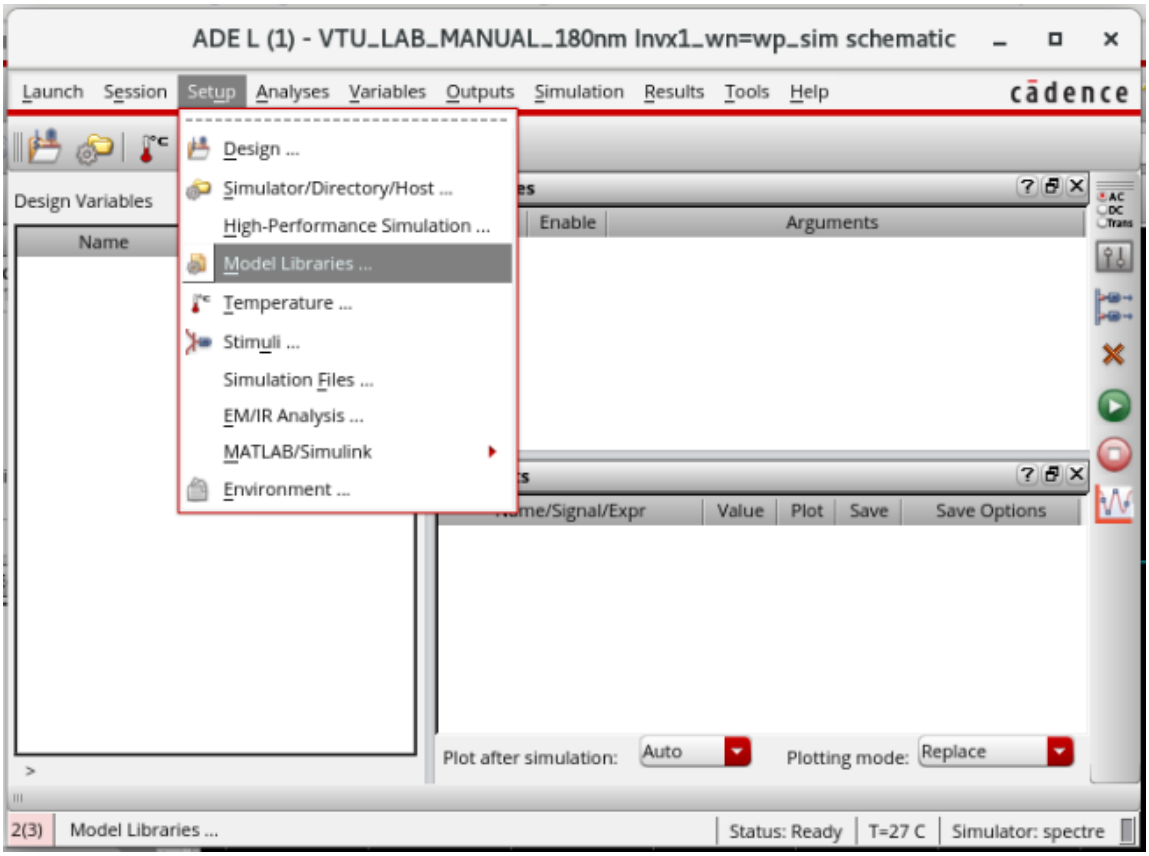



***Figure – 1.51: Setup → Model Libraries***

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

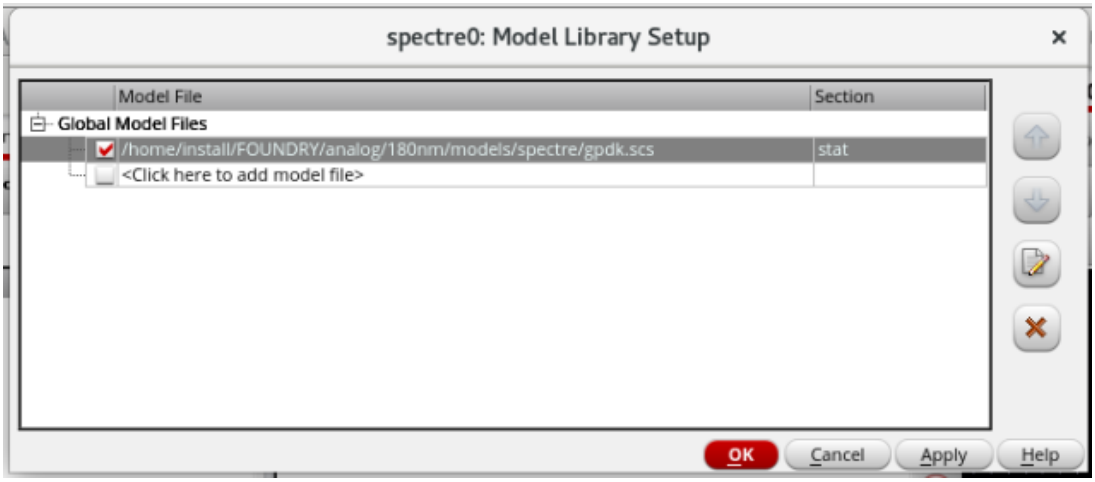



***Figure – 1.52: "spectre0: Model Library Setup" Window***

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.

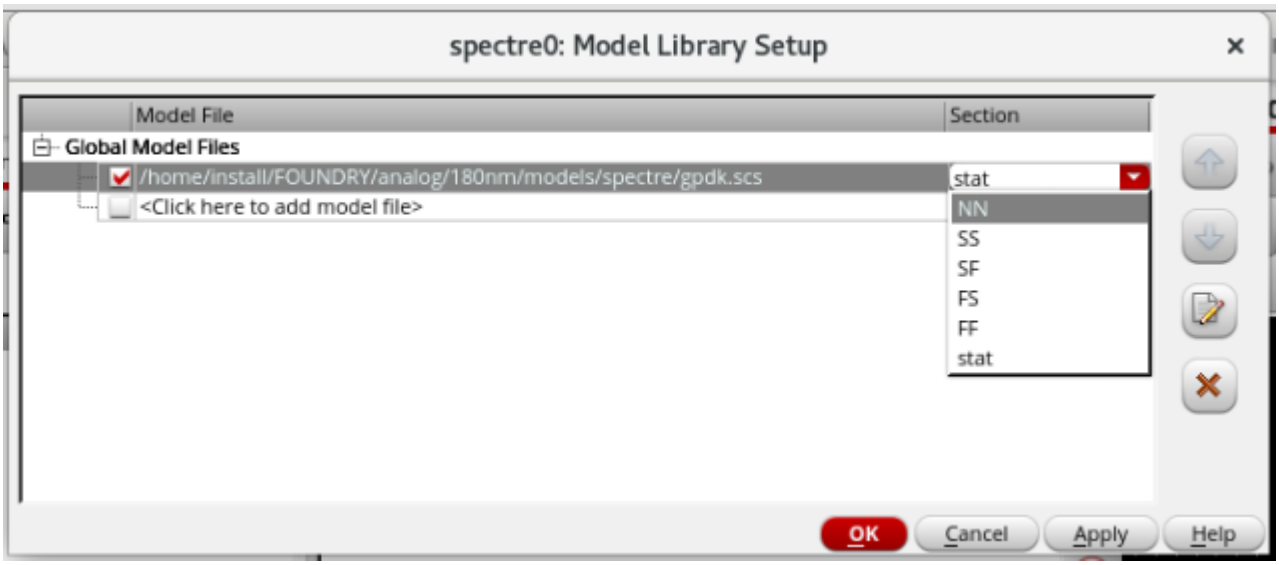



*Figure – 1.53: ".scs" file and Processing Corner Selection*

**SELECTING THE ANALYSIS:**

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

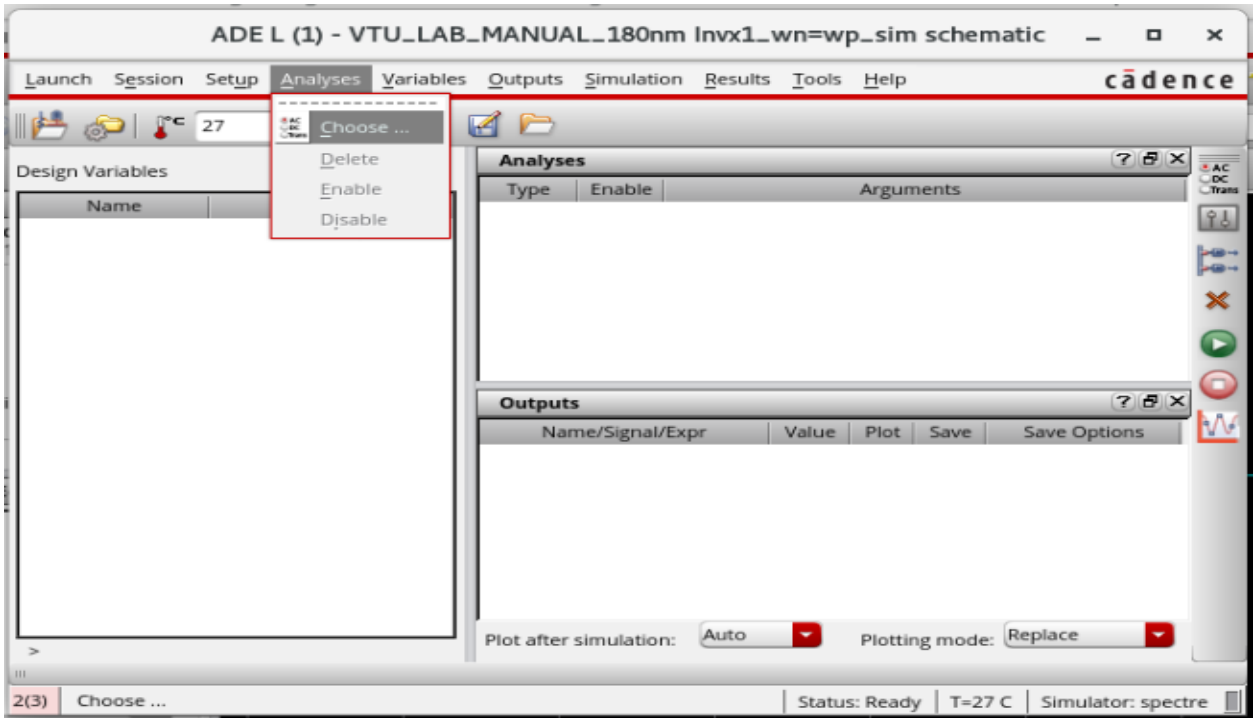



*Figure – 1.54: Analyses → Choose*

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

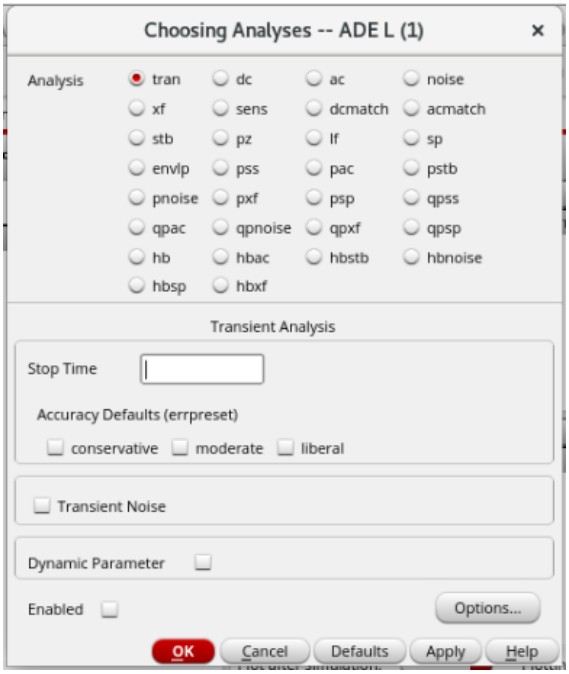



***Figure – 1.55: "Choosing Analysis – ADE L" Window***

**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.

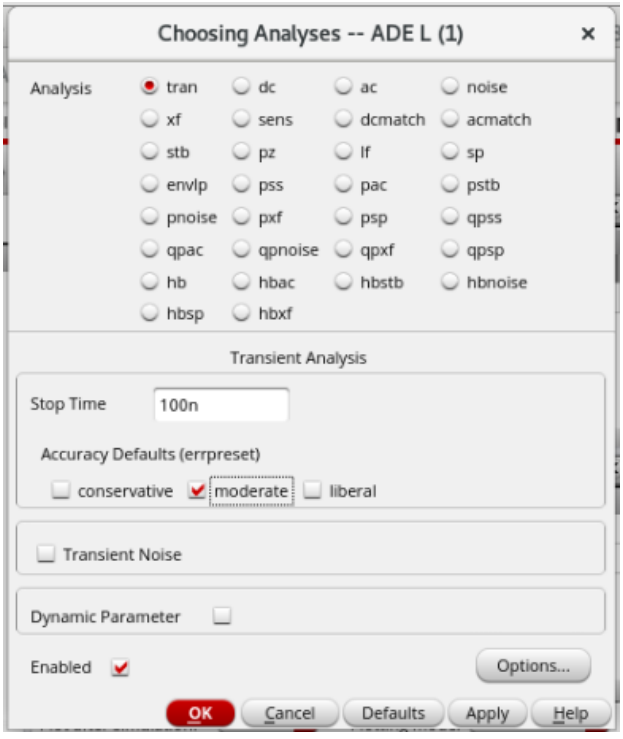



***Figure – 1.56: Setup for Transient Analysis***

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

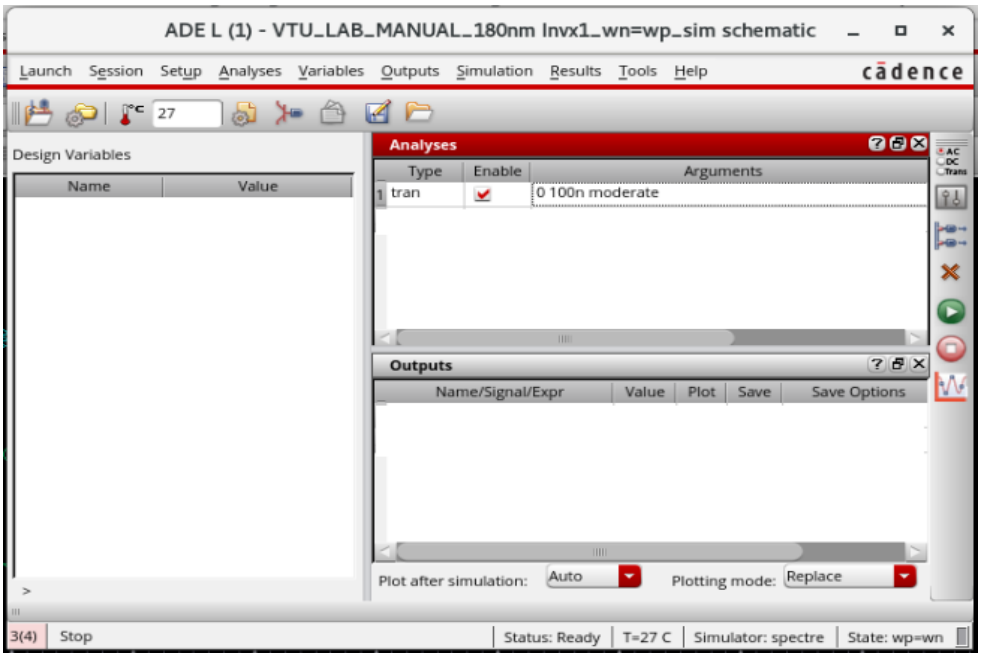



***Figure – 1.57: Chosen Analysis in the ADE L window***

**DC ANALYSIS:**

To set up a "**DC Analysis**", select "**dc**" and enable "**Save DC Operating Point**" as shown inFigure – 1.58.

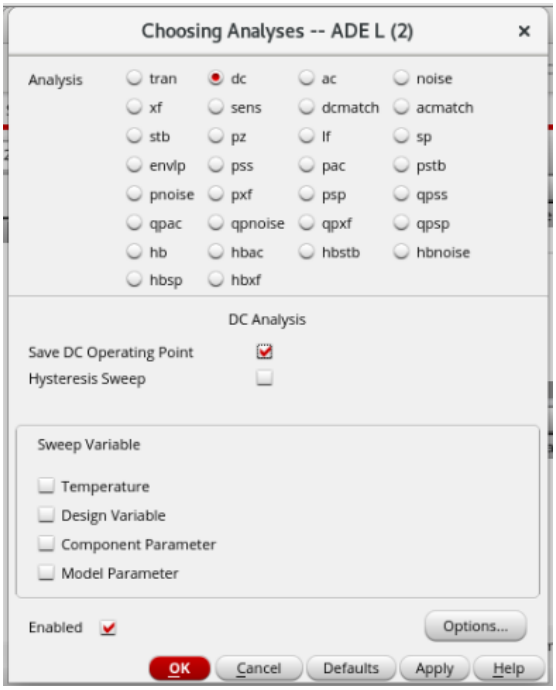



***Figure – 1.58: Select "dc"***

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

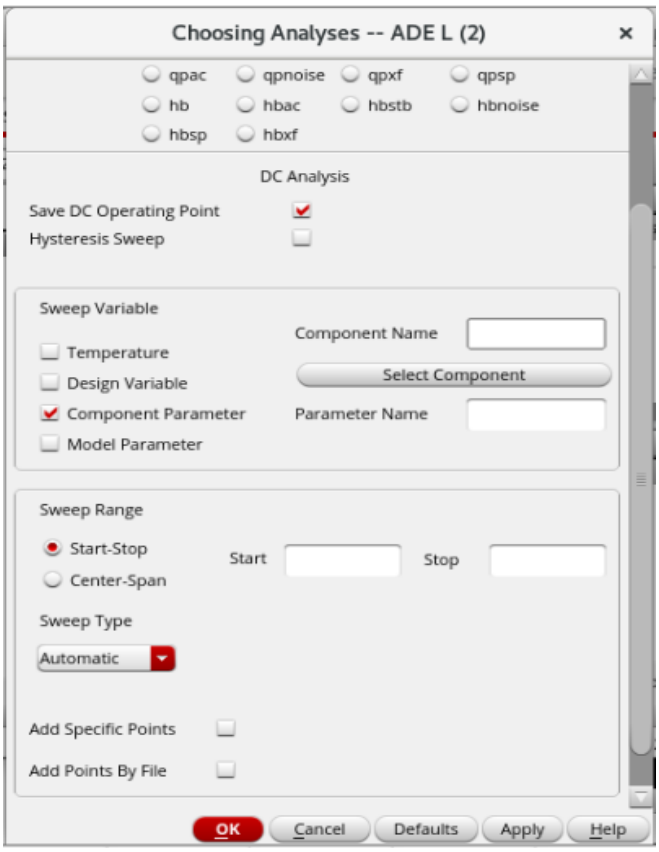



***Figure – 1.59: Enable "Component Parameter"***

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

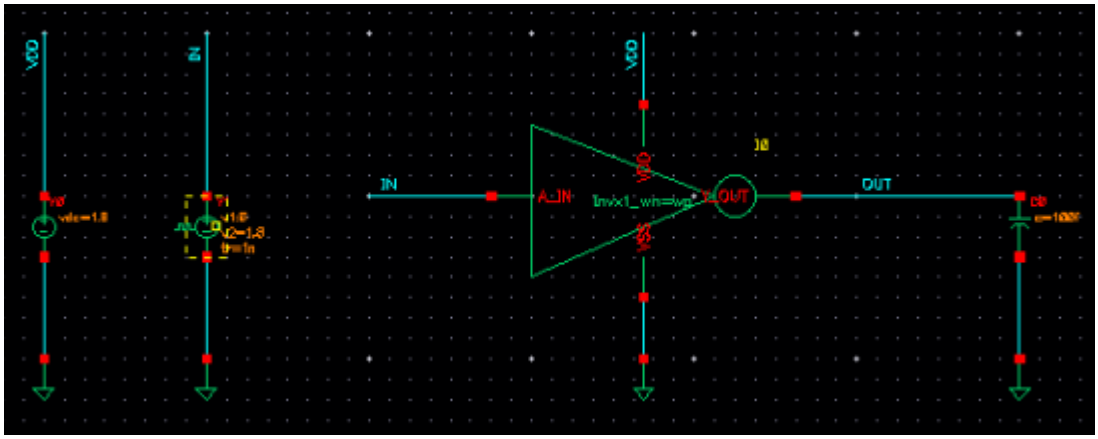



***Figure – 1.60: Selecting "vpulse" from the Test Schematic***

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.

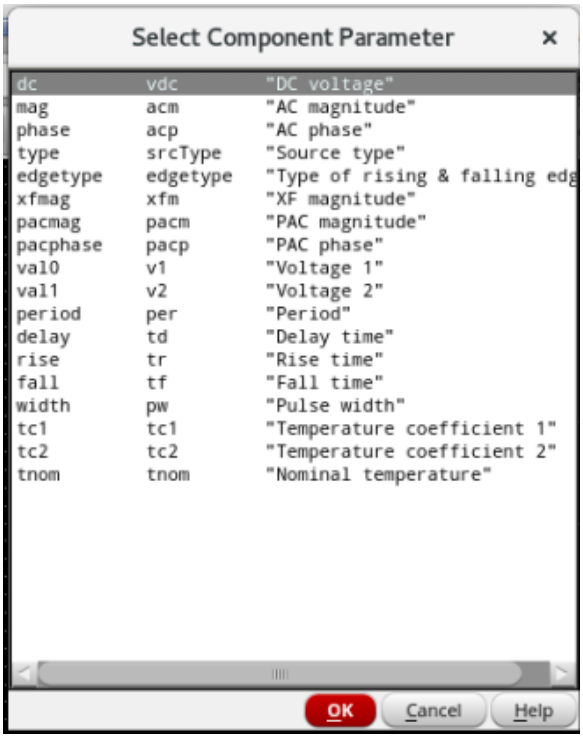



***Figure – 1.61: "Select Component Parameter" window***

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.

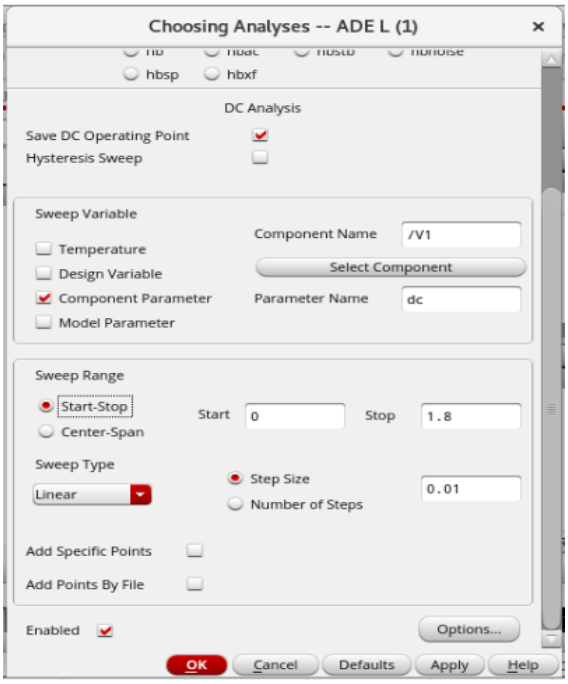



***Figure – 1.62: Mention the Sweep Range***

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

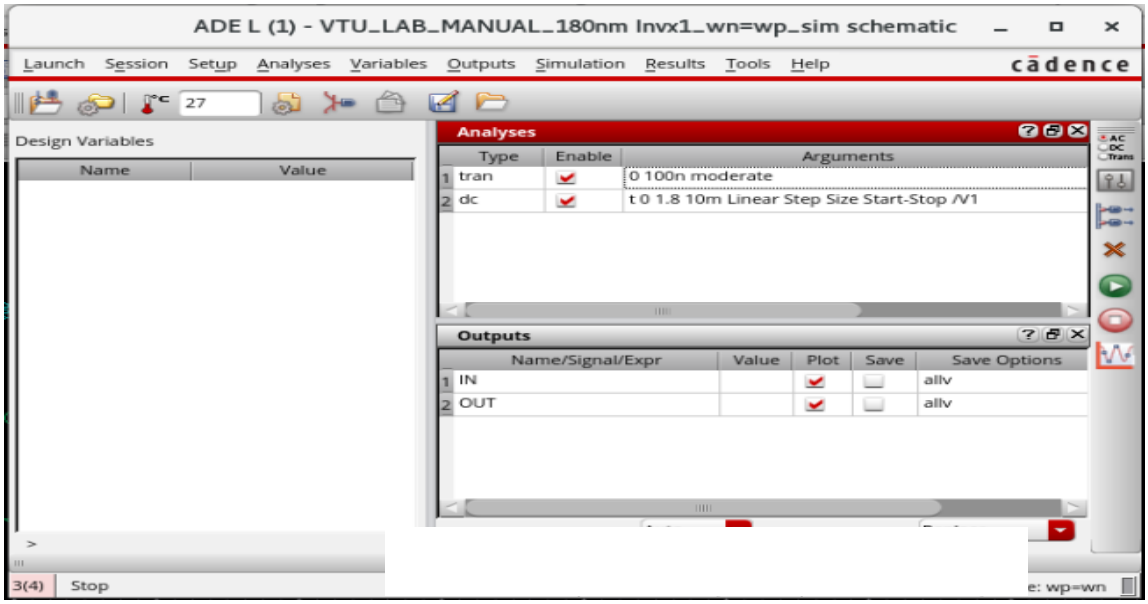



***Figure – 1.63: Updated ADE L window***

**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.

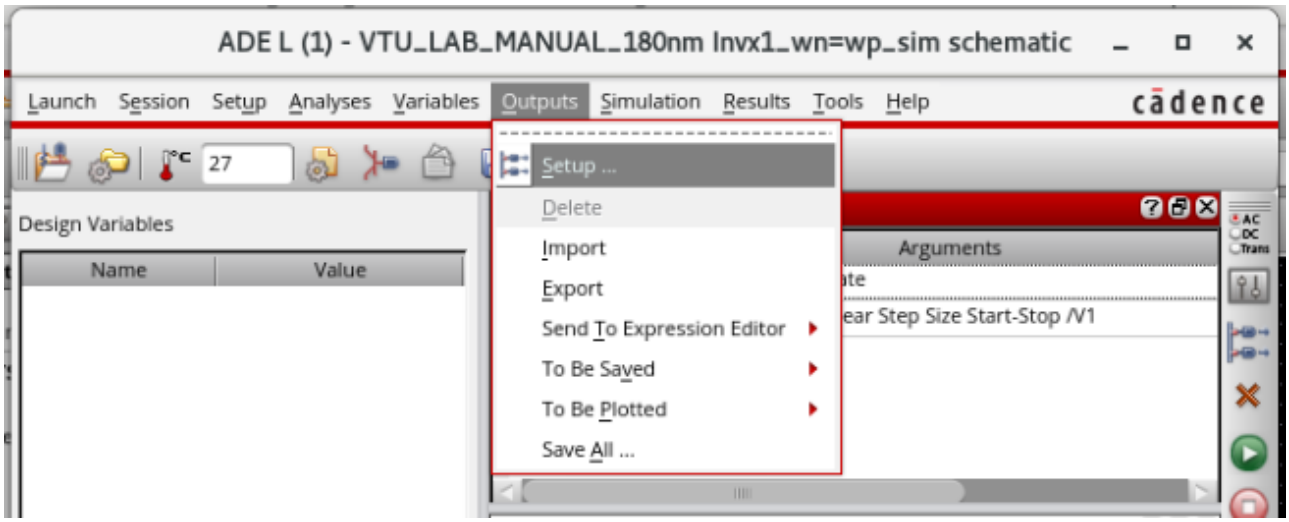



***Figure – 1.64: Outputs → Setup***

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

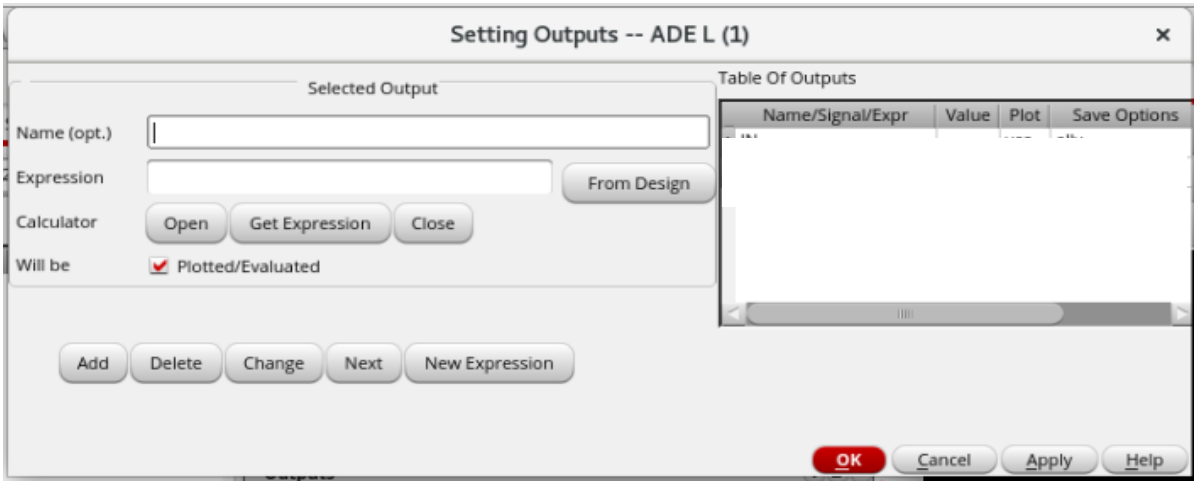



***Figure – 1.65: Setting Outputs – ADE L window***

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

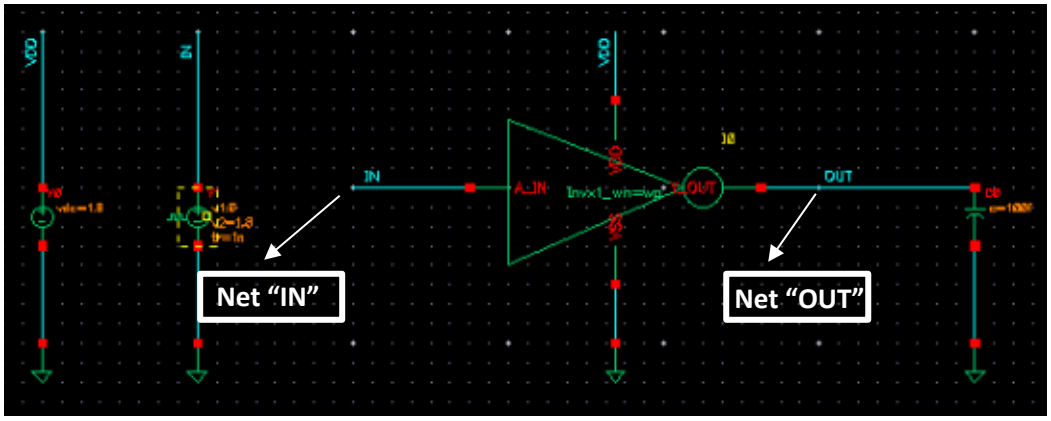



*Figure – 1.66: Select the Net "IN" and Net "OUT"*

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**".

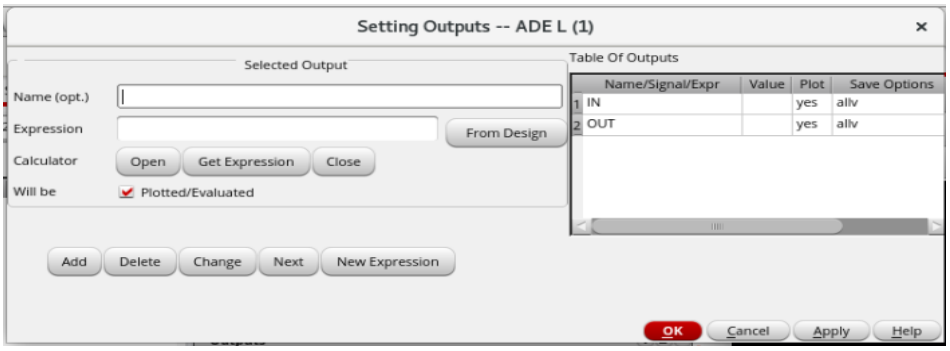



*Figure – 1.67: Updated "Setting Outputs" window*

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

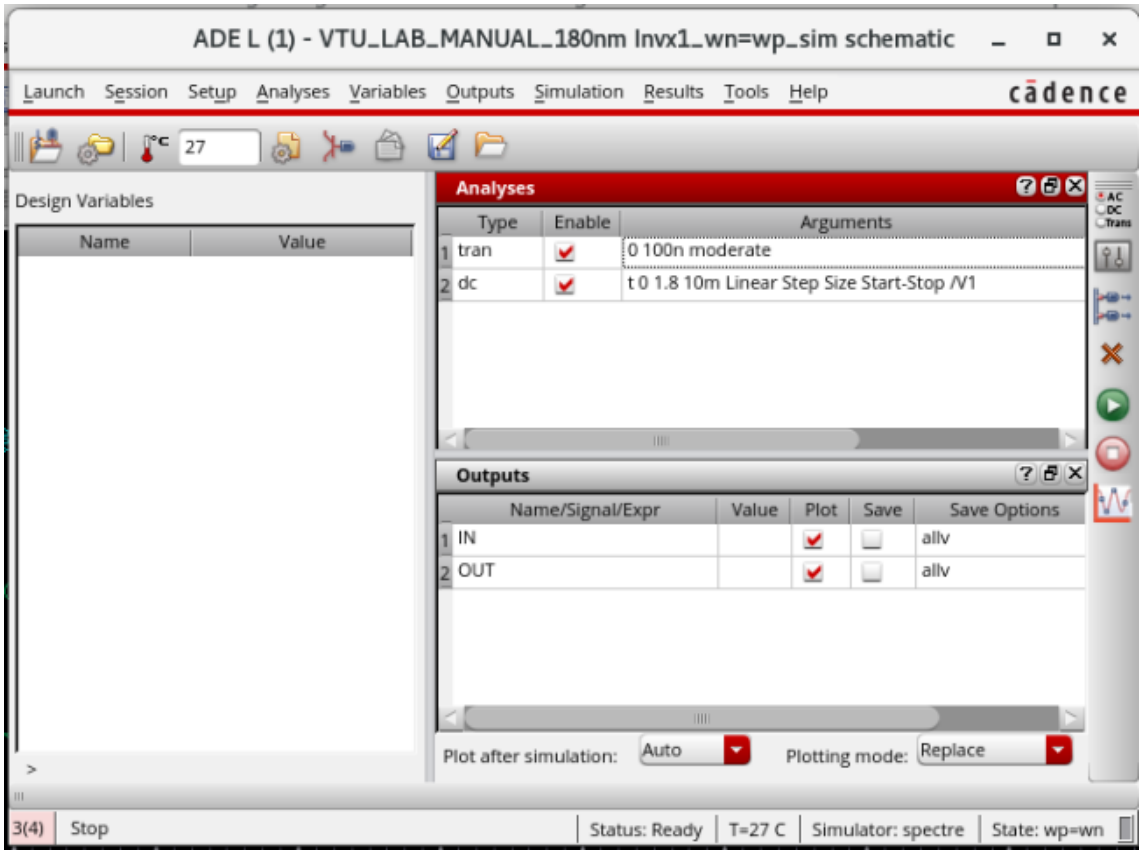



*Figure – 1.68: "Outputs" in ADE L window*

**RUNNING THE SIMULATION:**

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

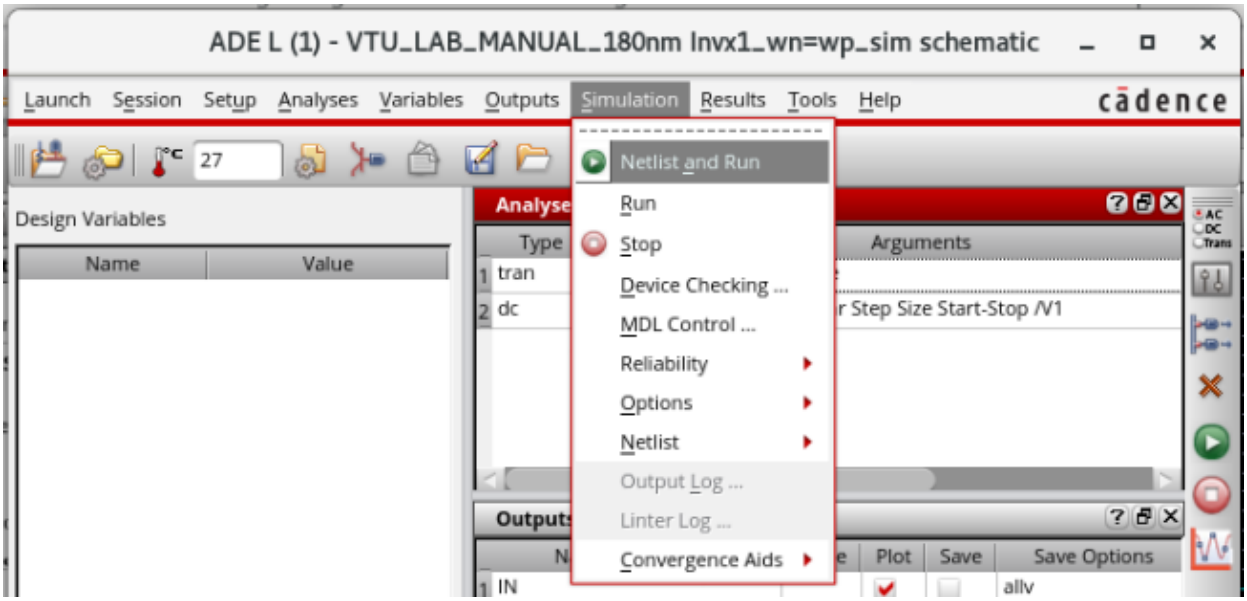



*Figure – 1.69: Simulation → Netlist and Run*

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

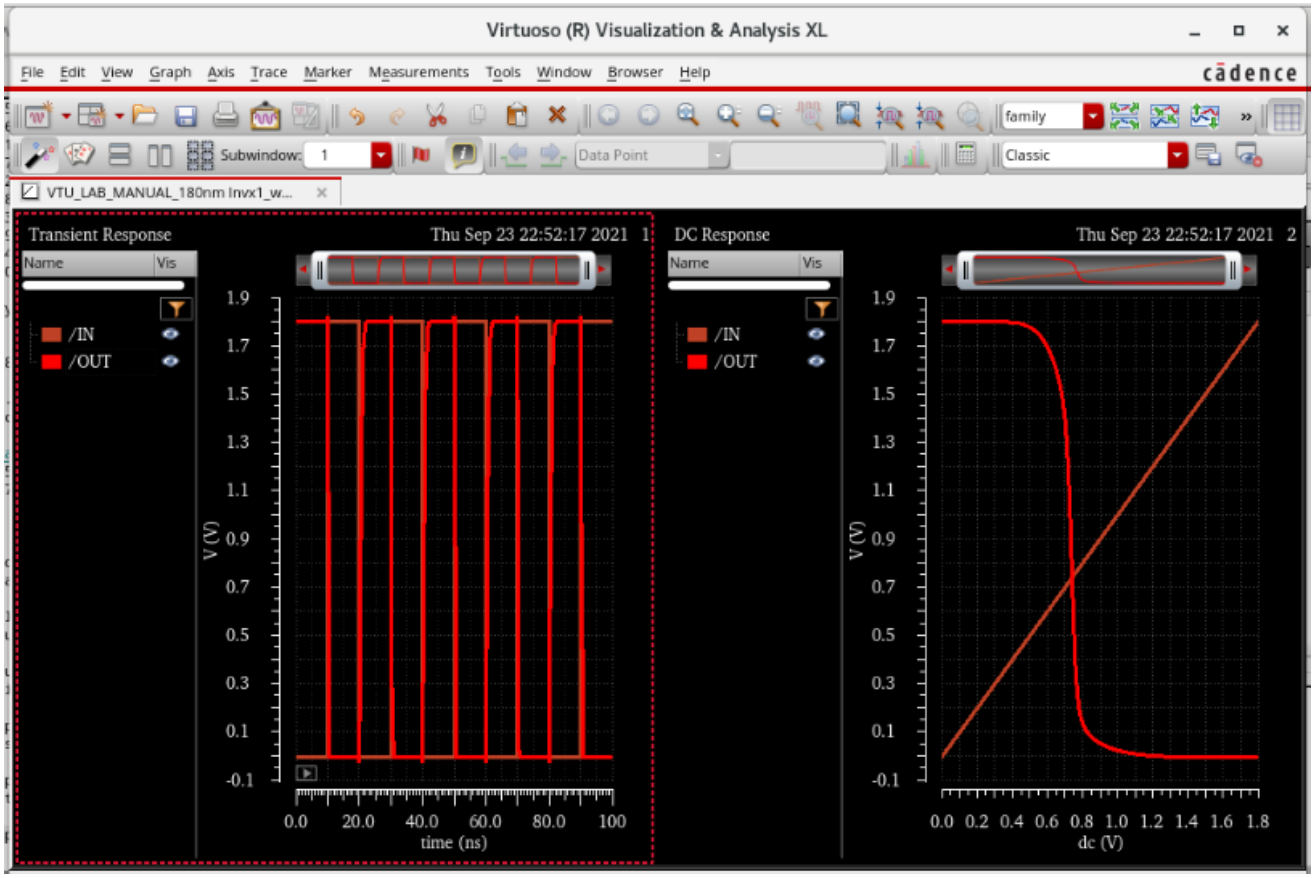



***Figure – 1.70: Simulated waveforms***

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

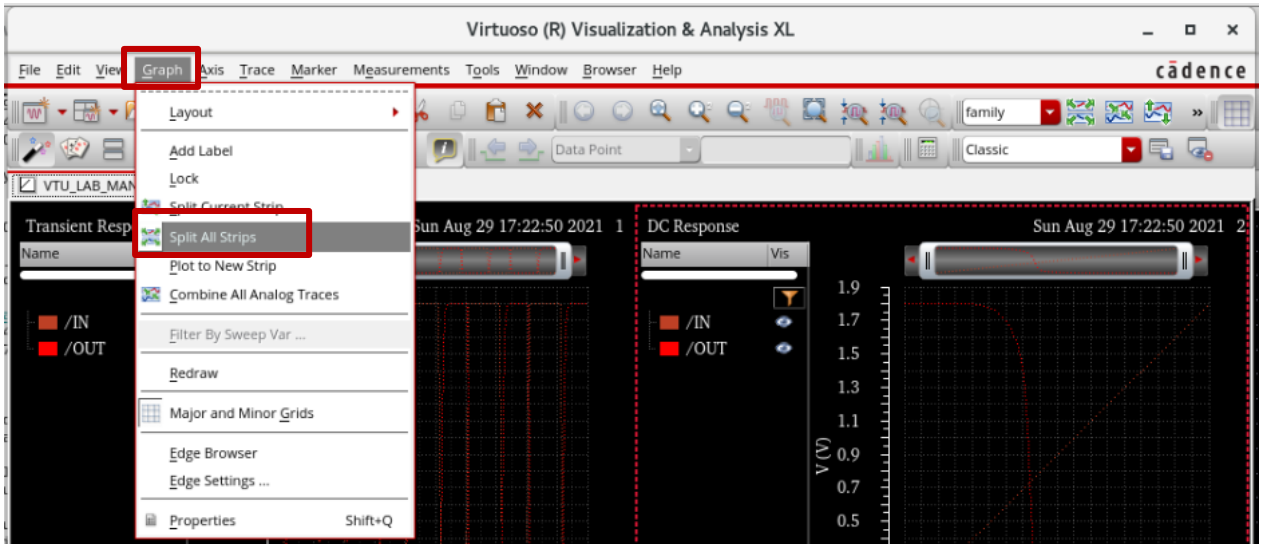



***Figure – 1.71: Graph → Split All Strips***

## SAVING THE ADE L STATE:

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

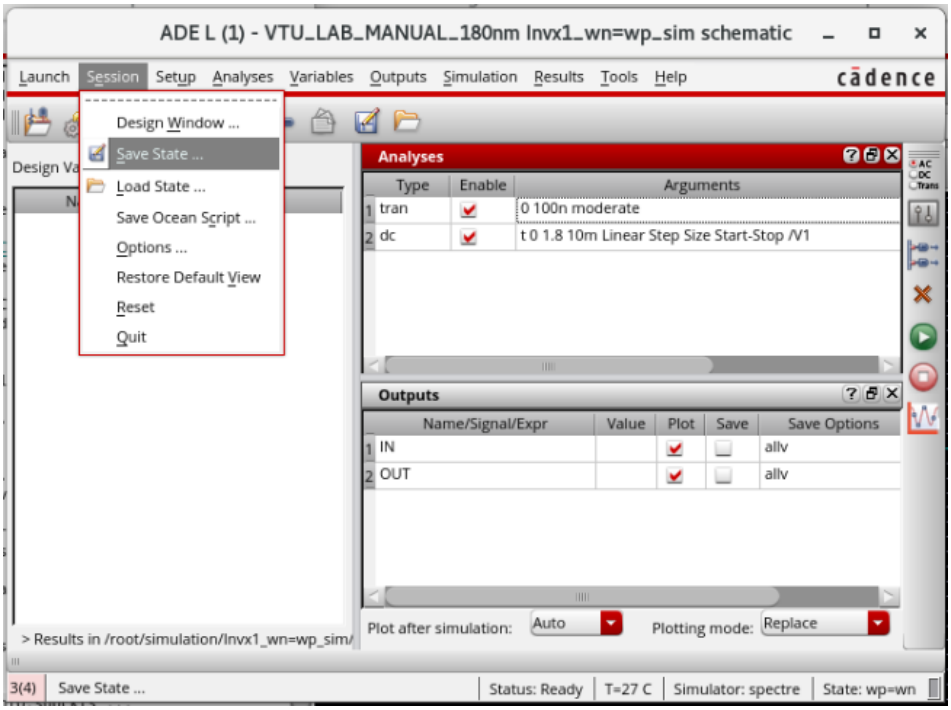



***Figure – 1.72: Session → Save State***

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

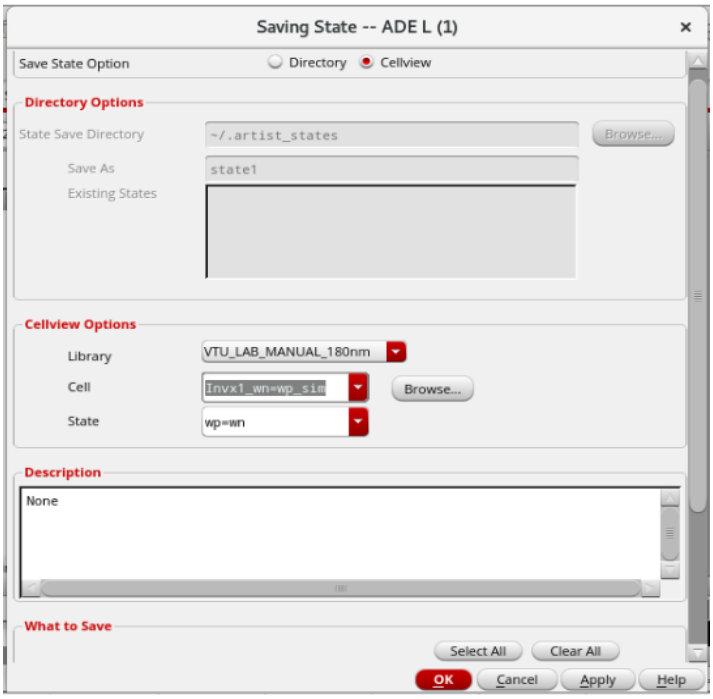



***Figure – 1.73: Saving State***

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

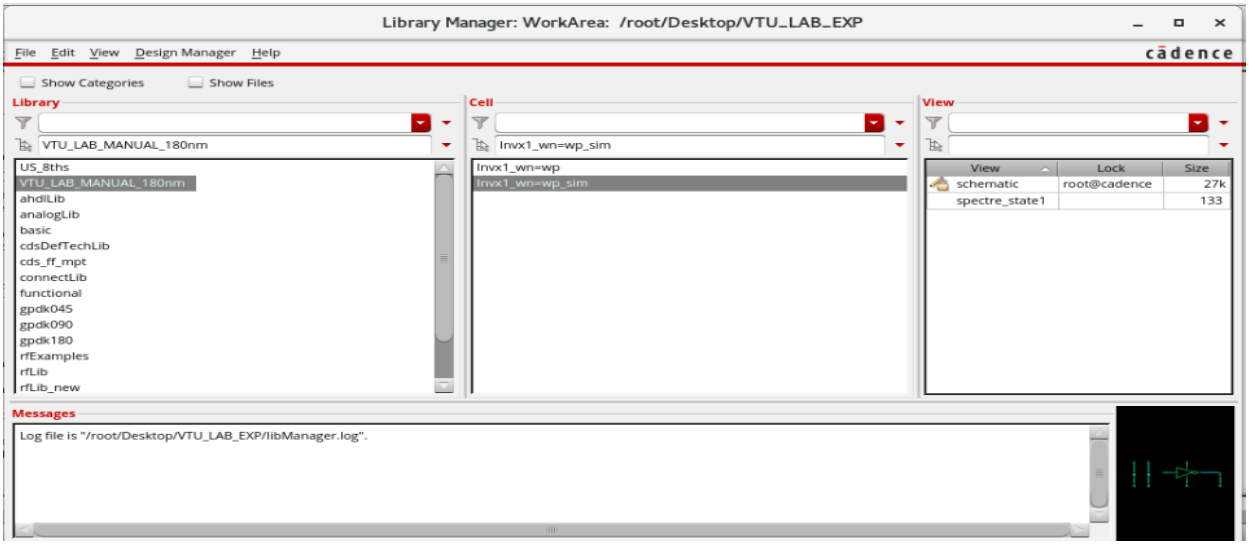



***Figure – 1.74: Test Schematic and State in Library Manager***

## OPEN THE SAVED ADE L STATE:

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

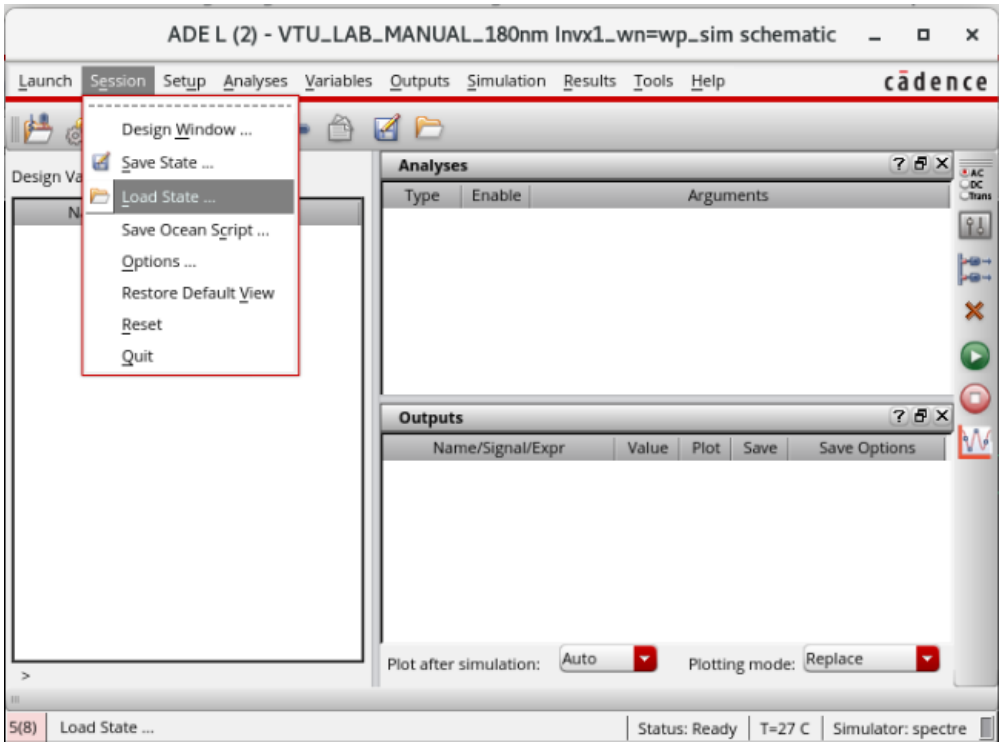



***Figure – 1.75: Session → Load State***

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

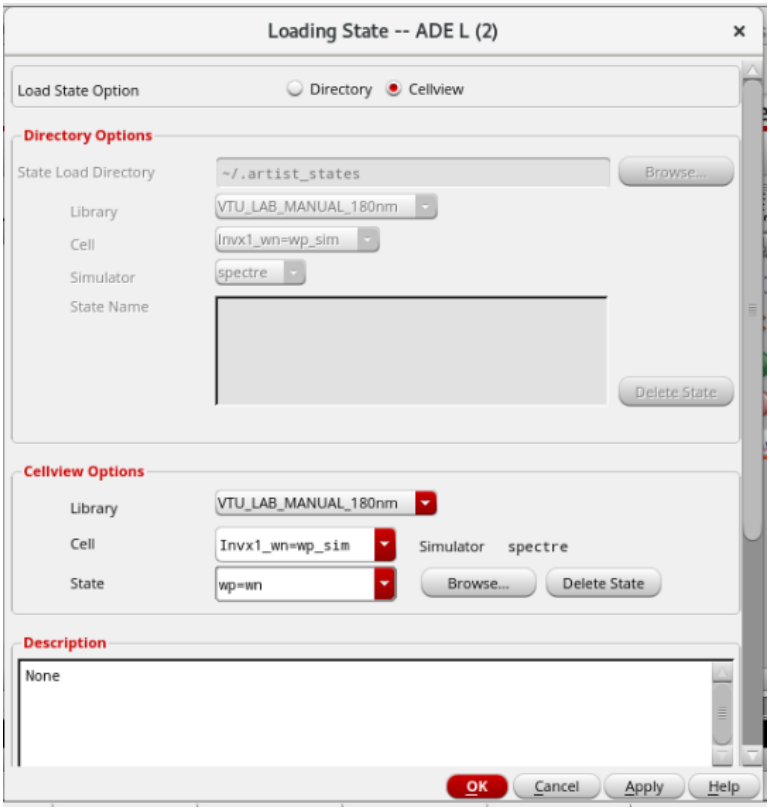



***Figure – 1.76: "Loading State – ADE L" window***

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

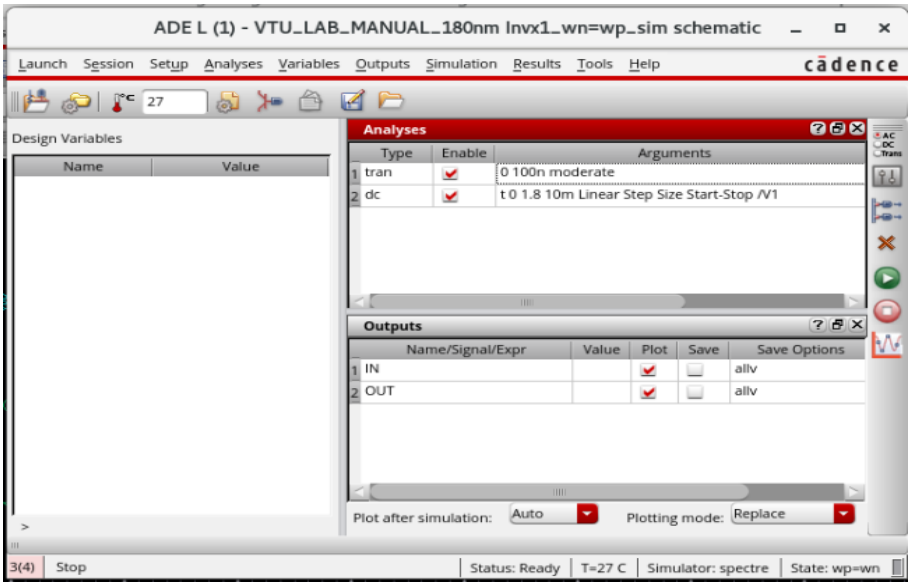



***Figure – 1.77: Reloaded ADE L State***

### (2) CALCULATION OF $tp_{HL}$, $tp_{LH}$ 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) Visualization and Analysis**" window. So, select "**Tools → Calculator**" or click on the icon as shown in Figure – 1.78.

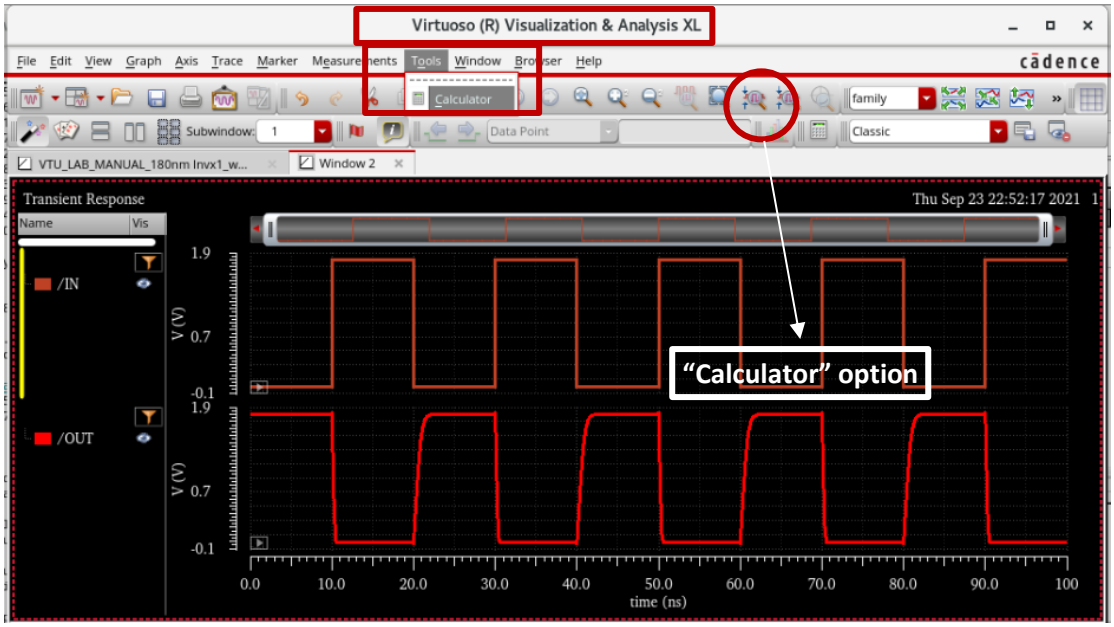



***Figure – 1.78: Tools → Calculator***

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

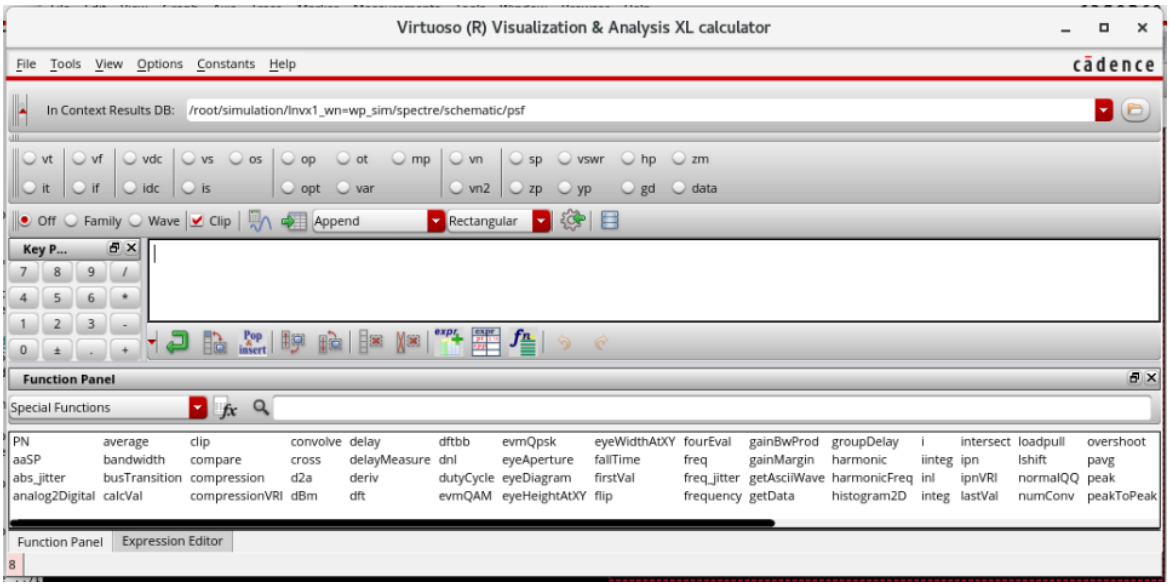



***Figure – 1.79: Virtuoso (R) Visualization and Analysis XL calculator window***

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

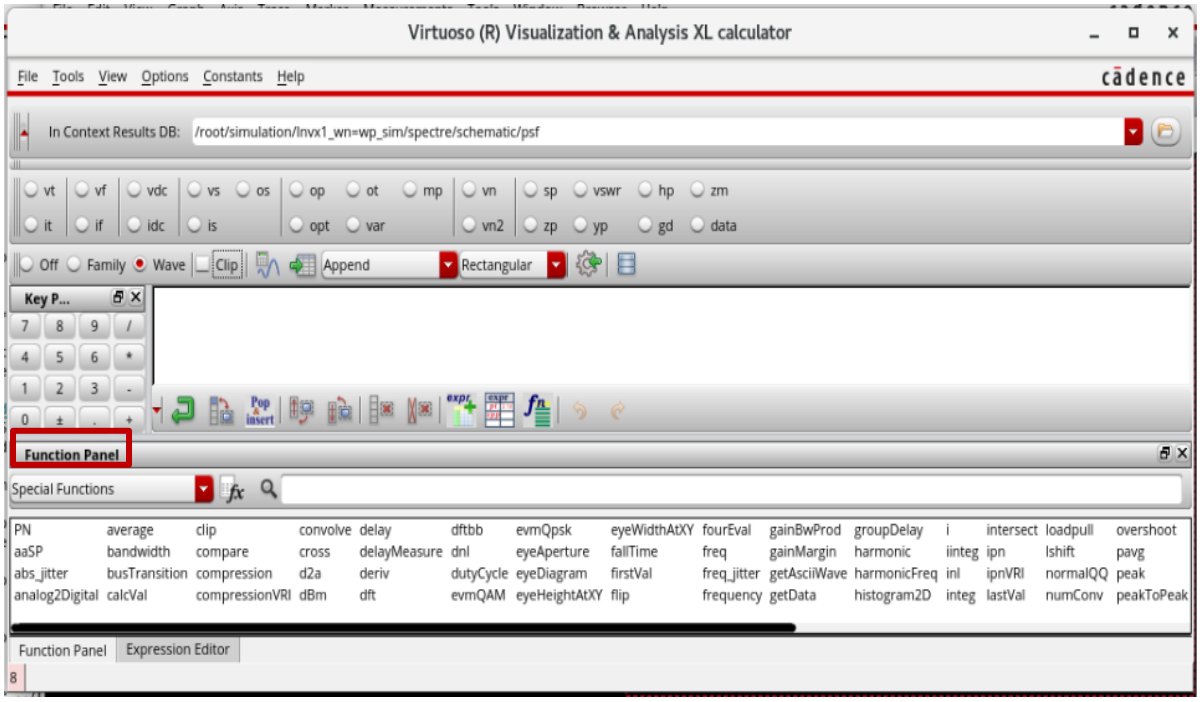



***Figure – 1.80: "Enable → Wave" and "Disable → Clip"***

Select "**delay**" from the "**Function Panel**" as shown in Figure – 1.80. The "**Function Panel**" gets updated as shown in Figure – 1.81.

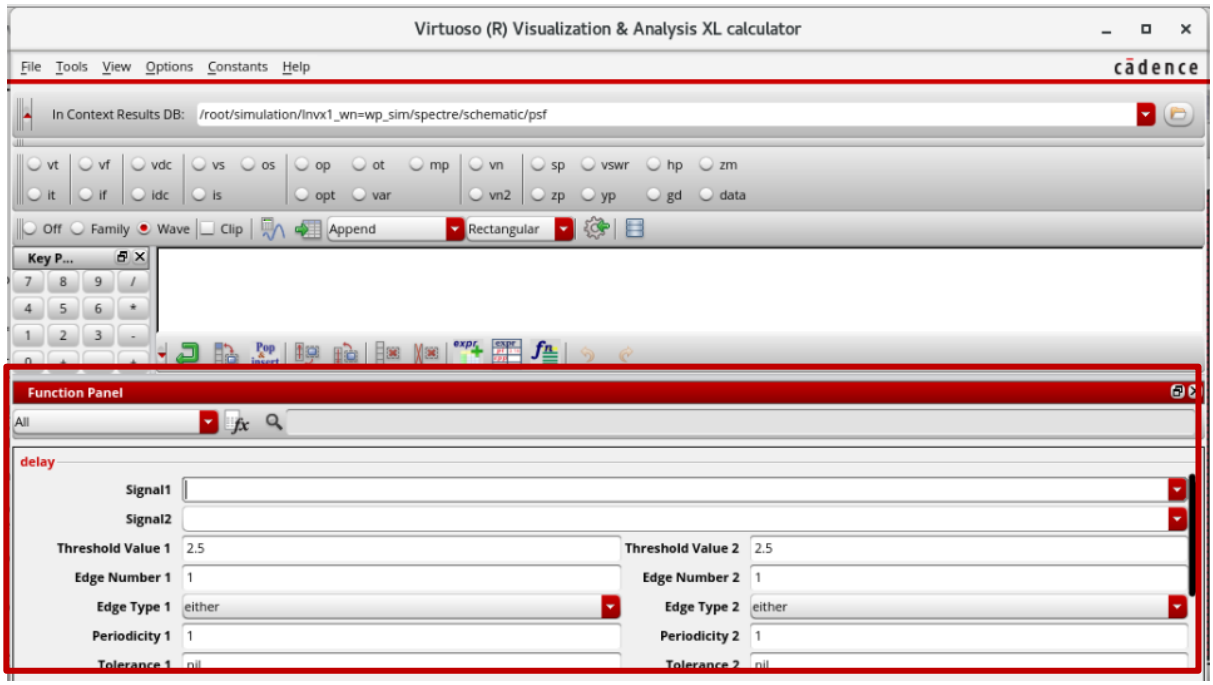



***Figure – 1.81: Updated Function Panel***

Place the cursor in "**Signal1**", select the signal "**IN**" from the waveform window as shown in Figure – 1.82.

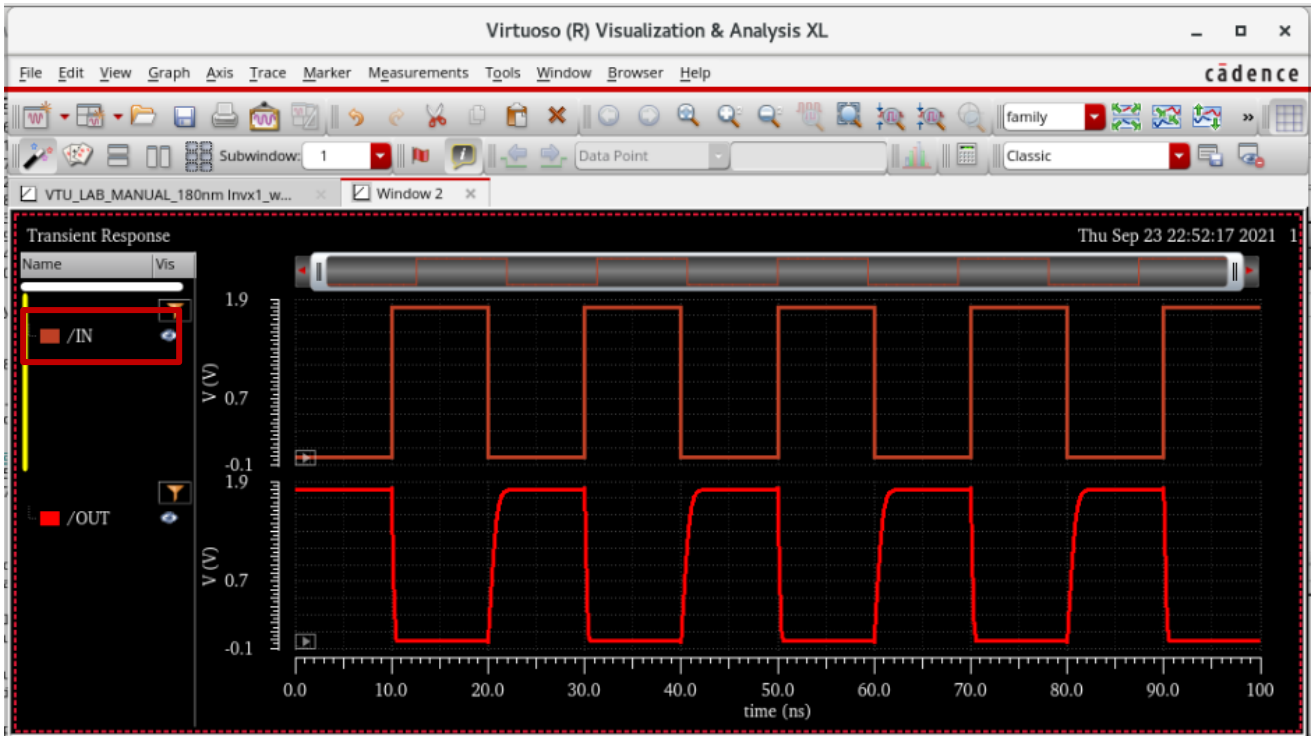



***Figure – 1.82: Selecting "IN" signal from the waveform***

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

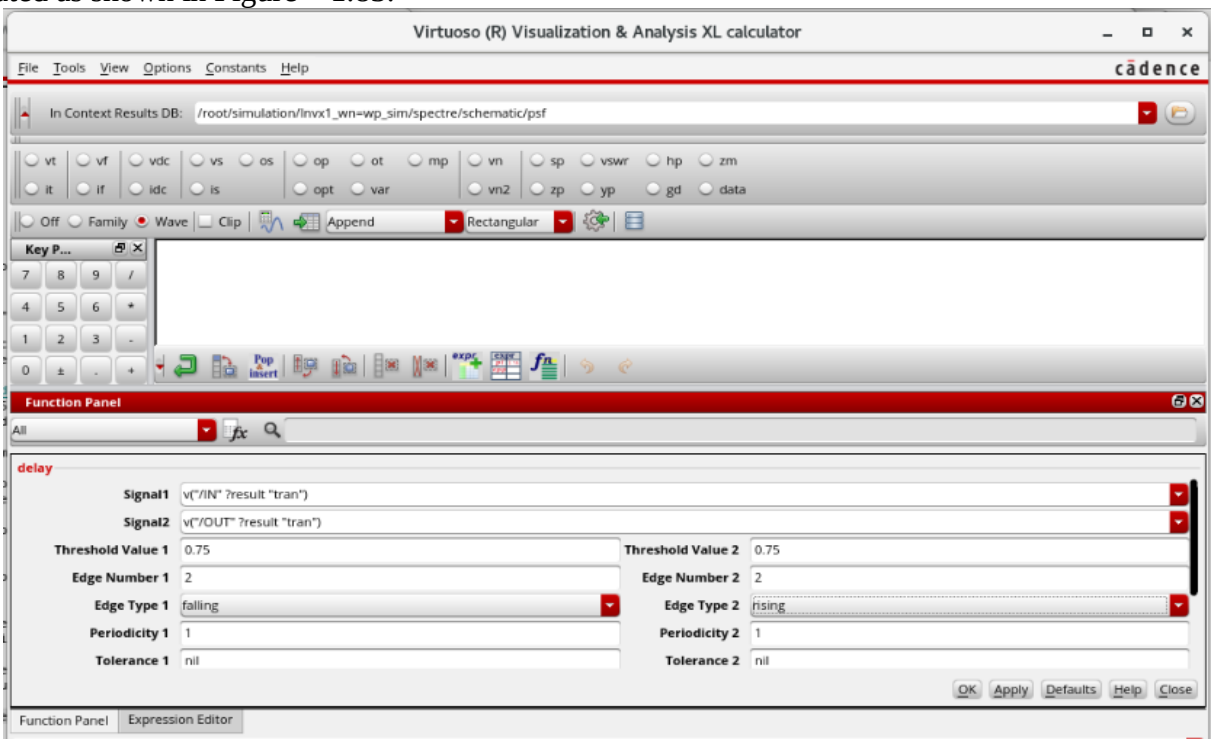



***Figure – 1.83: "Signal 2" and other parameter selection***

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 results in 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.

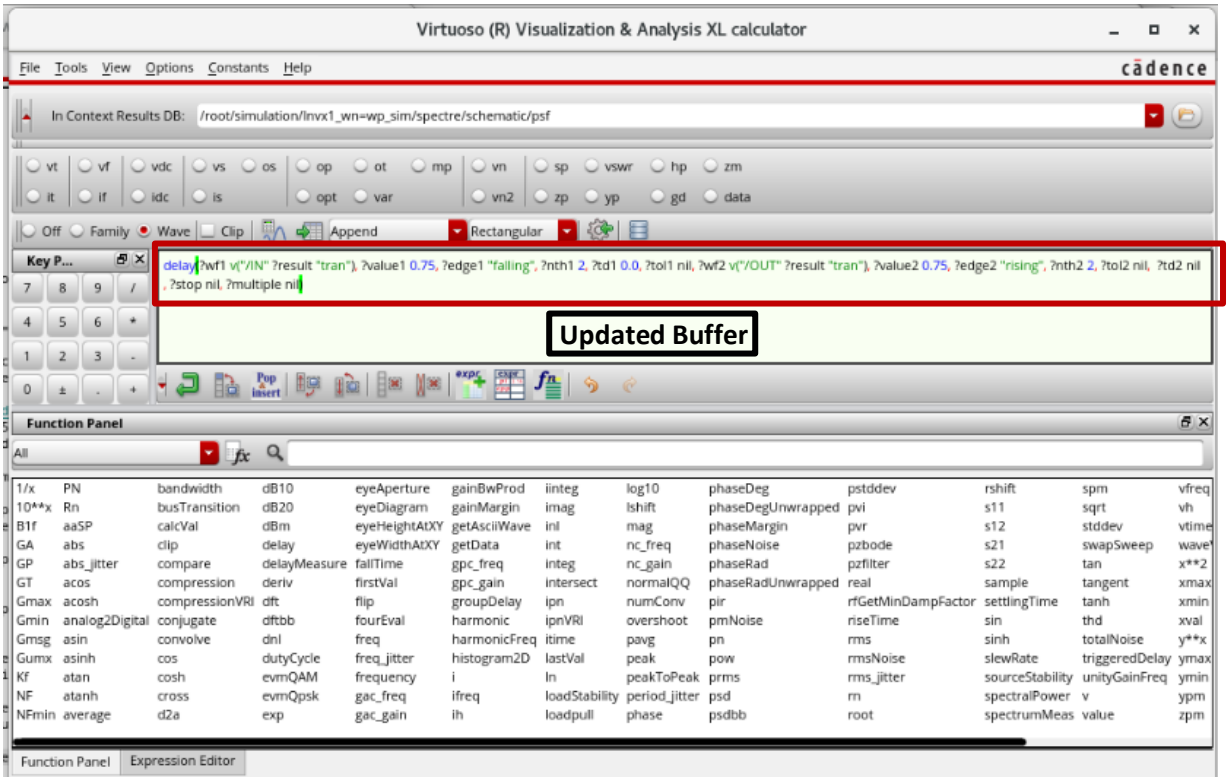



***Figure – 1.84: Updated Buffer***

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 $\frac{W_P}{W_N} = \frac{40}{20}$

**Objective:**

To draw the Layout of CMOS Inverter with $\frac{W_P}{W_N} = \frac{40}{20}$ 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<br>Length, L = 180 n |
| gpdk180 | Pmos | Width, $W_P$ = 40 u<br>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.

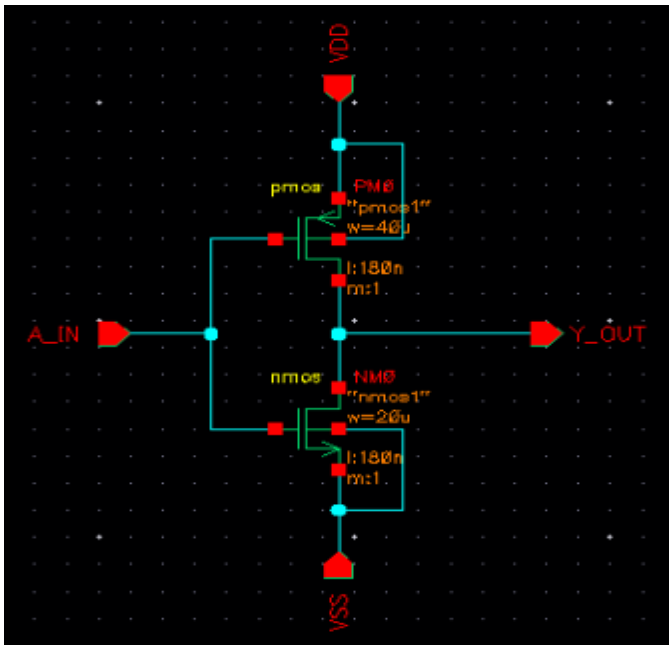



***Figure – 1.85: Schematic for CMOS Inverter with*** $\frac{W_P}{W_N} = \frac{40}{20}$

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

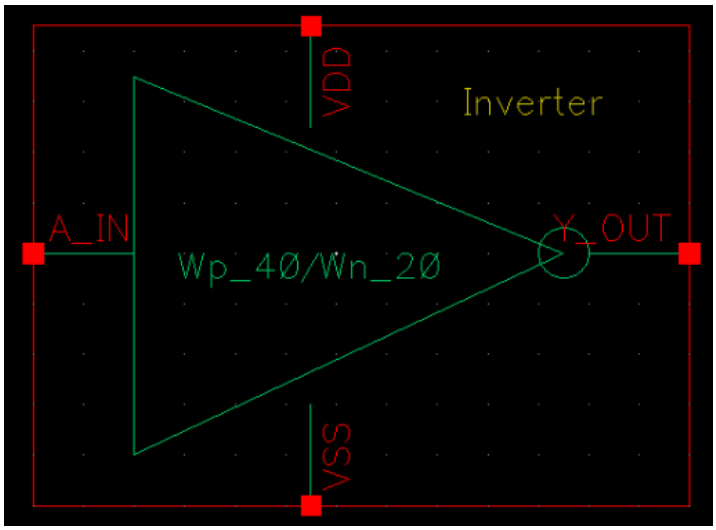



***Figure – 1.86: Symbol for CMOS Inverter with*** $\frac{W_P}{W_N} = \frac{40}{20}$

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.

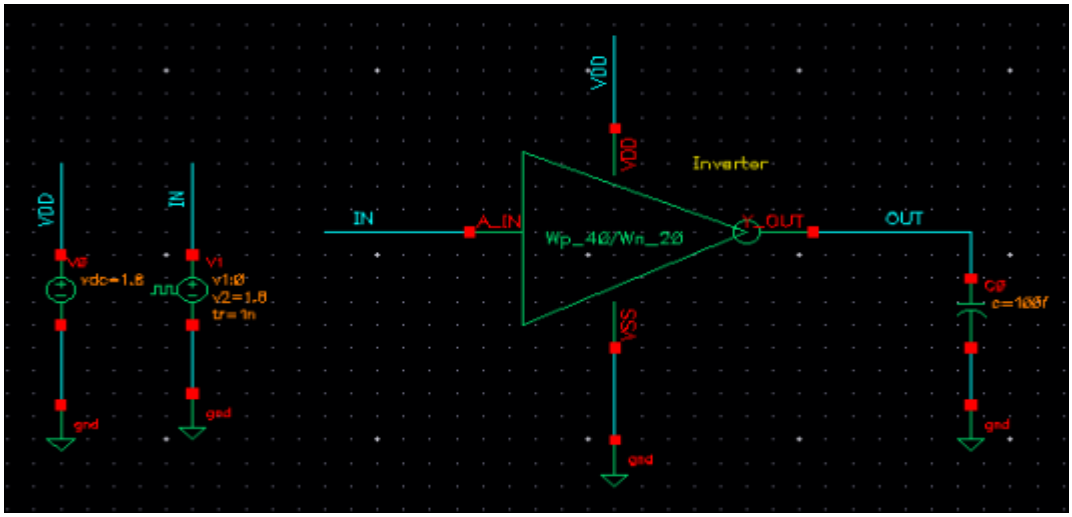



***Figure – 1.87: Test Schematicfor CMOS Inverter with*** $\frac{W_P}{W_N} = \frac{40}{20}$

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.

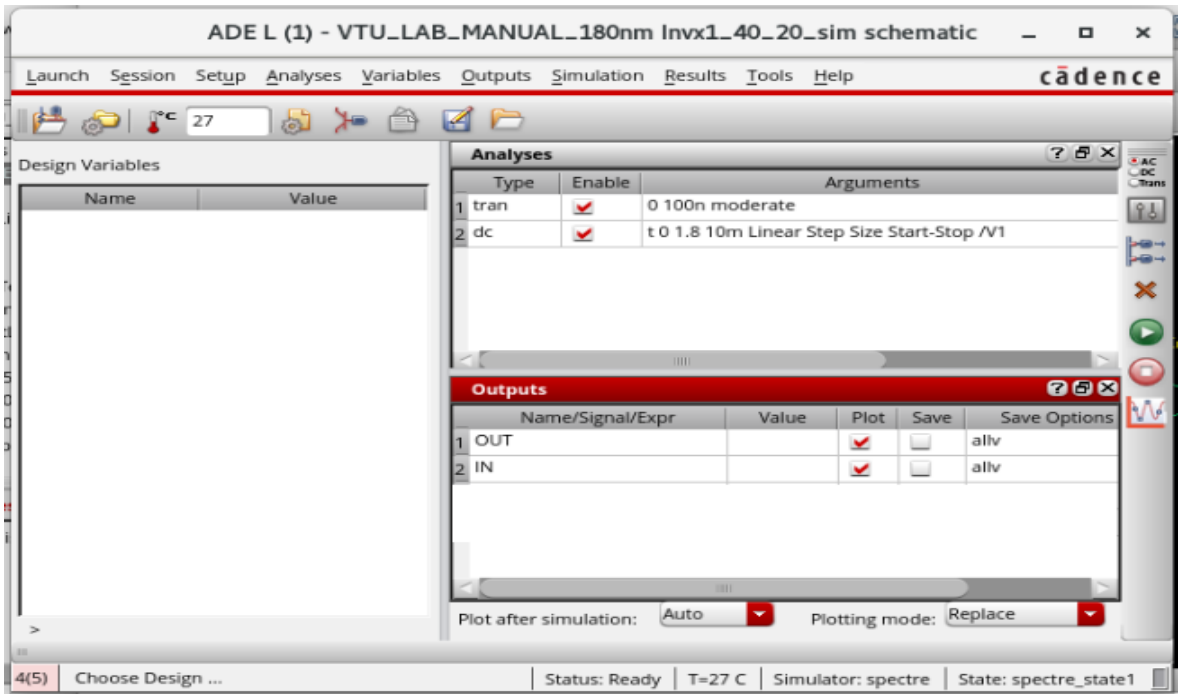



***Figure – 1.88: Updated ADE L window***

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

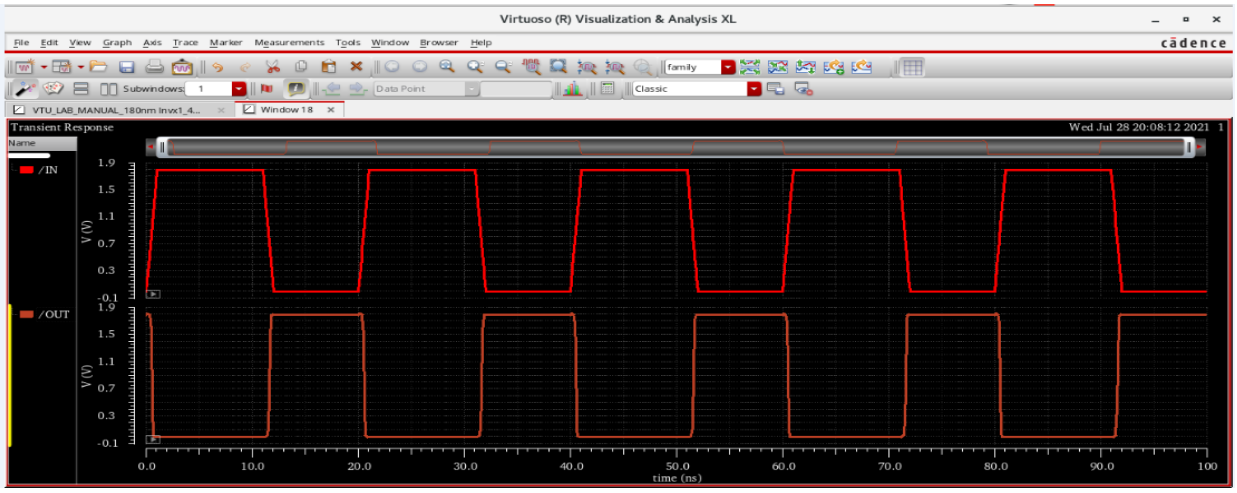



*Figure – 1.89: Transient Analysis for CMOS Inverter with* $\frac{W_P}{W_N} = \frac{40}{20}$

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

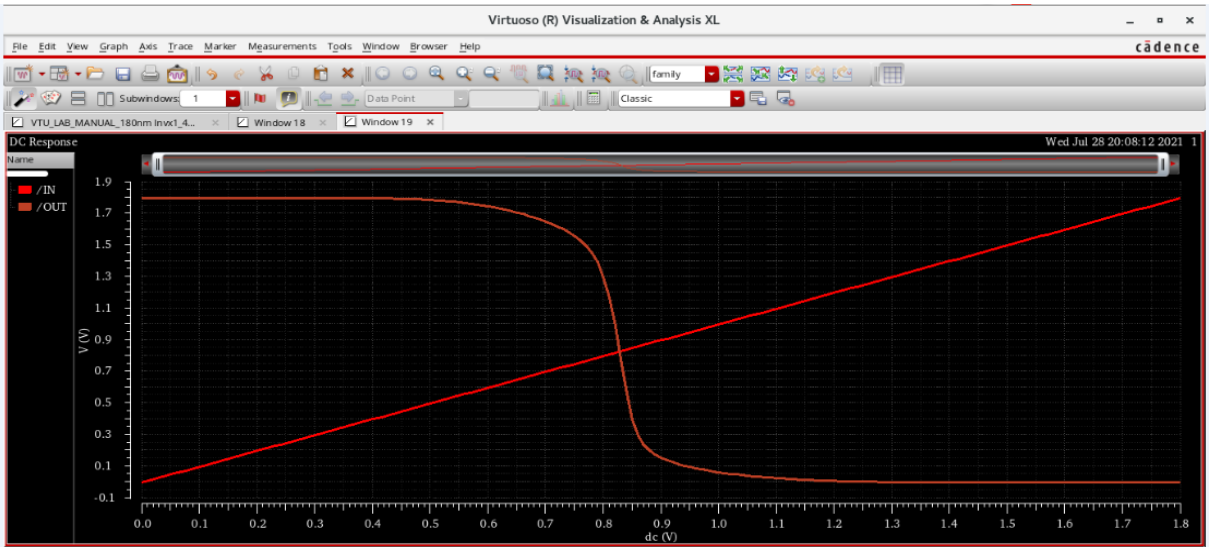



*Figure – 1.90: DC Analysis for CMOS Inverter with* $\frac{W_P}{W_N} = \frac{40}{20}$

**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* $\frac{W_P}{W_N} = \frac{40}{20}$

| 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.

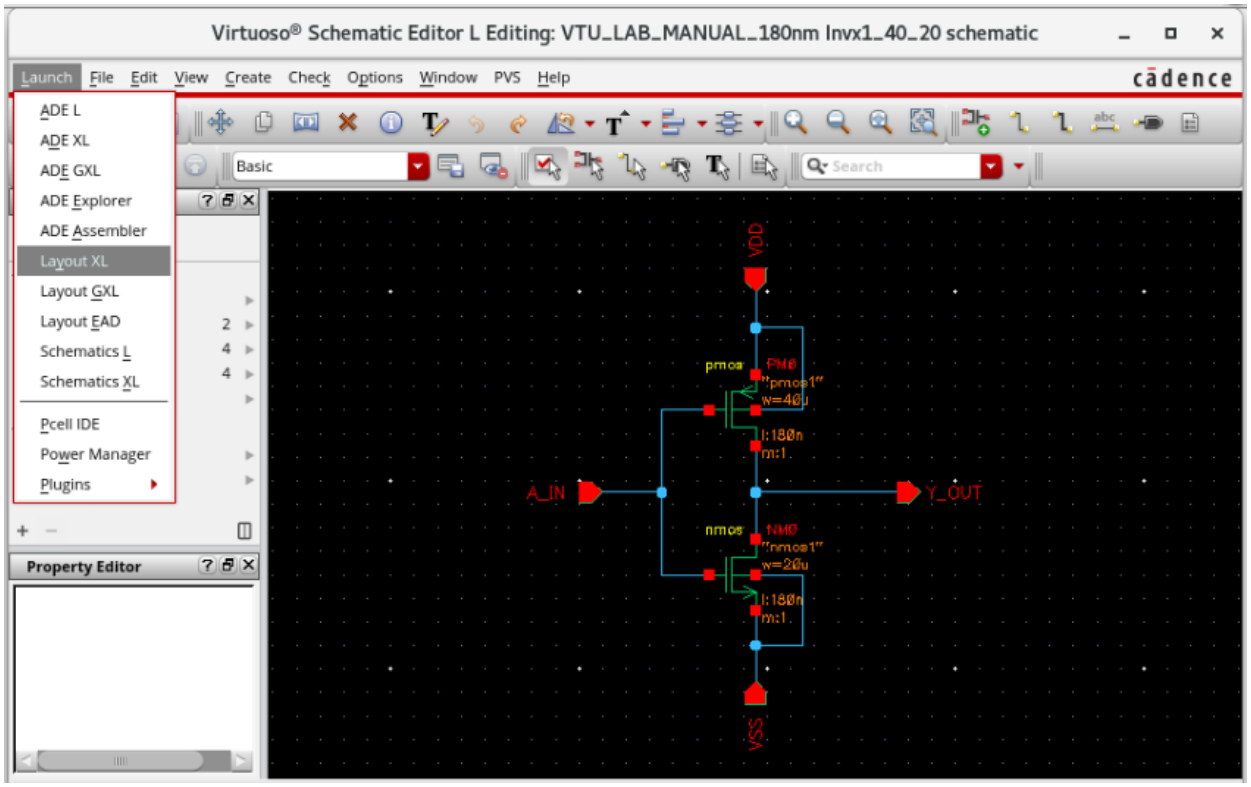



*Figure – 1.91: Launch → Layout XL*

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.

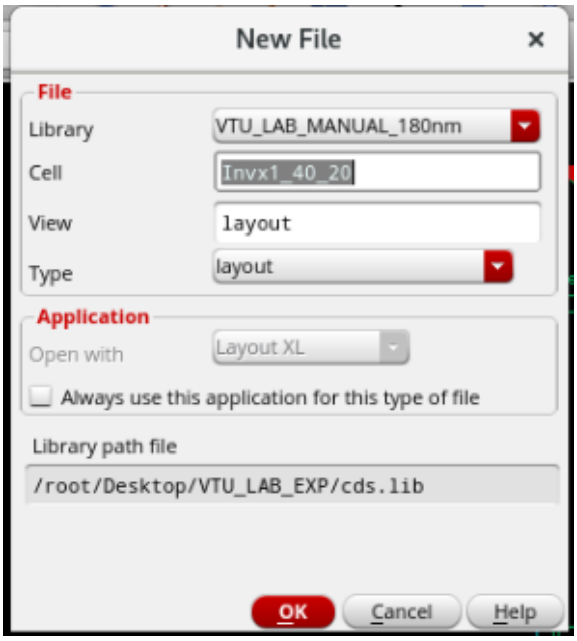



***Figure – 1.93: New File window***

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.

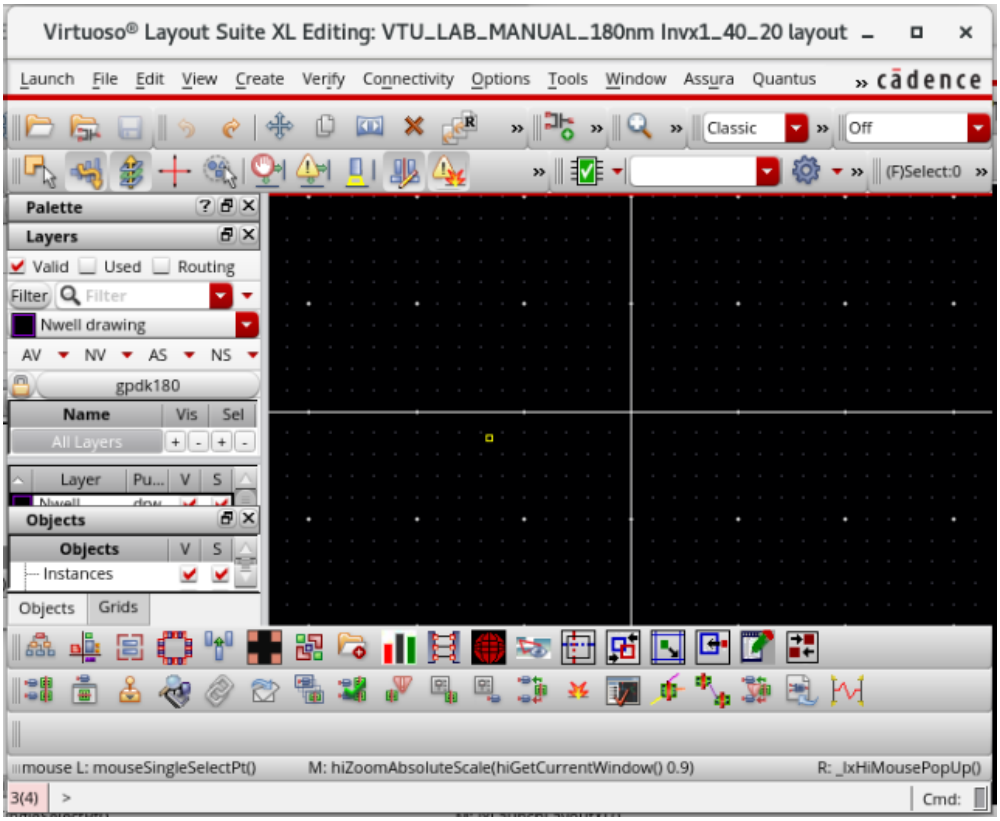



***Figure – 1.94: Virtuoso Layout Suite XL Editing***

***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.

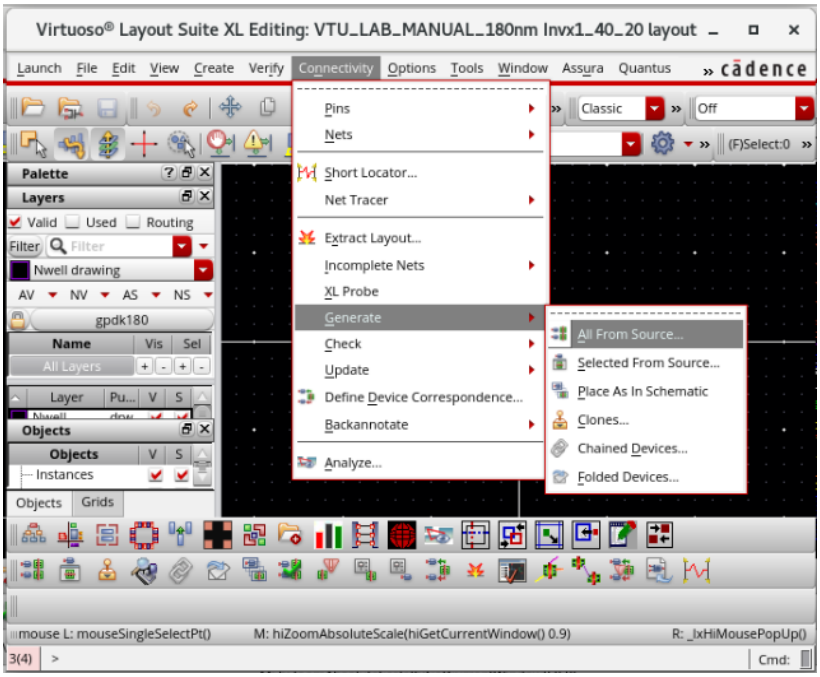



***Figure – 1.95: Connectivity → Generate → All From Source***

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.

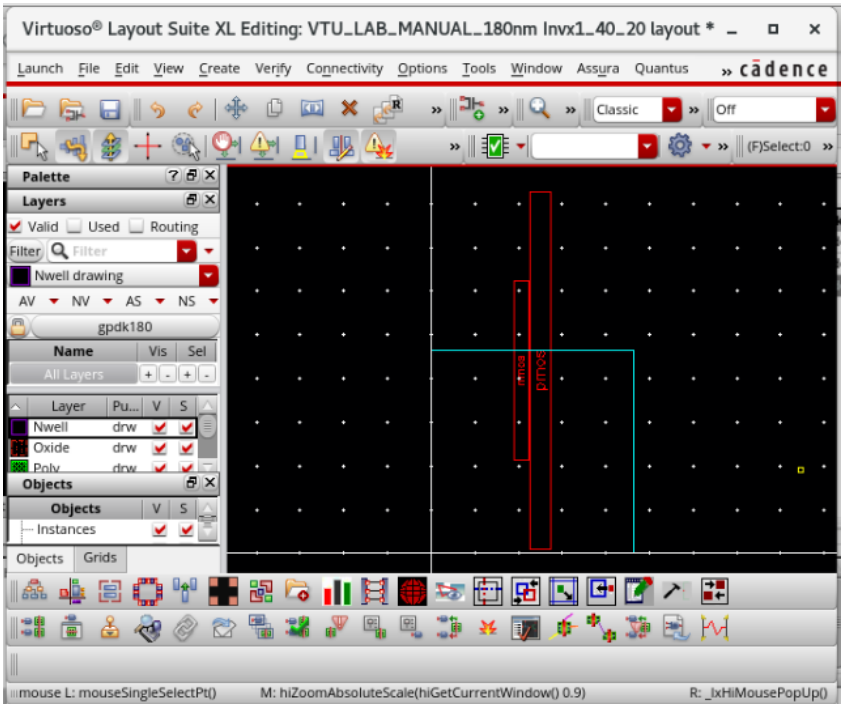



***Figure – 1.97: Updated Virtuoso Layout Editor***

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.

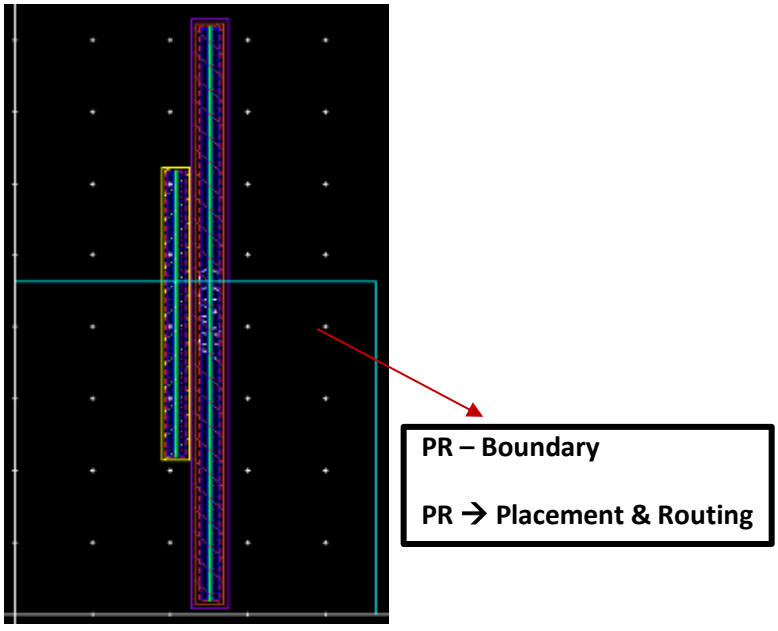



***Figure – 1.98: Updated devices after "Shift + F"***

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.

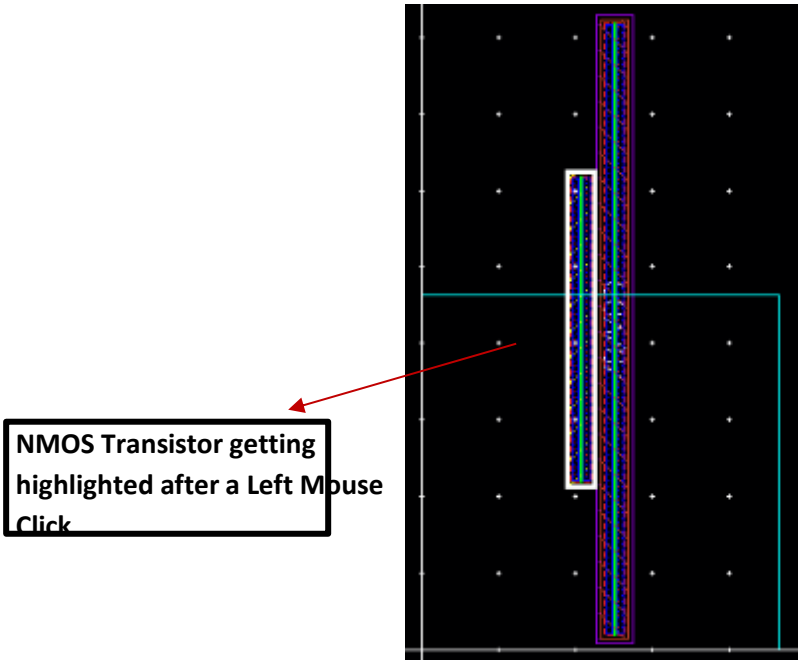



***Figure – 1.99: NMOS Transistor after Left Mouse Click***

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**".

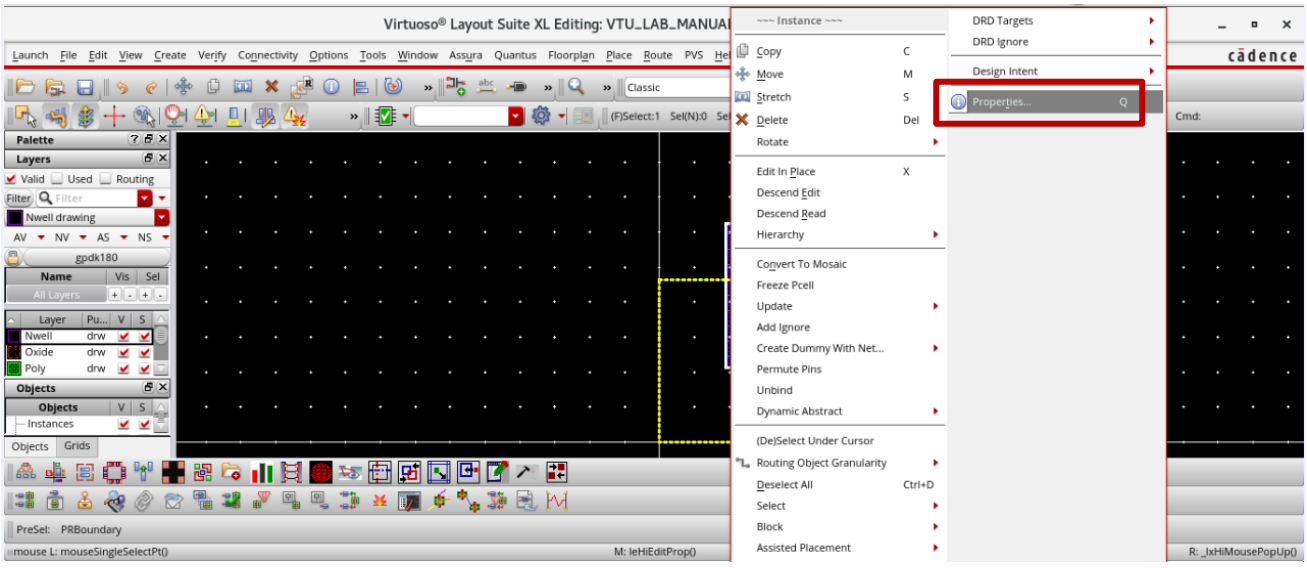



***Figure – 1.100: Select "Properties" after a Right Mouse Click***

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

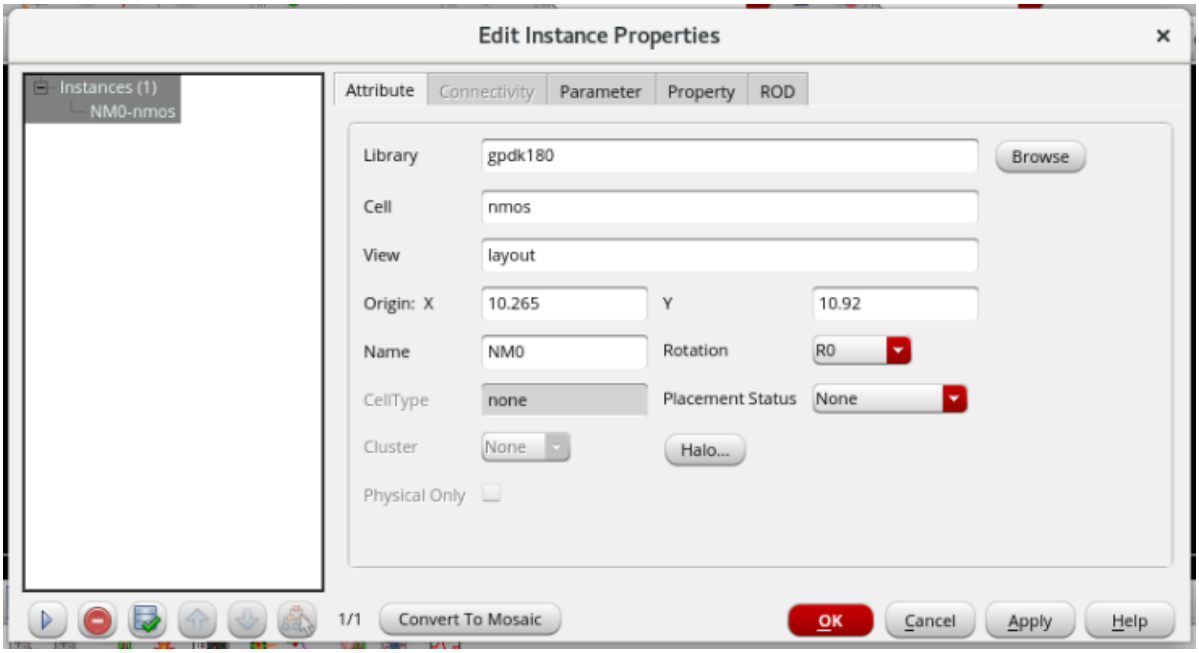



***Figure – 1.101: Edit Instance Properties window***

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.

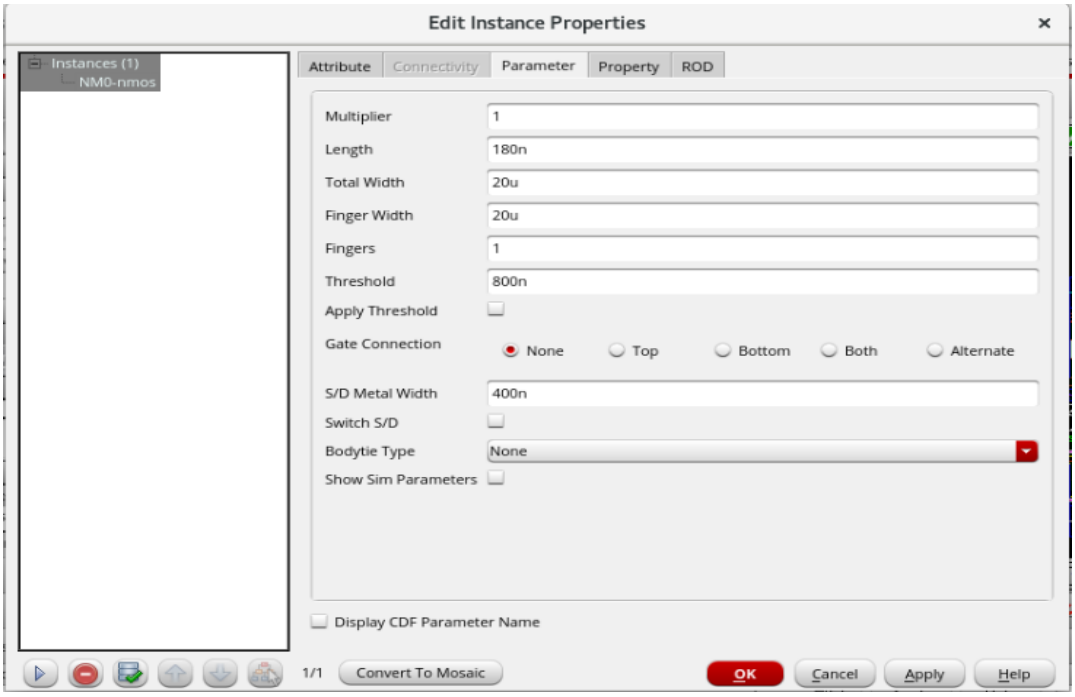



***Figure – 1.102: Parameter tab before update***

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

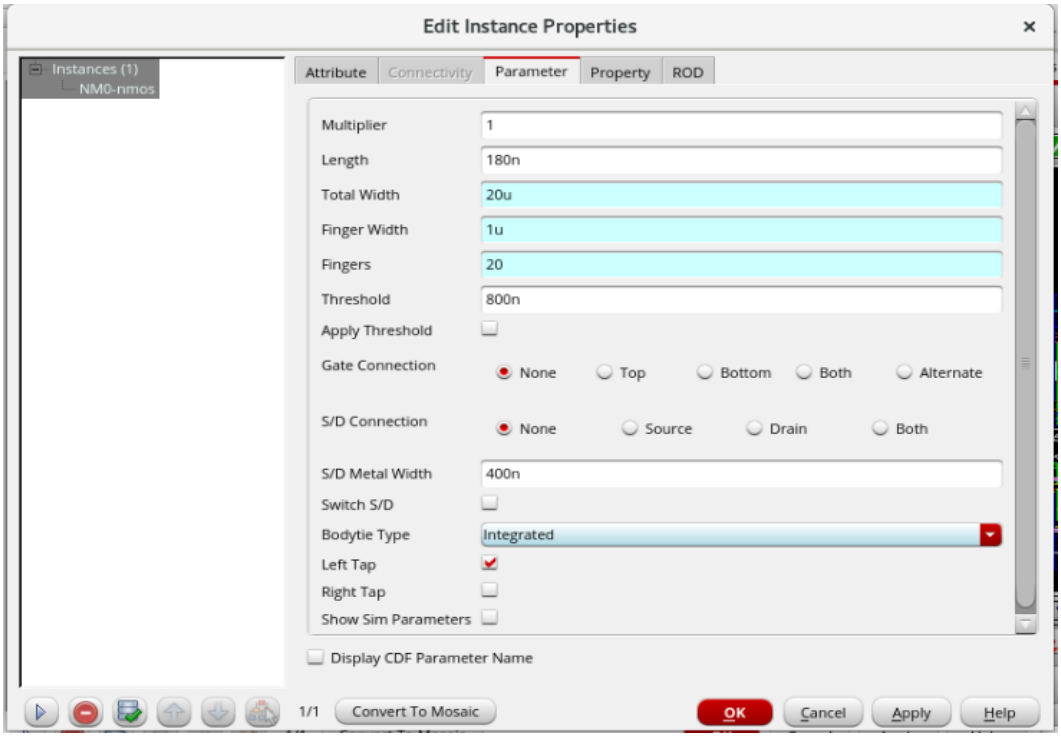



***Figure – 1.103: Parameter tab after updating the values for NMOS transistor***

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.

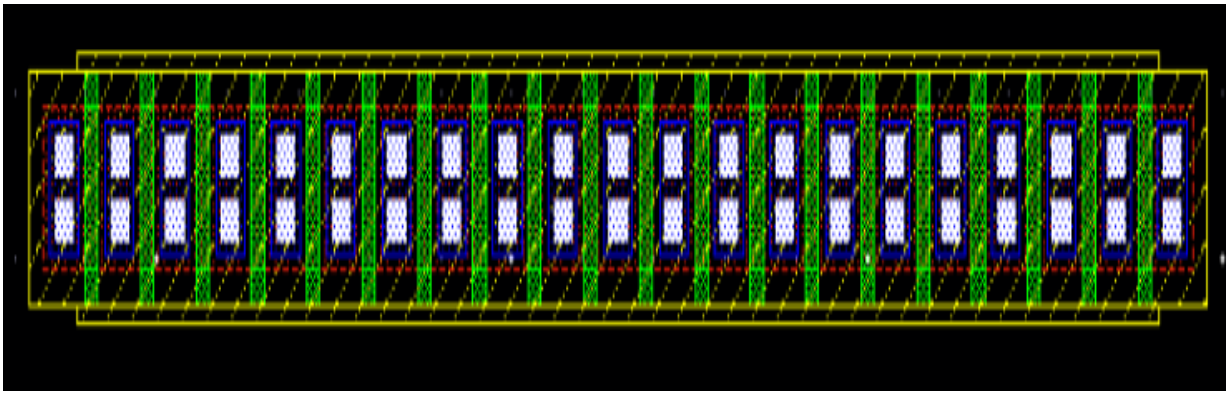


***Figure – 1.104: Updated NMOS Transistor***

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.

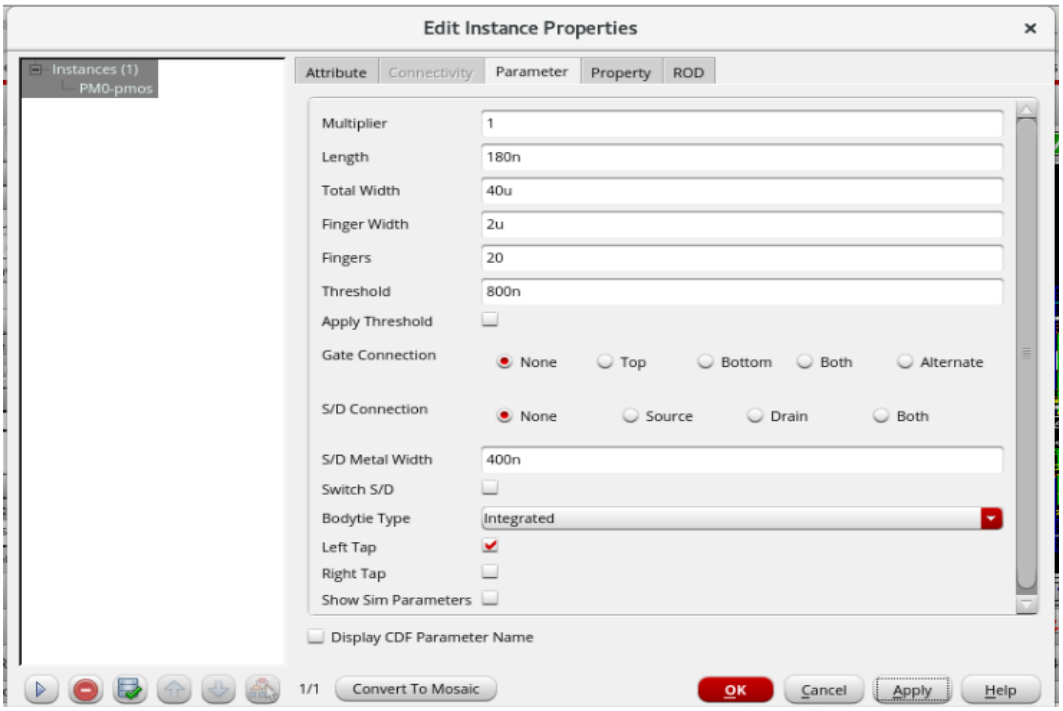



***Figure – 1.105: Parameter tab after updating the values for PMOS transistor***

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

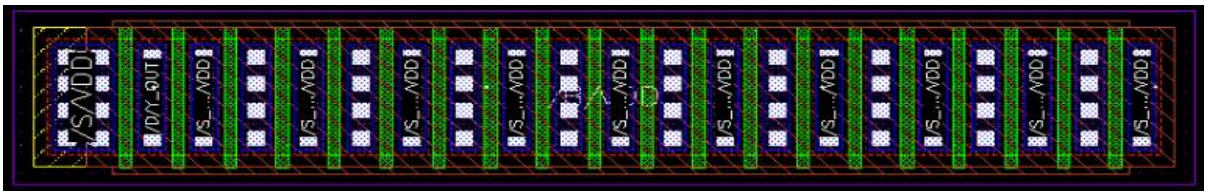


***Figure – 1.106: Updated PMOS Transistor***

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

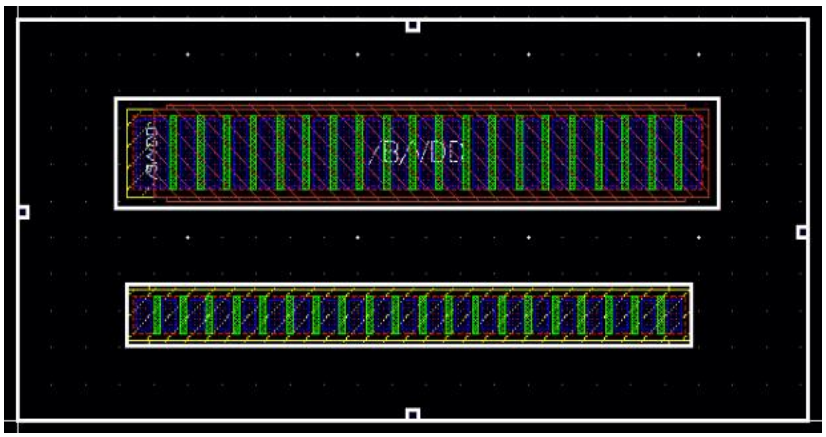


***Figure – 1.107: Selecting the layout using "Ctrl + A"***

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

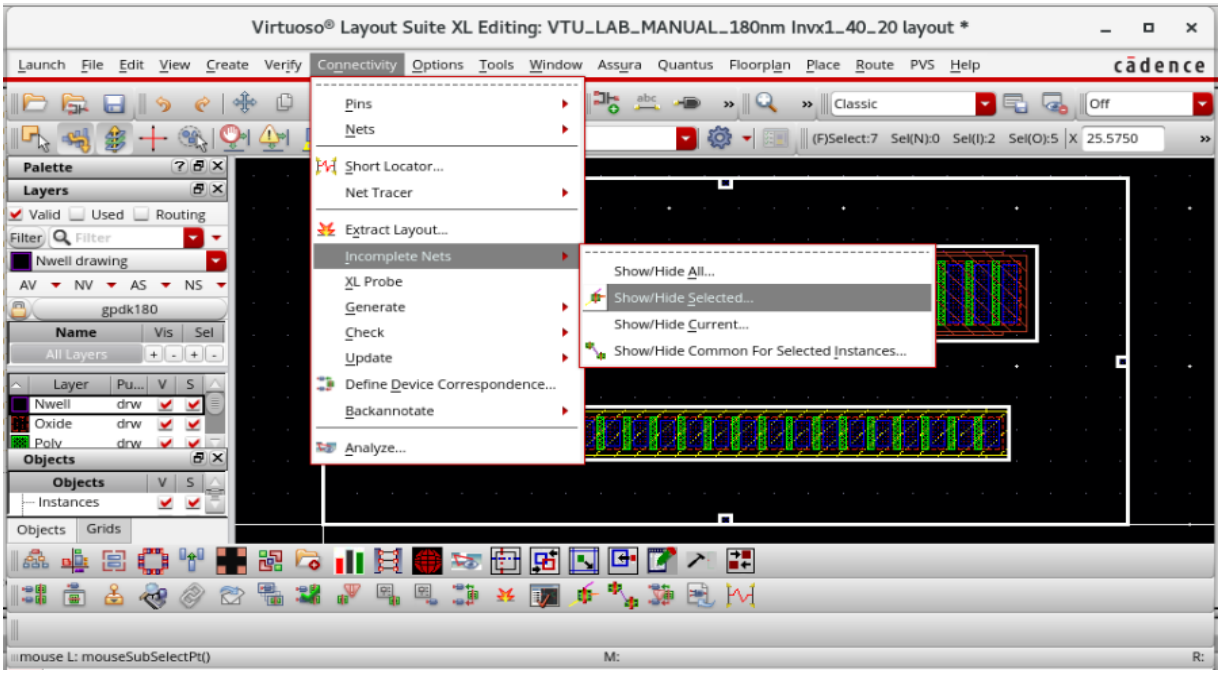



***Figure – 1.108: Connectivity → Incomplete Nets → Show/Hide Selected..***

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.

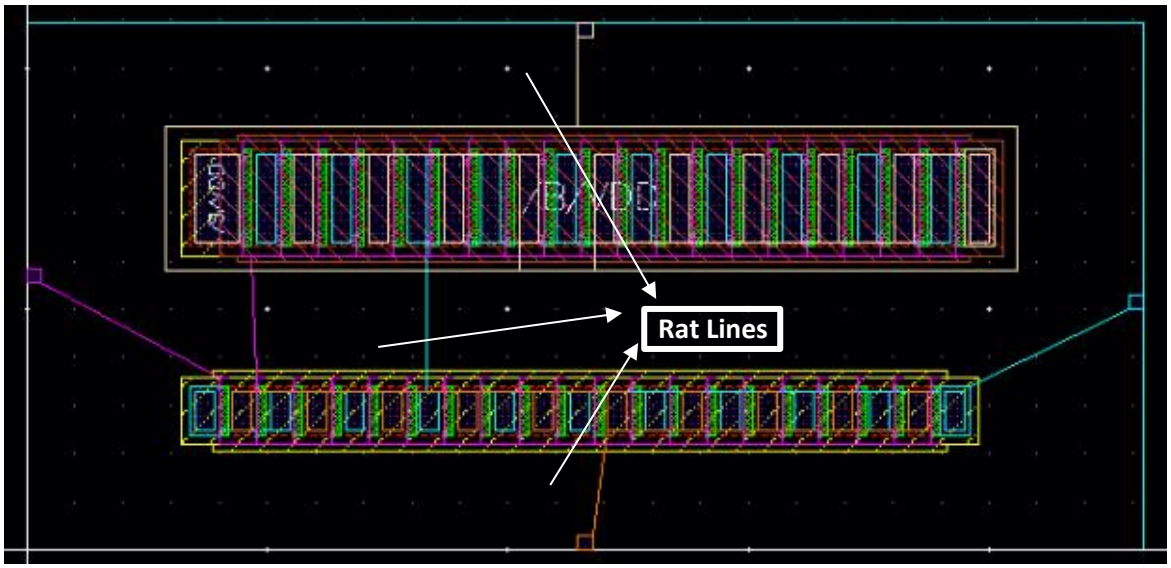



***Figure – 1.109: Layout with Rat Lines***

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.

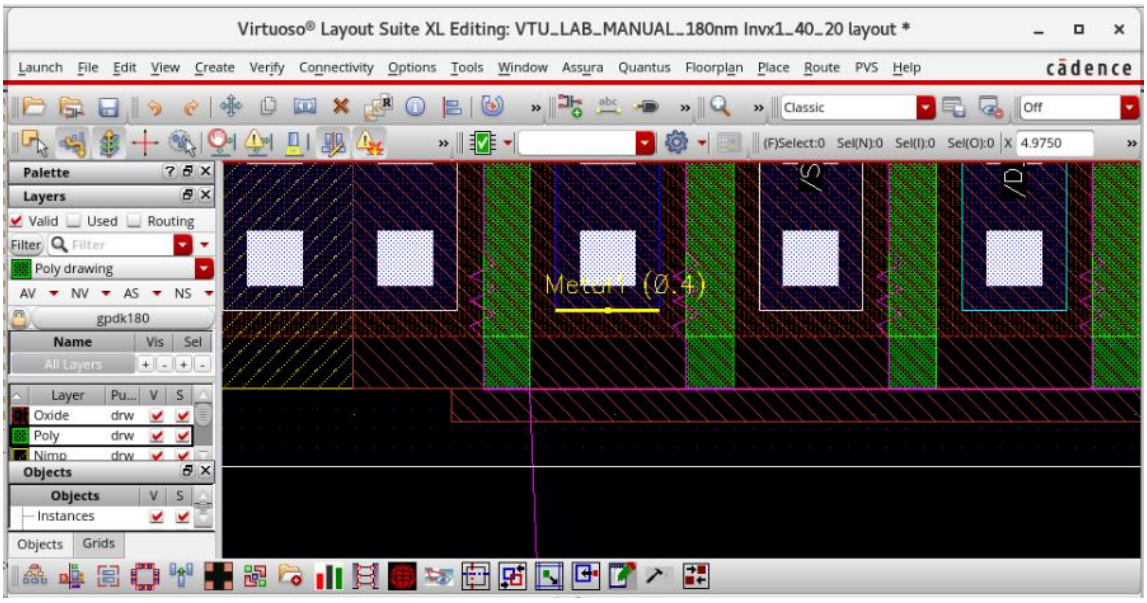



*Figure – 1.110: Terminal getting highlighted*

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.

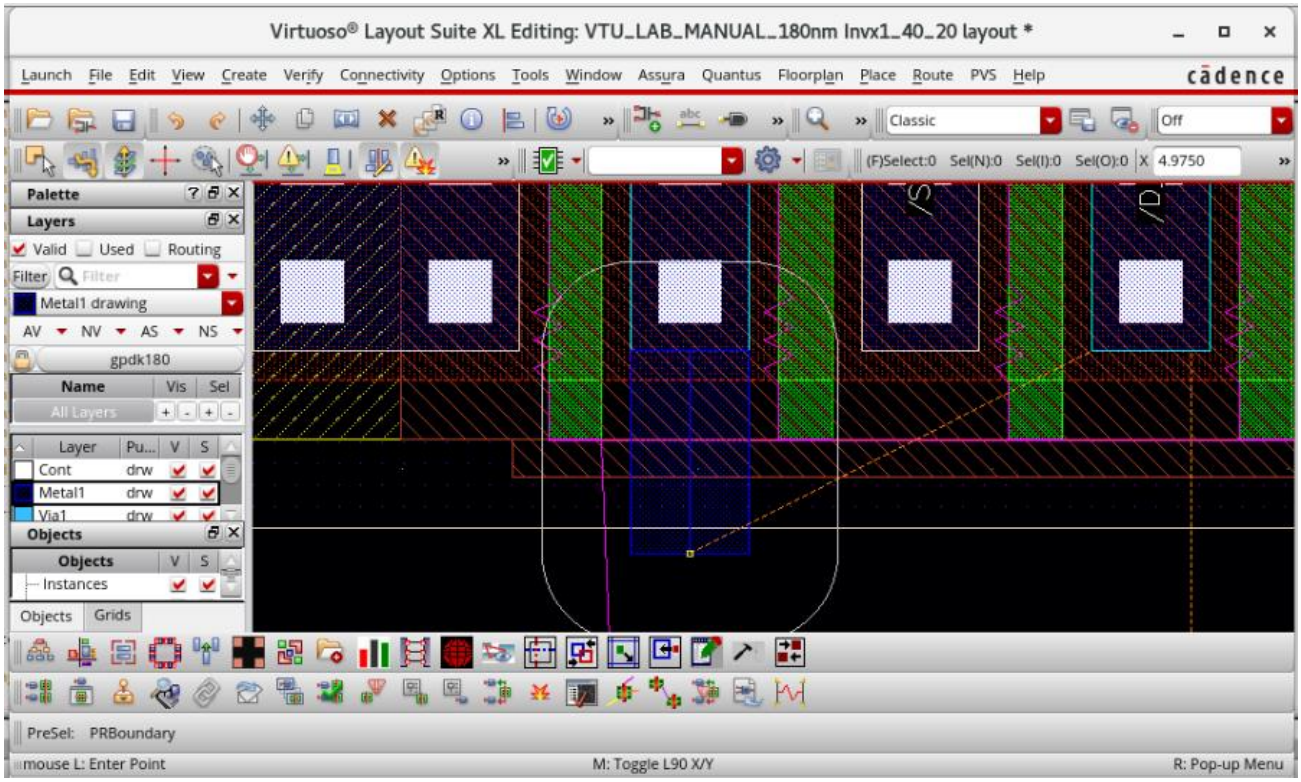



*Figure – 1.111: Start of Interconnection after Left Mouse Click*

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

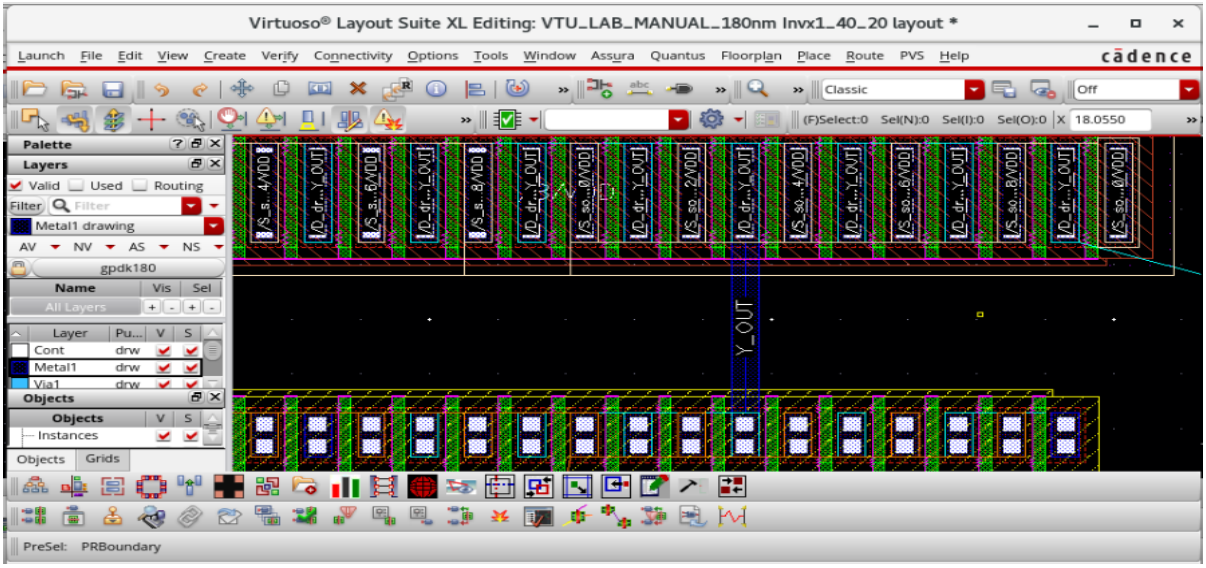



***Figure – 1.112: Layout after the interconnections***

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.

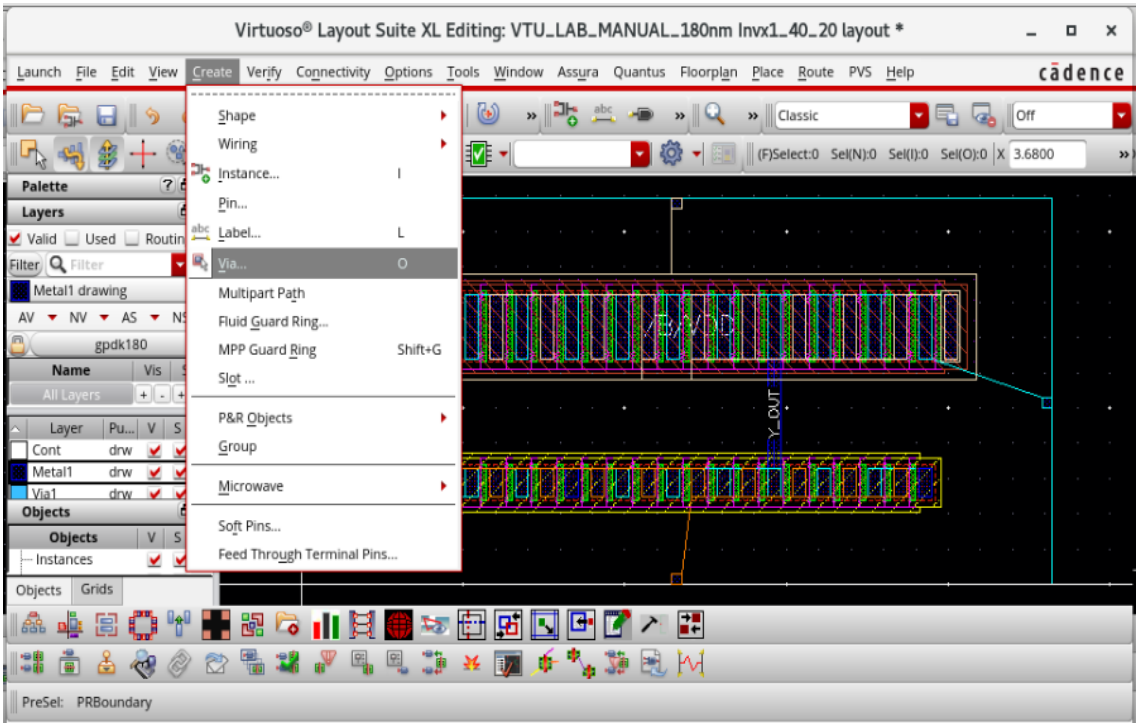



***Figure – 1.113: Create → Via***

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


***Figure – 1.114: Create Via window***

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.

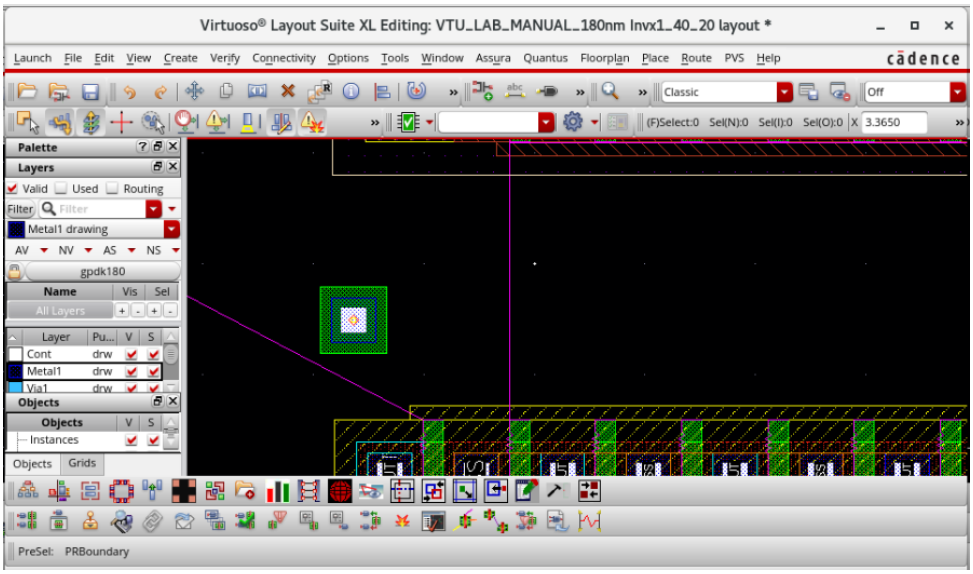



***Figure – 1.115: Selected Via***

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).

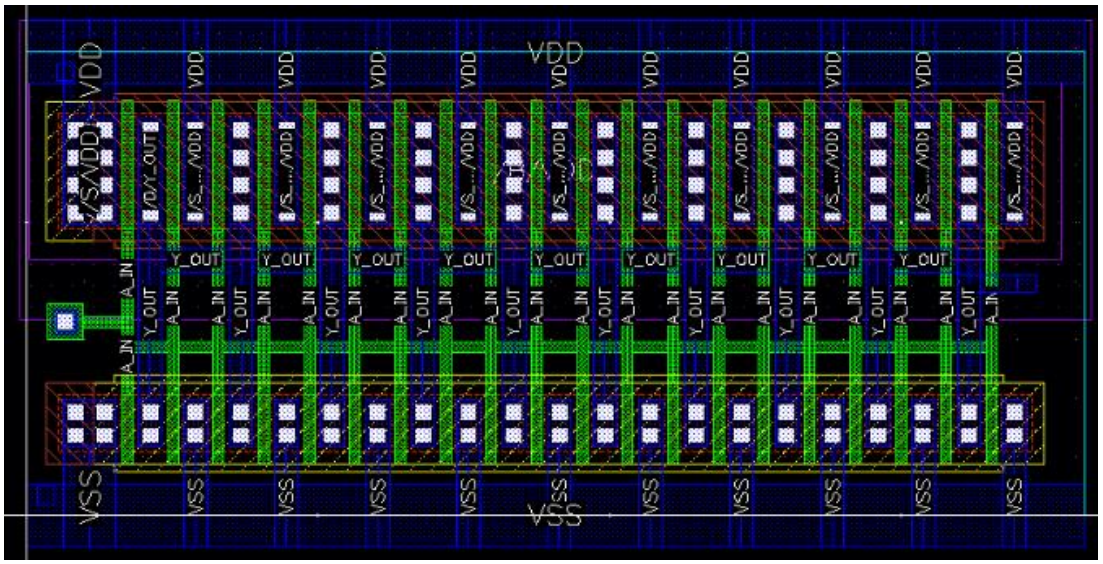



***Figure – 1.116(a): Completed Layout***

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

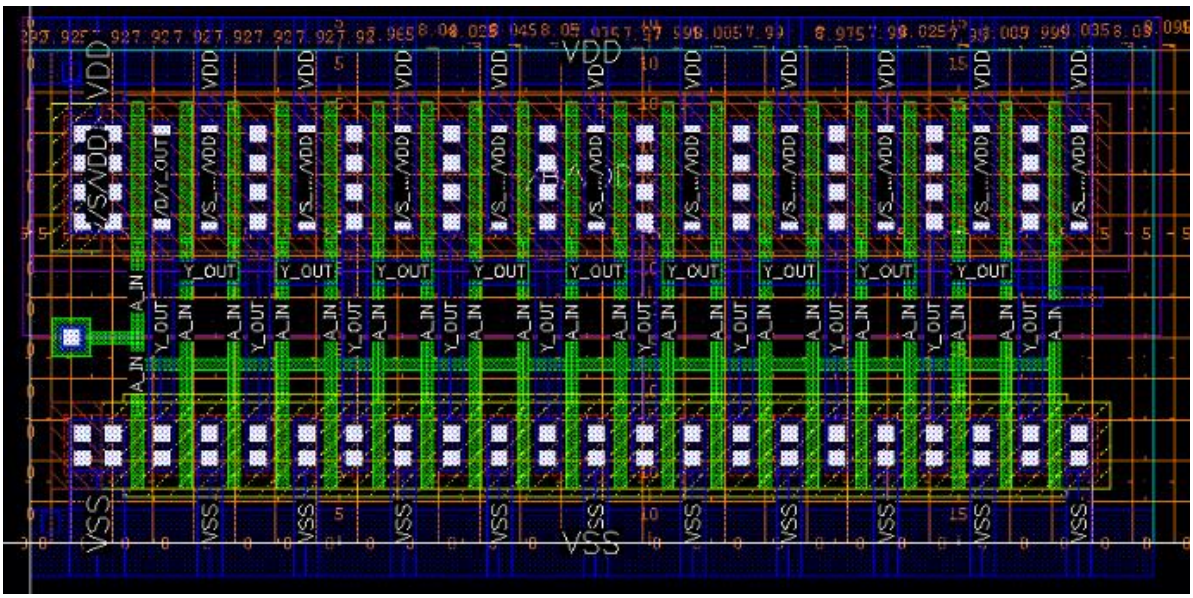



***Figure – 1.116(b): Completed Layout with Template***

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.

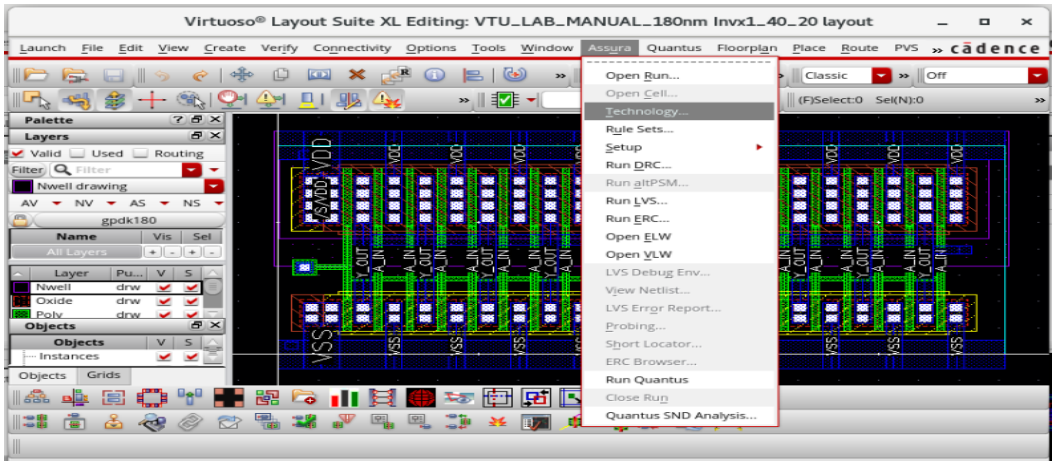



***Figure – 1.117: Assura → Technology..***

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

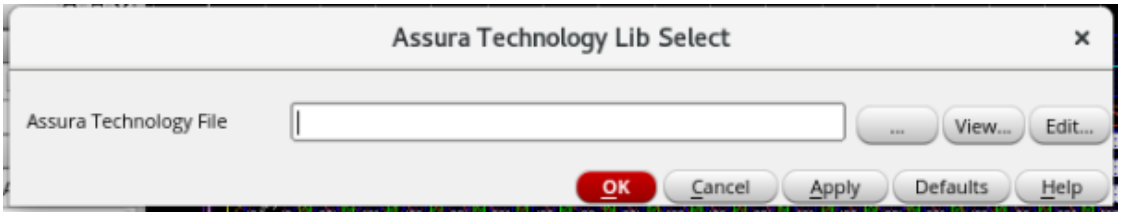



***Figure – 1.118: Assura Technology Lib Select***

Click on "**Browse**" option as shown in Figure – 1.118. The "**File Selector**" window pops up as shown in Figure – 1.119.

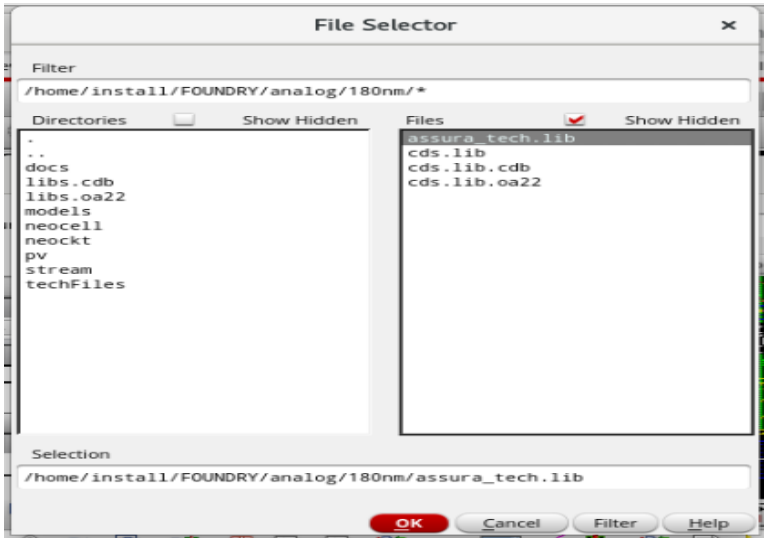



***Figure – 1.119: File Selector window***

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**".

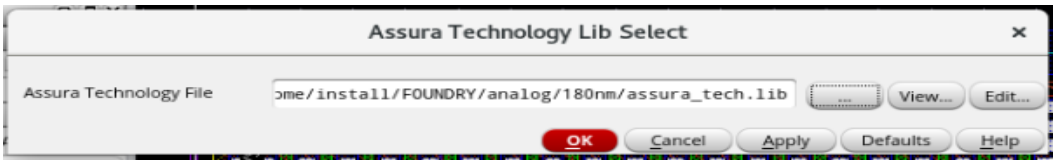



***Figure – 1.120: Assura Technology Lib Select window after file selection***

**DRC (DESIGN RULE CHECK):**

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

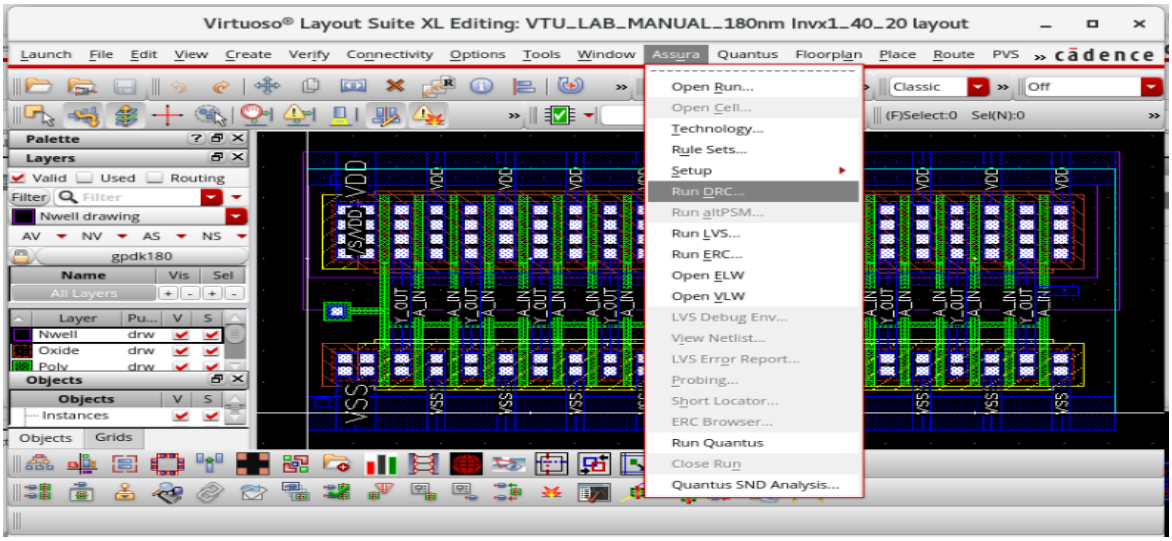



***Figure – 1.121: Assura → Run DRC***

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

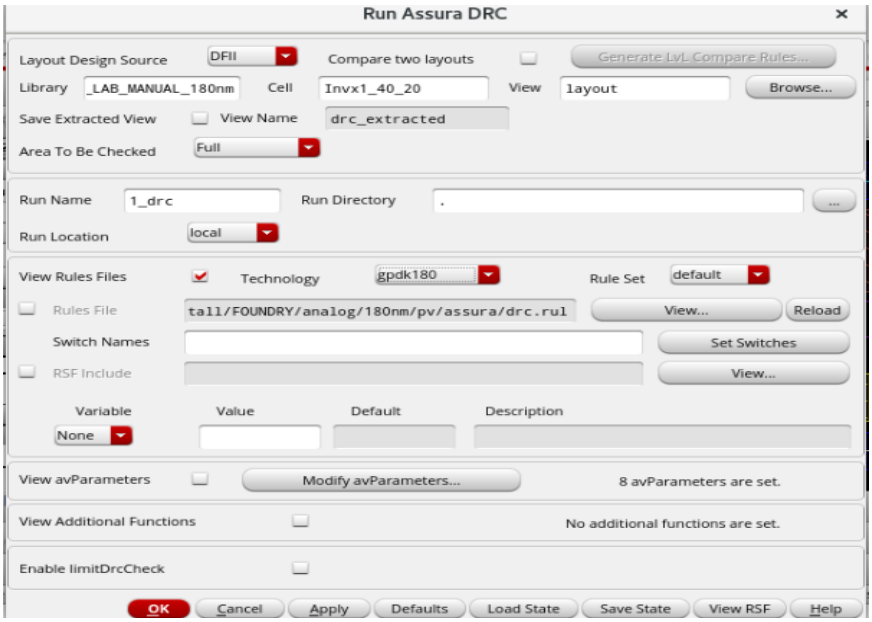



***Figure – 1.122: Run Assura DRC window***

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.

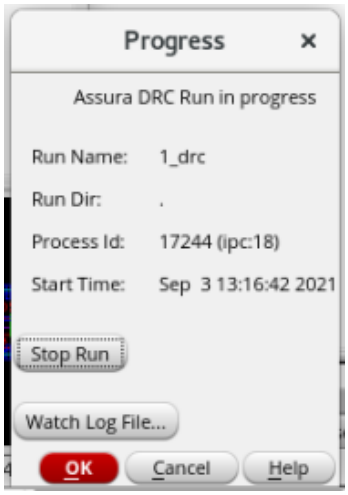



***Figure – 1.123: "Progress" window***

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.

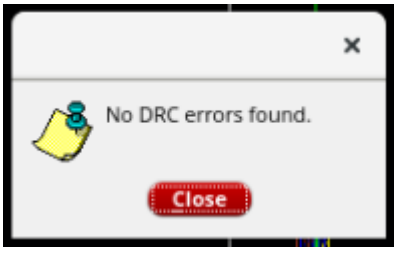



***Figure – 1.125: Result of DRC Check if the layout is DRC clean***

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.

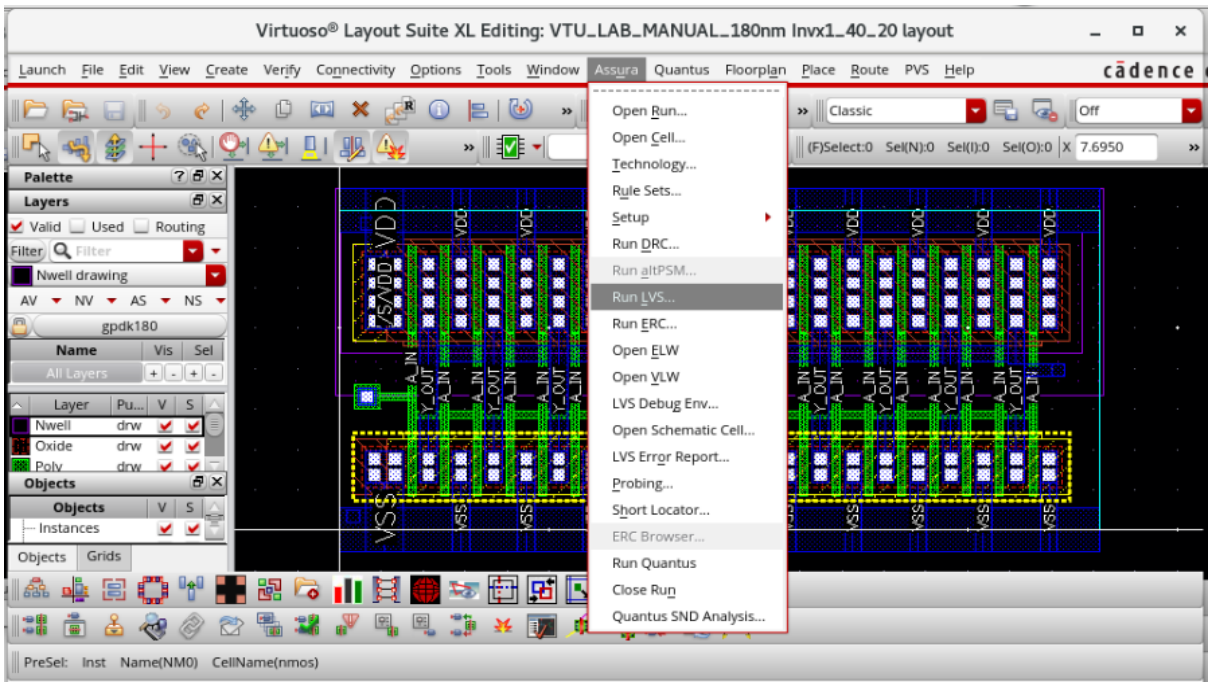



***Figure – 1.126: Assura → Run LVS***

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

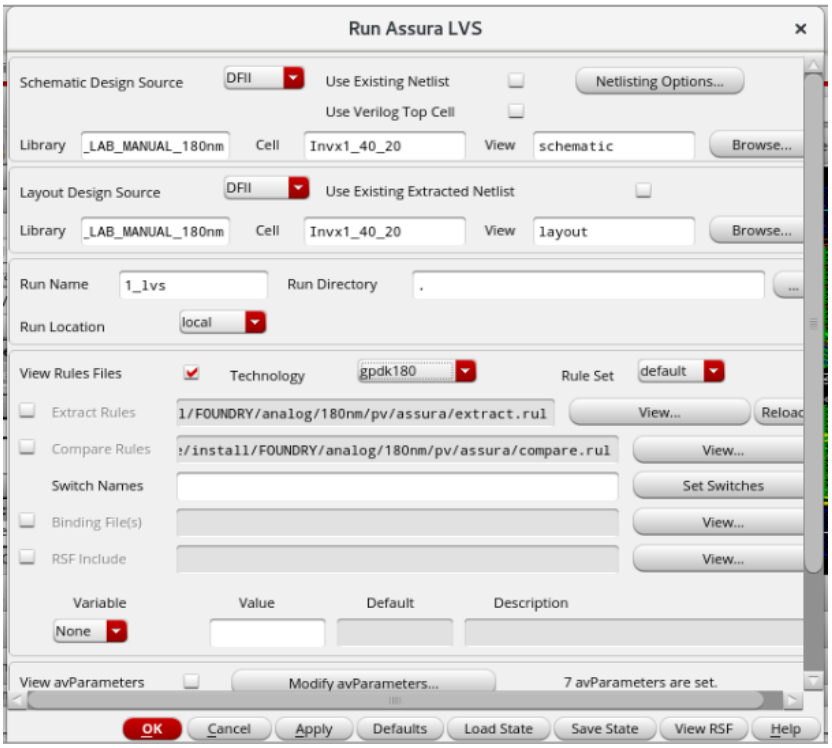



***Figure –1.127: Run Assura LVS window***

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.

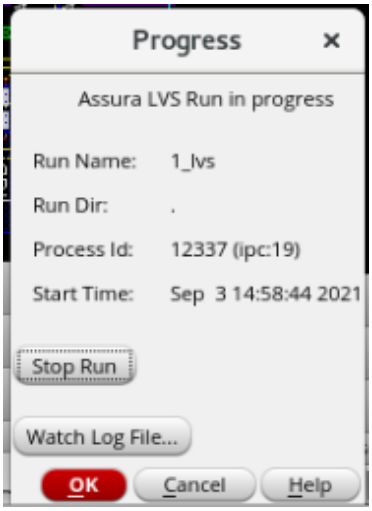



***Figure – 1.128: LVS "Progress" window***

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**".

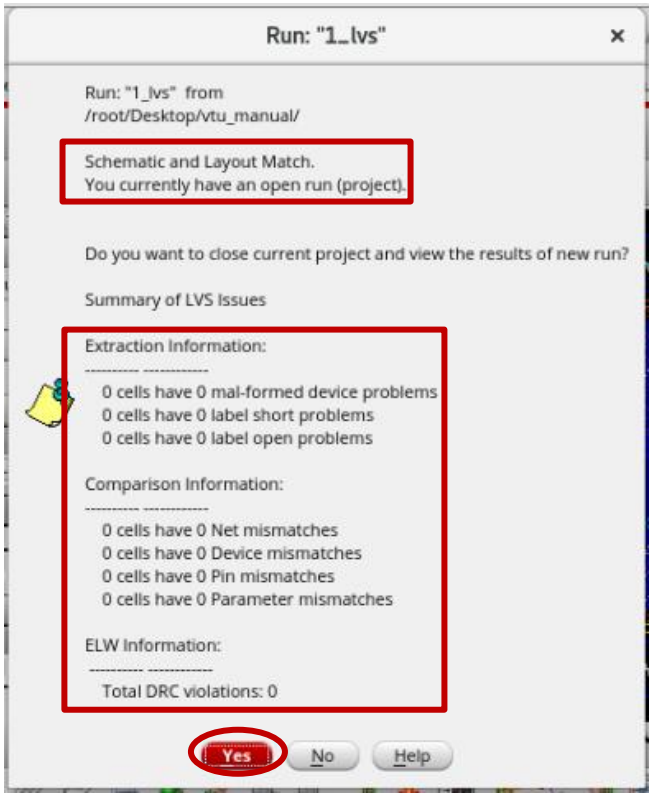



***Figure – 1.129: "Run: "1_lvs"" window***

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

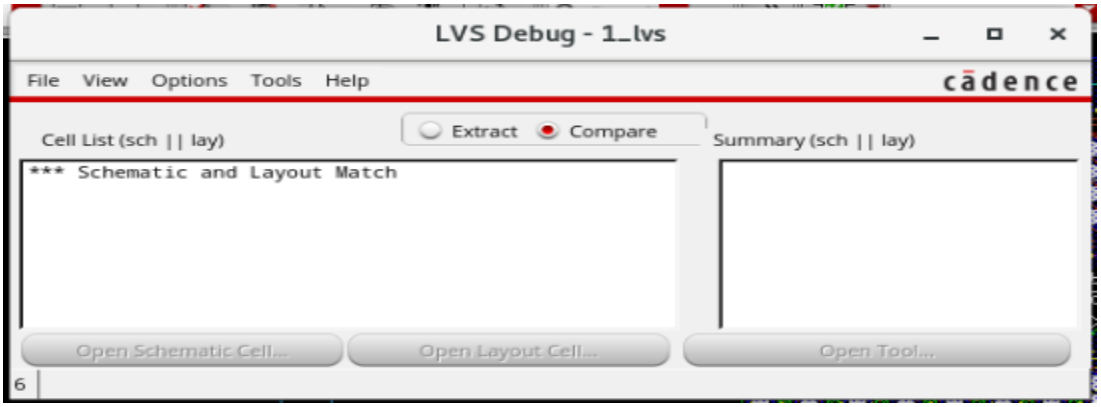



*Figure – 1.130: "LVS Debug" window*

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**".

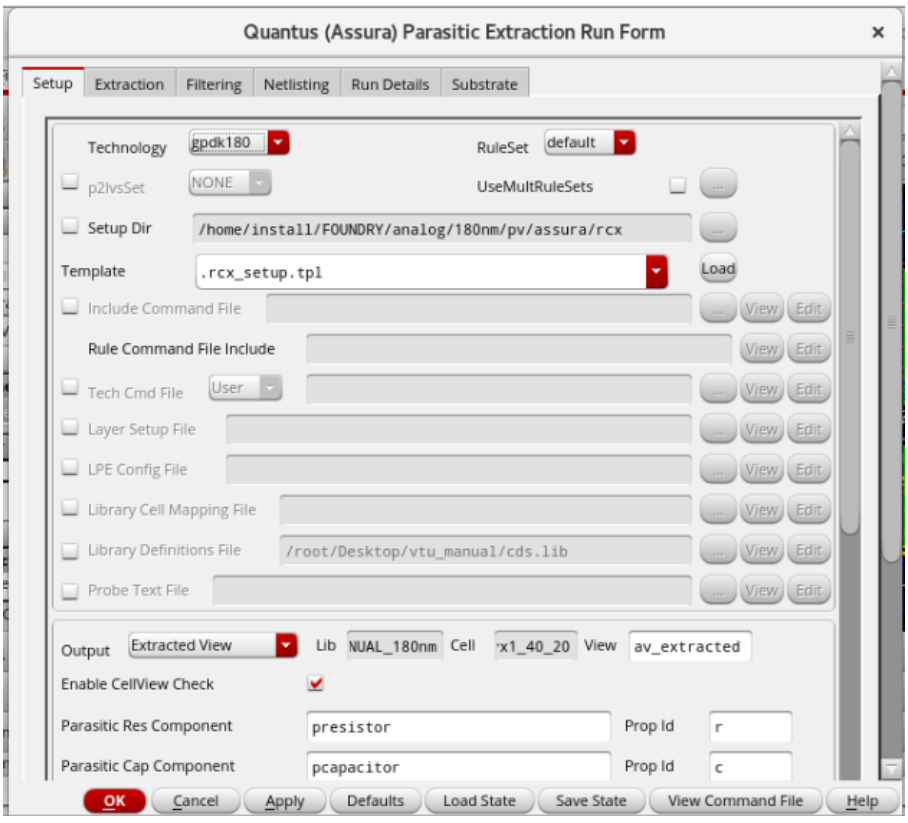



*Figure – 1.131: Quantus (Assura) Parasitic Extraction Run Form*

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.

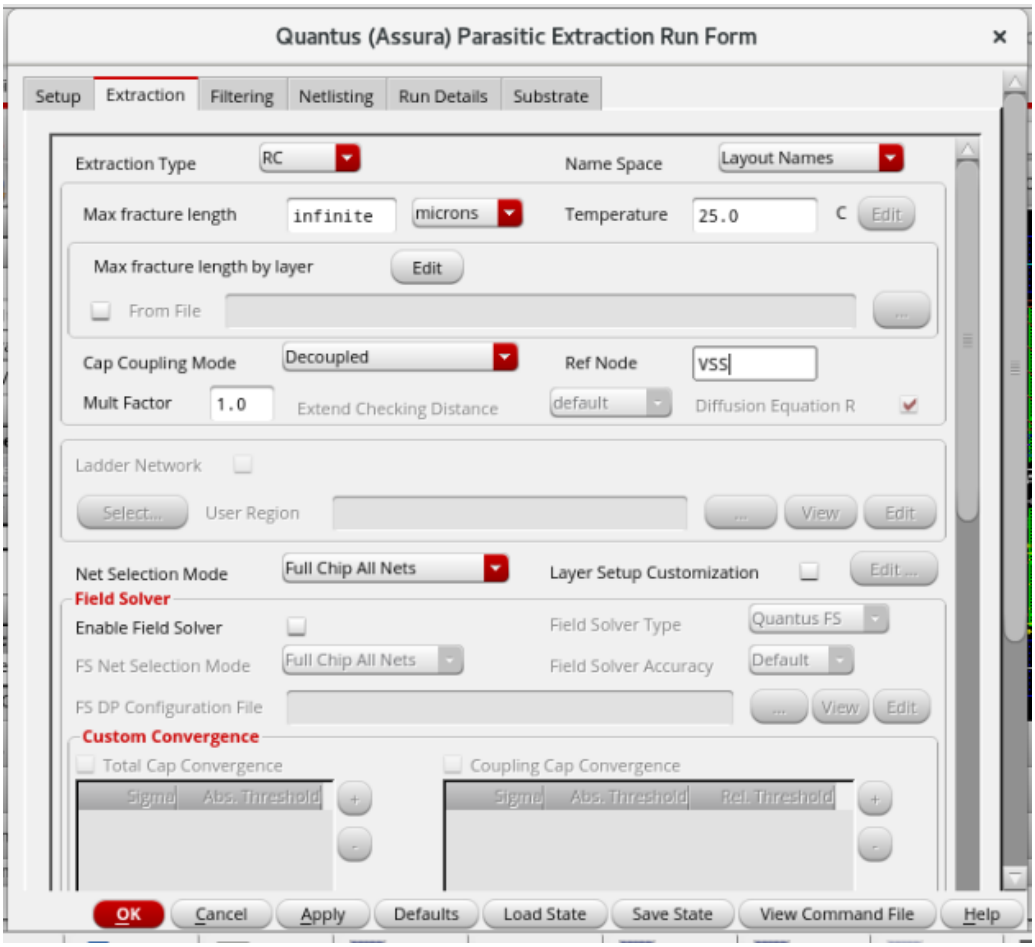



***Figure – 1.132: "Extraction" Tab***

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.

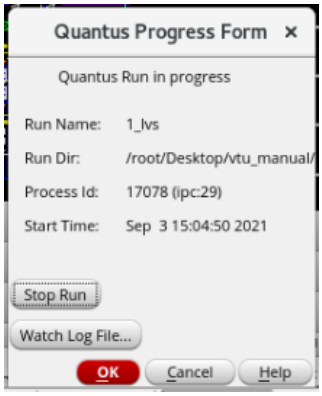



***Figure – 1.133: Quantus Progress Form***

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

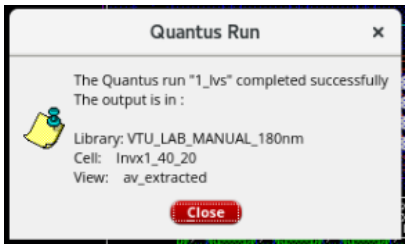



***Figure – 1.134: Quantus Run form***

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.

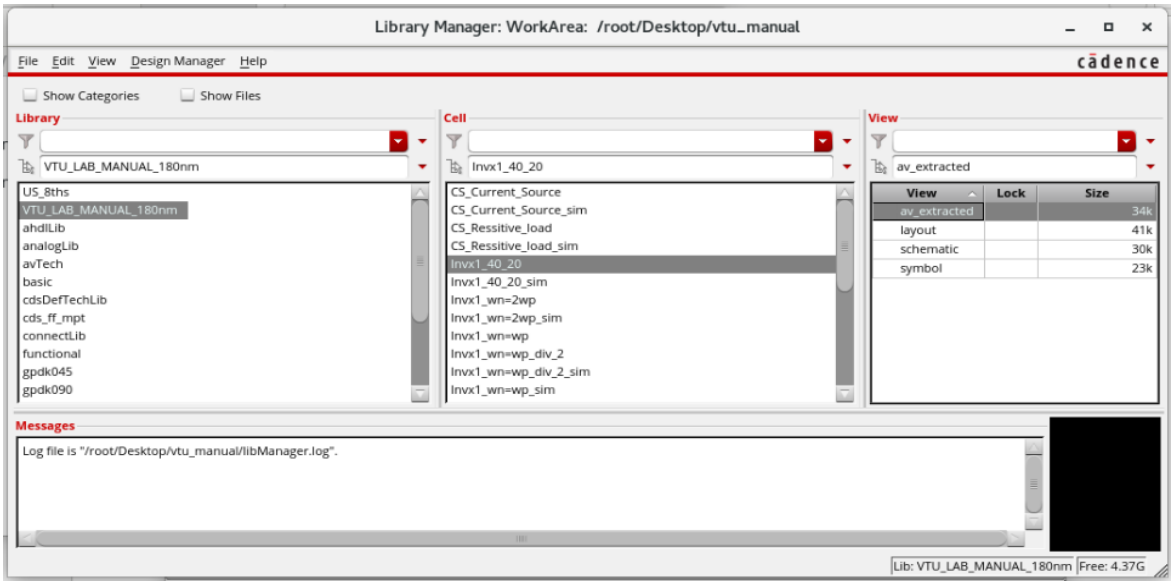



***Figure – 1.135: "av_extracted" view from Library Manager***

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

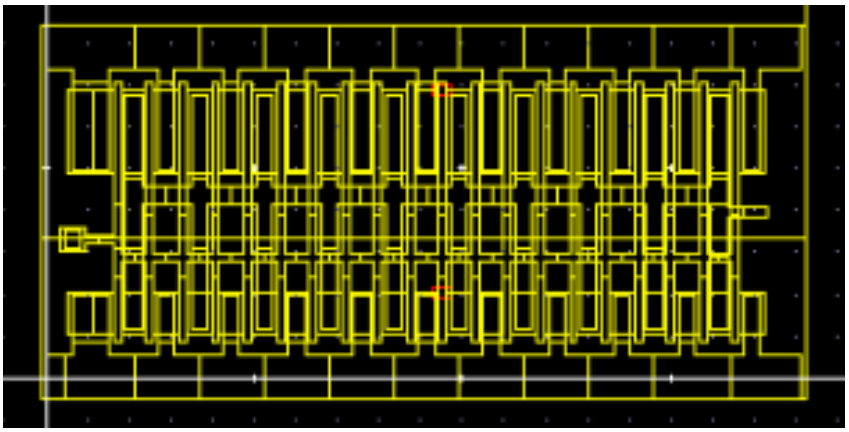


***Figure – 1.136: Extracted View***

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

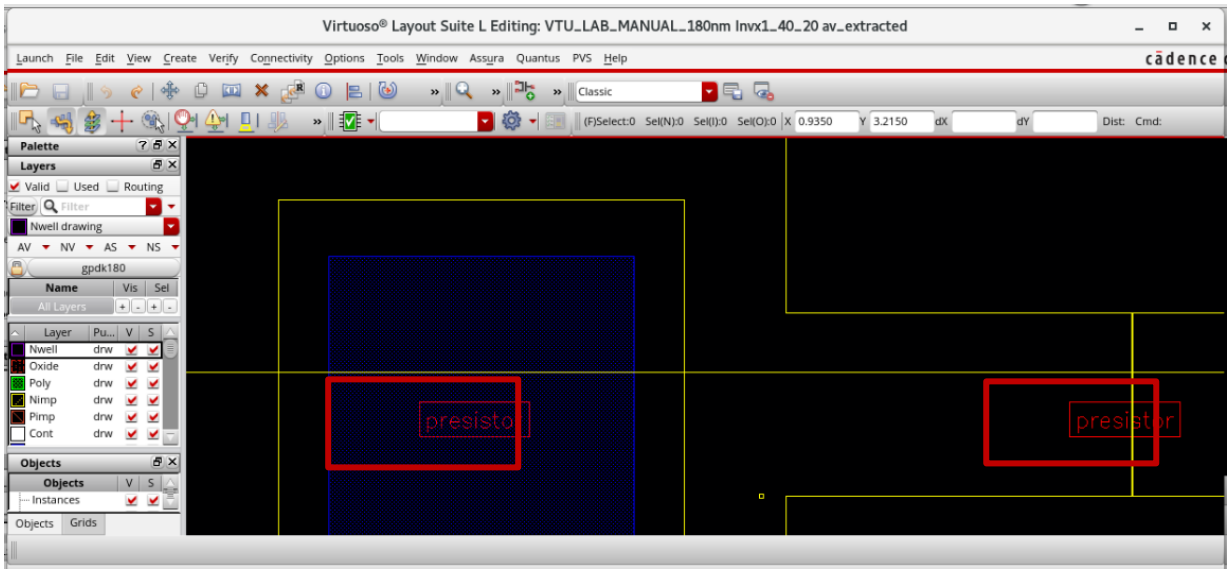



***Figure – 1.137: Parasitic Resistance (presistor) after Zoom In***

To check out the values of these Parasitic Resistance, click on "**Shift + F**". By zooming in further, the values can be checked out as shown in Figure – 1.138.

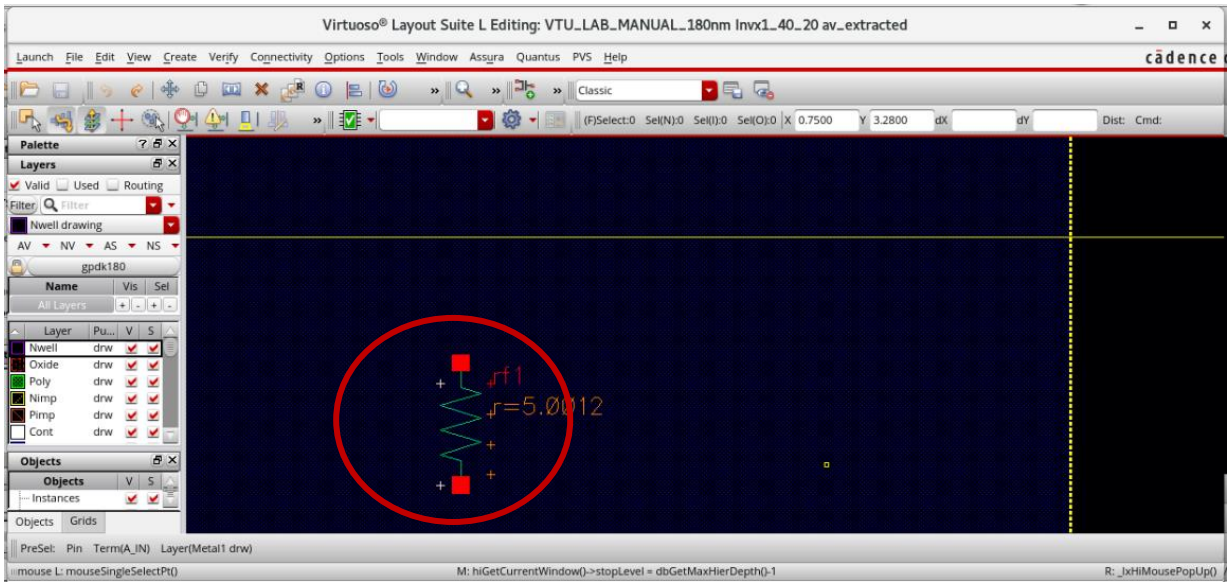



***Figure – 1.138: Value of Parasitic Resistance***

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.

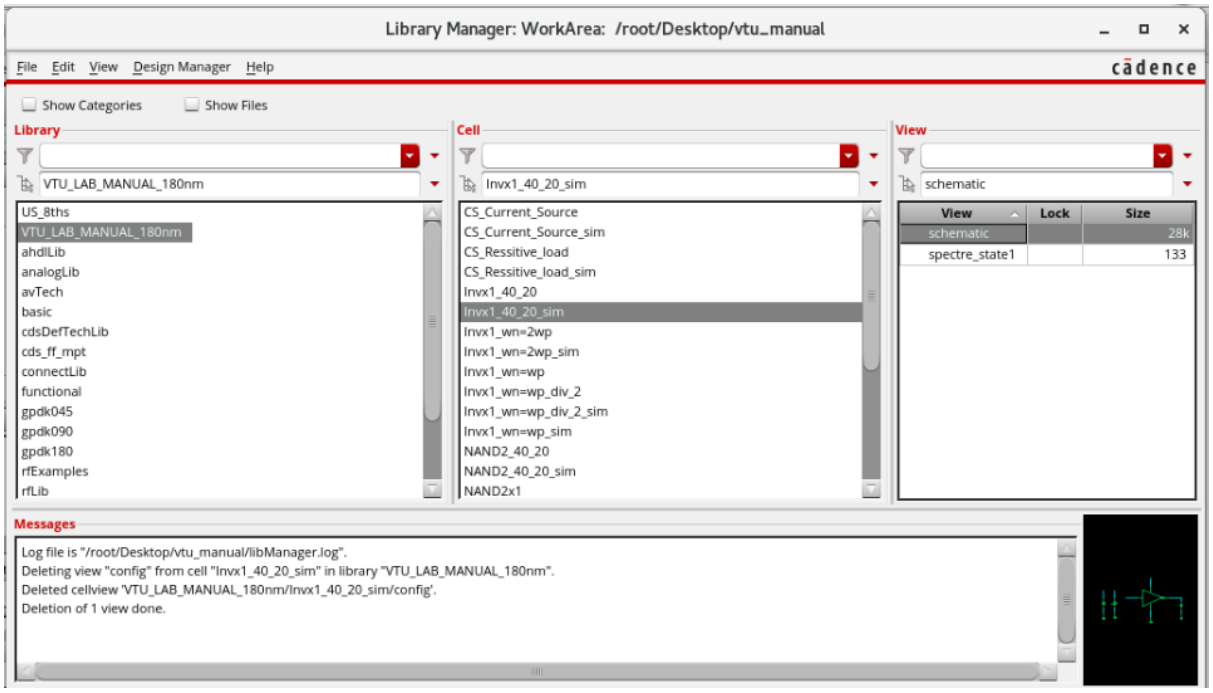



***Figure – 1.139: Selecting the "Schematic" view***

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

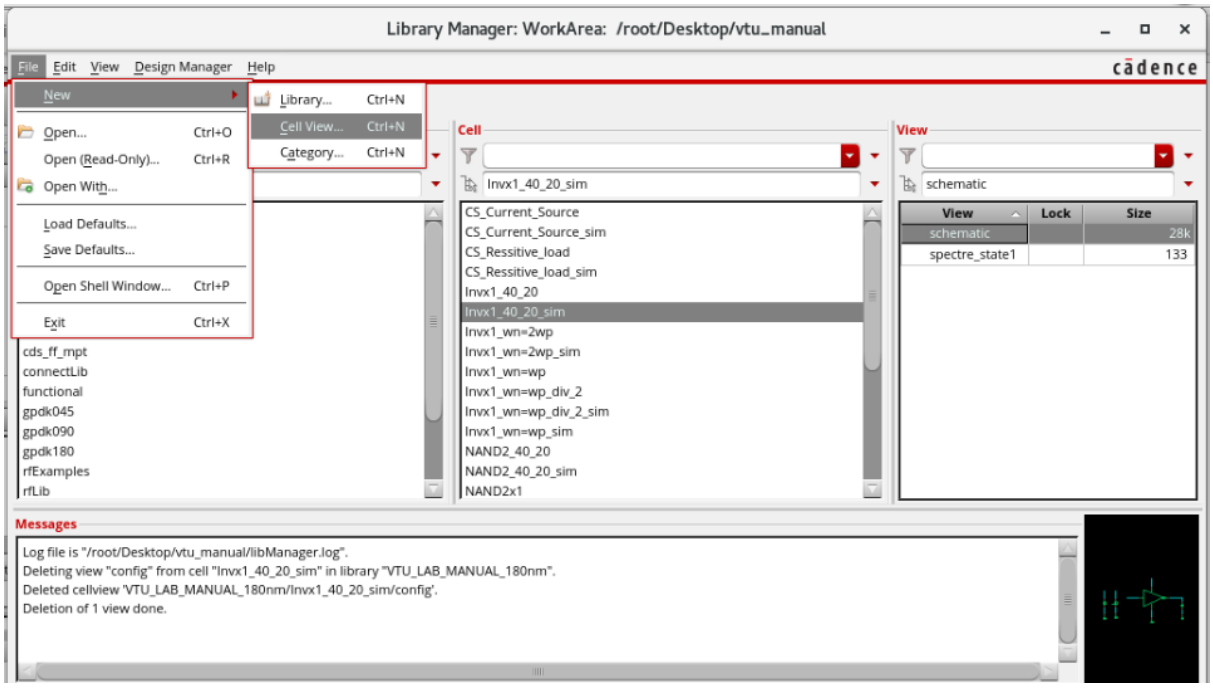



***Figure – 1.140: File → New → Cell View***

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**".

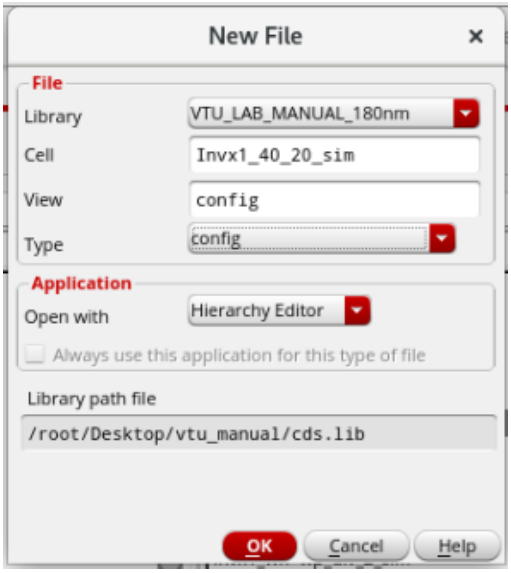



***Figure – 1.141: "New File" window***

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.

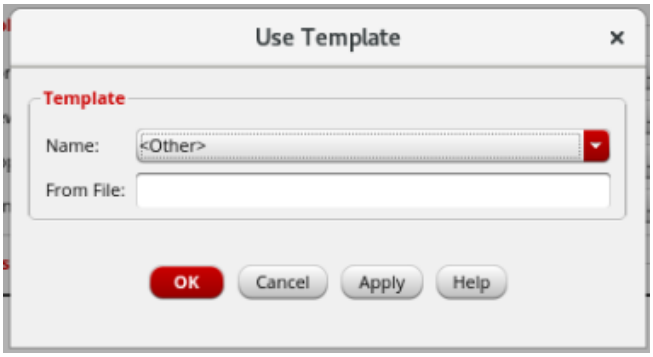



***Figure – 1.143: Use Template window***

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

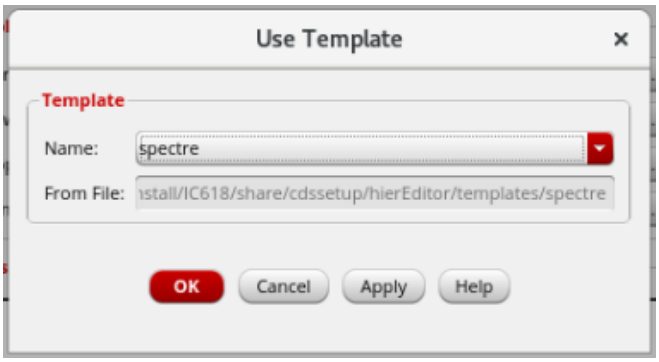



***Figure – 1.144: Name → Spectre***

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.

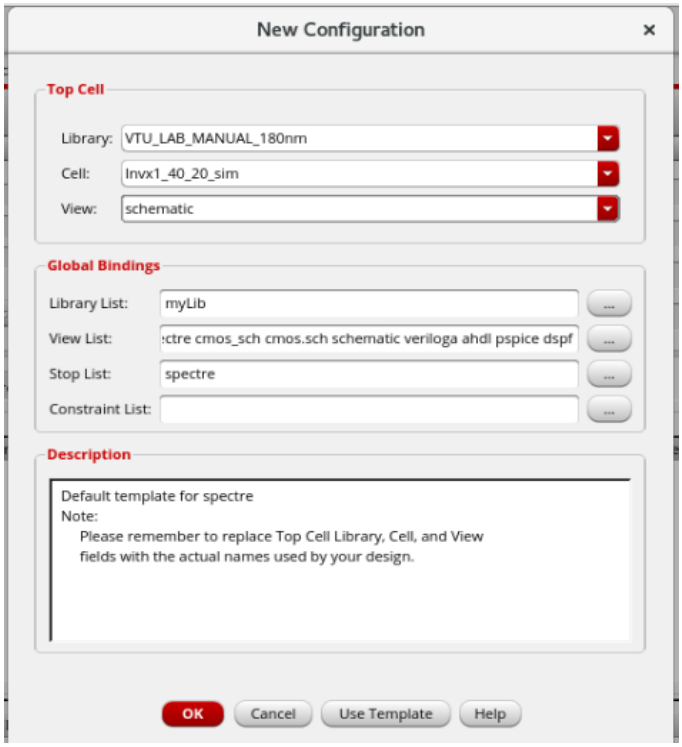



***Figure – 1.146: Top Cell → View → Schematic***

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

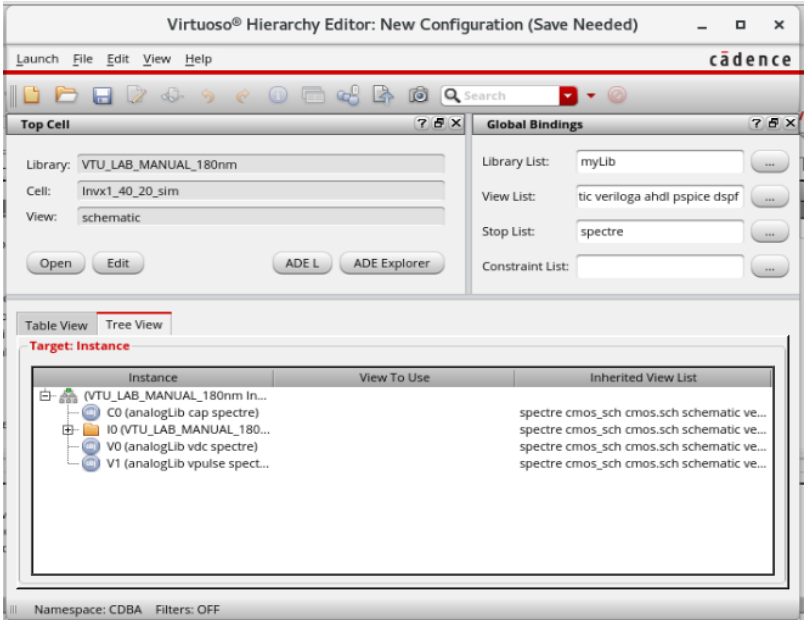



***Figure – 1.147: Virtuoso Hierarchy Editor: New Configuration window***

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**".

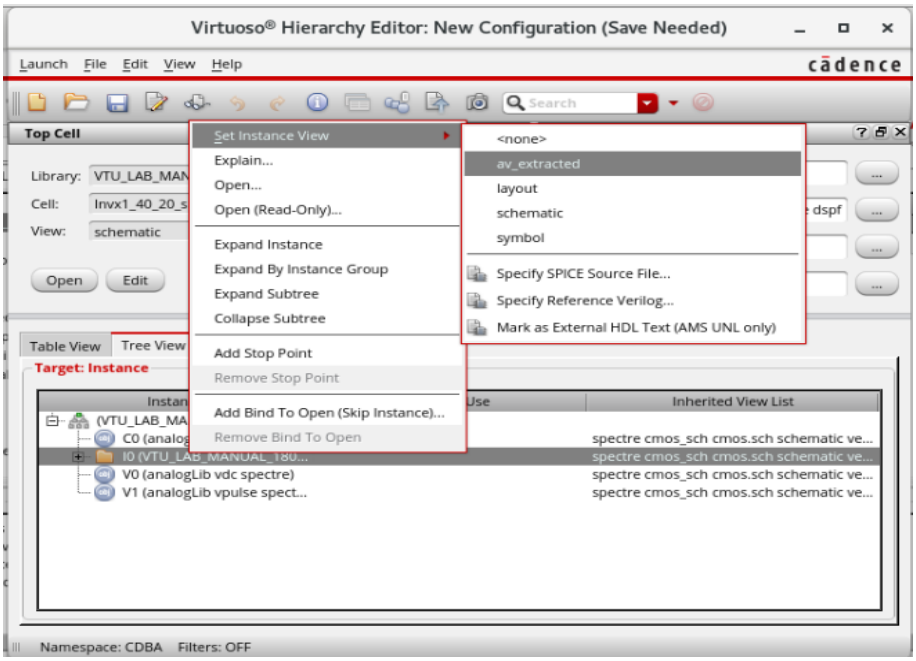



***Figure – 1.148: Set Instance View → av_extracted***

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

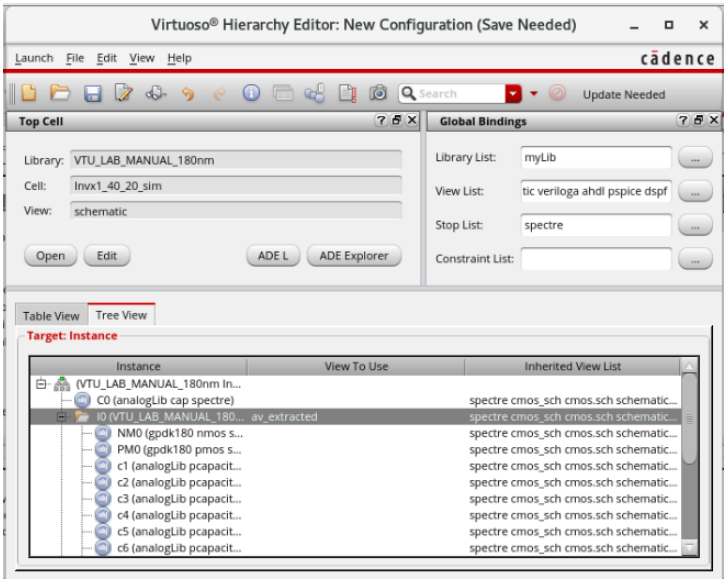



***Figure – 1.149: Imported Parasitics***

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

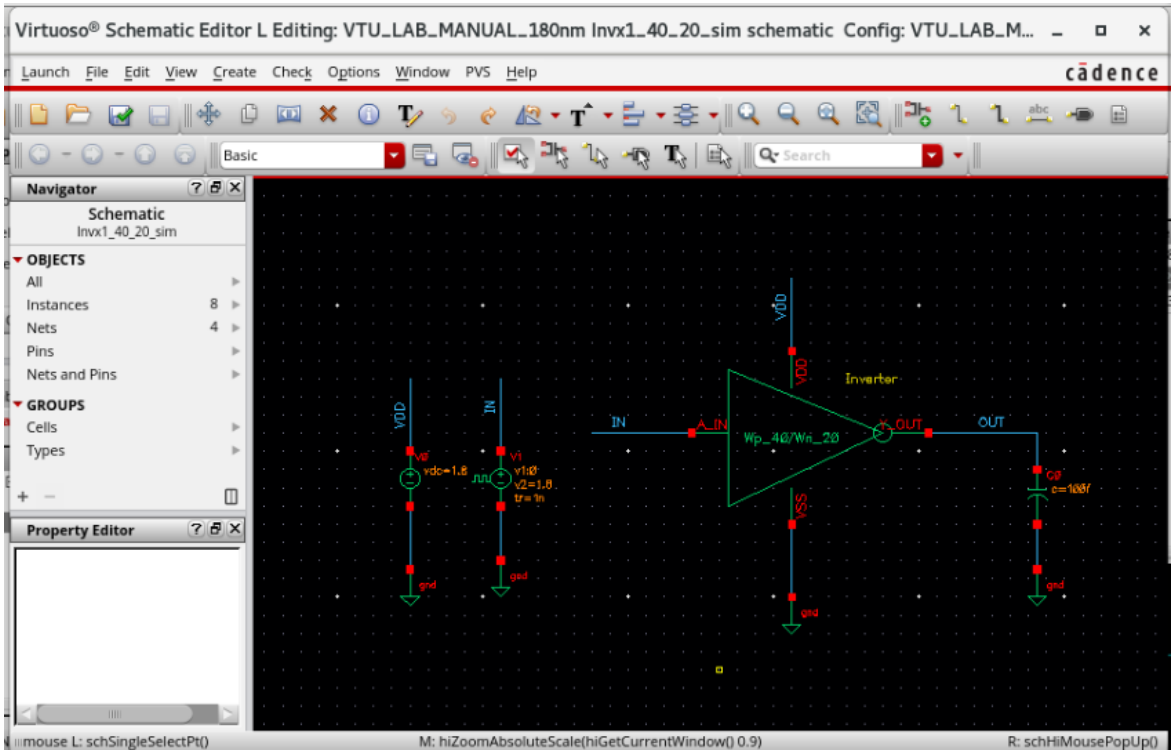



***Figure – 1.150: Test Schematic***

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**"

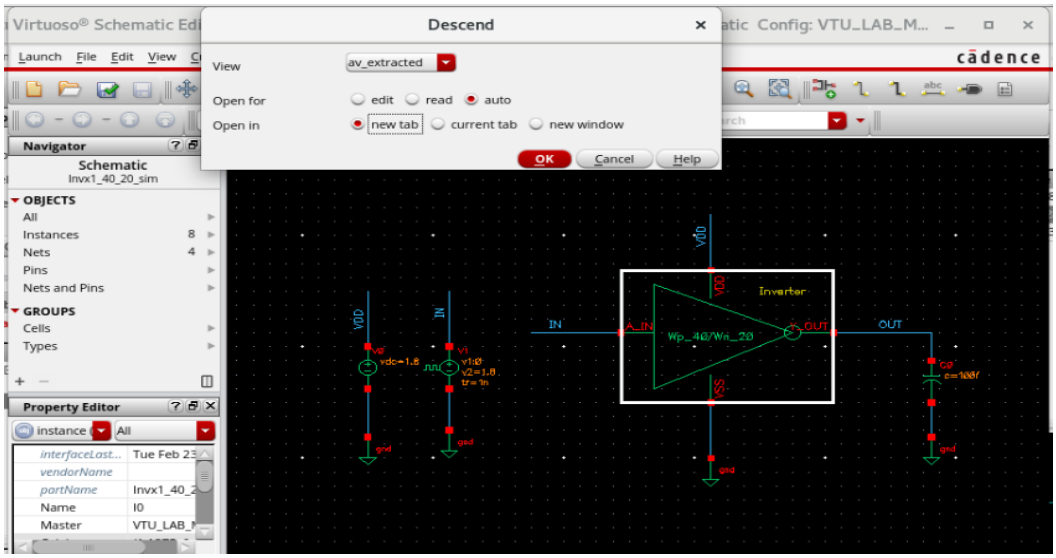



***Figure – 1.151: Descend window***

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.

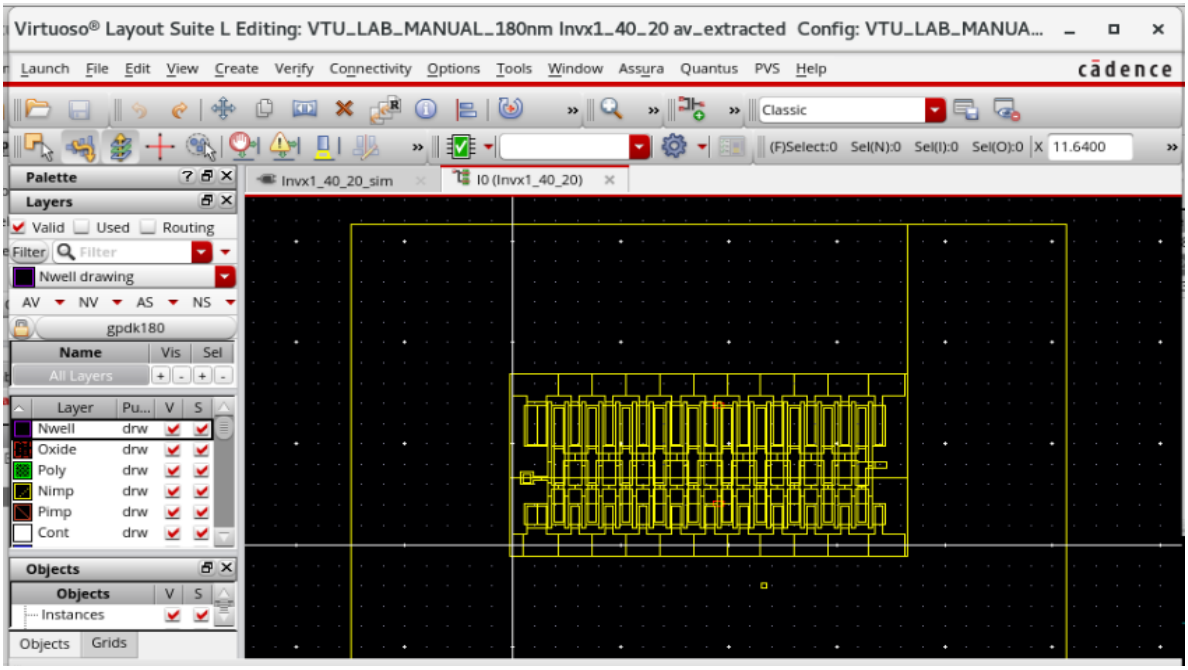



***Figure – 1.152: av_extracted view in a new tab***

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

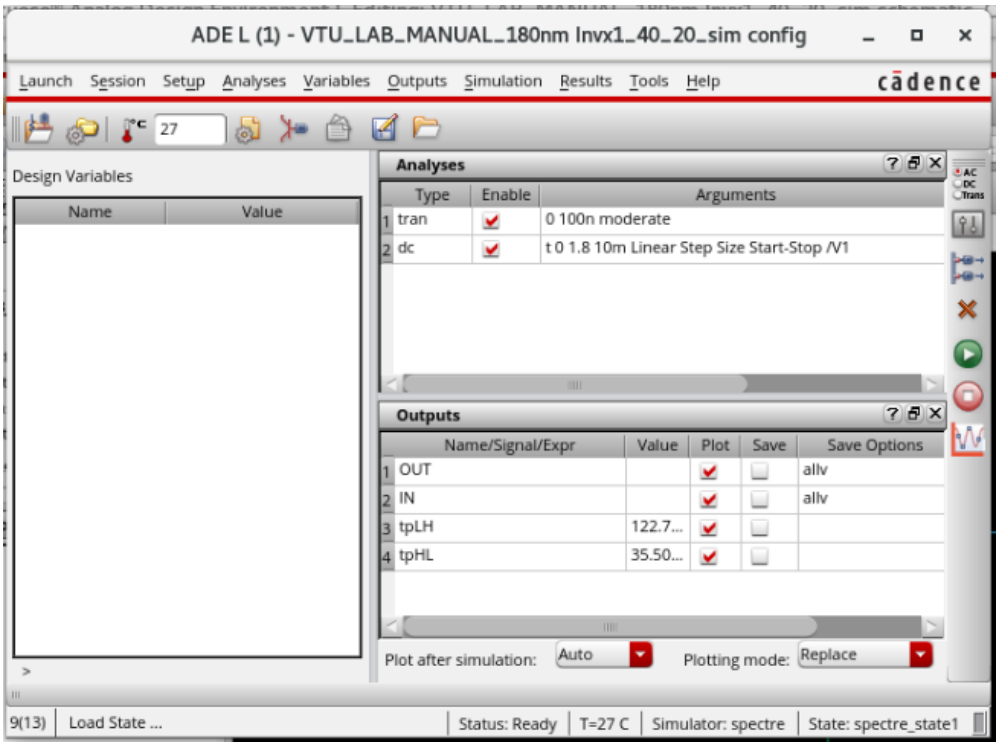



***Figure – 1.153: Saved State***

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

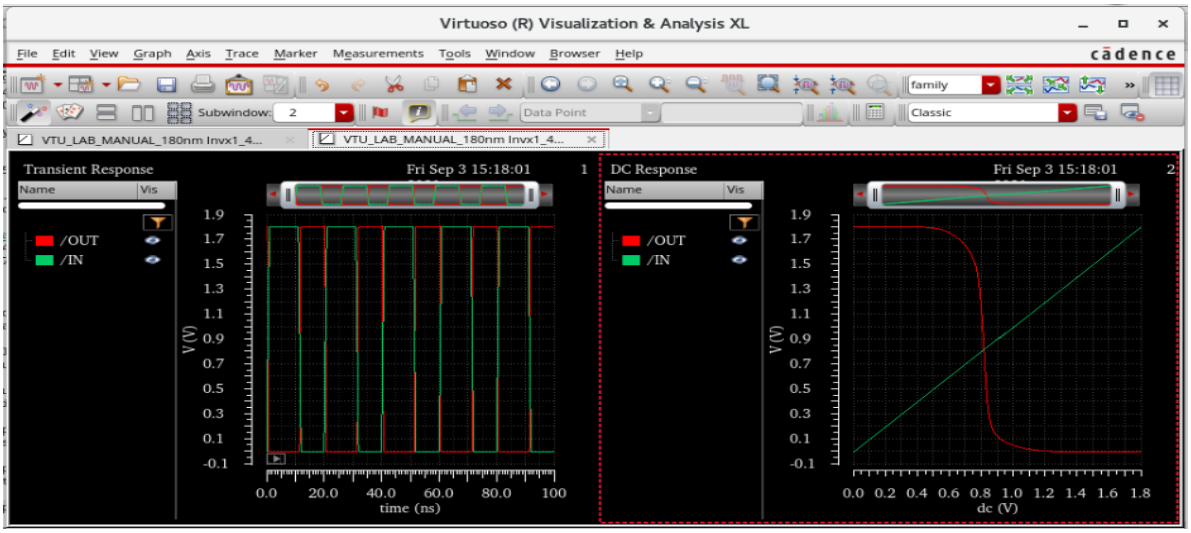



***Figure – 1.154: Waveforms of Transient Analysis and DC Analysis***

Using the Calculator, obtain the Switching Potential, $tp_{HL}$, $tp_{LH}$ and $t_{PD}$.The results are tabulated 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 $\frac{W_P}{W_N} = \frac{40}{20}$, 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<br>Length, L = 180 n |
| gpdk180 | Pmos | Width, $W_P$ = 1.275 u<br>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 Gate with Drive Strength "2"***

| Library Name | Cell Name | Comments / Properties |
|---|---|---|
| gpdk180 | Nmos | Width, $W_N$ = 3.4 u<br>Length, L = 180 n |
| gpdk180 | Pmos | Width, $W_P$ = 2.55 u<br>Length, L = 180 n |

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

| Library Name | Cell Name | Comments / Properties |
|---|---|---|
| gpdk180 | Nmos | Width, $W_N$ = 6.8 u<br>Length, L = 180 n |
| gpdk180 | Pmos | Width, $W_P$ = 5.1 u<br>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.

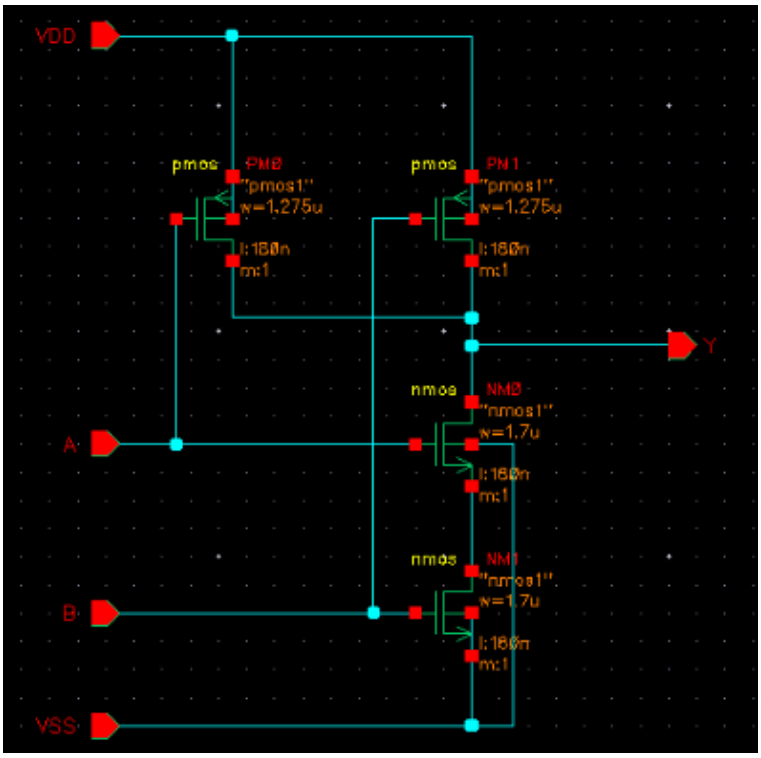



*Figure – 2.1: Schematic Capture of 2 – input CMOS NAND Gate (NAND2X1)*

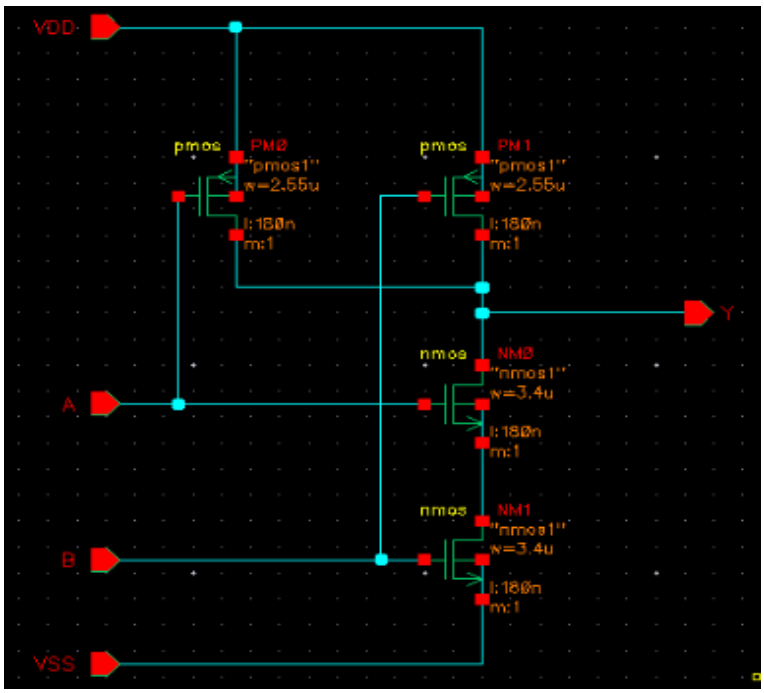



*Figure – 2.2: Schematic Capture of 2 – input CMOS NAND Gate with drive strength 2 (NAND2X2)*

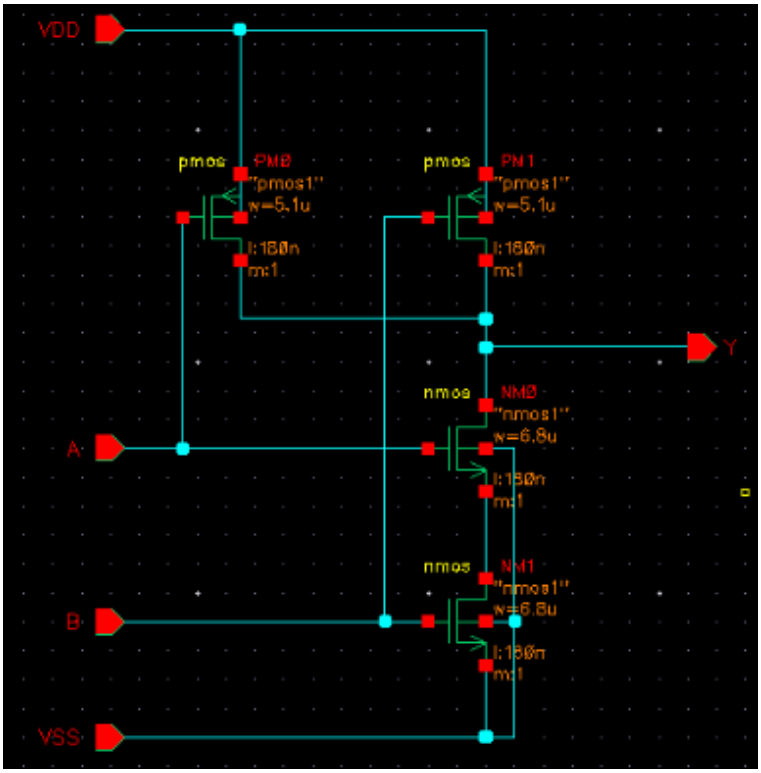



***Figure – 2.3: Schematic Capture of 2 – input CMOS NAND Gate with drive strength 2 (NAND2X4)***

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

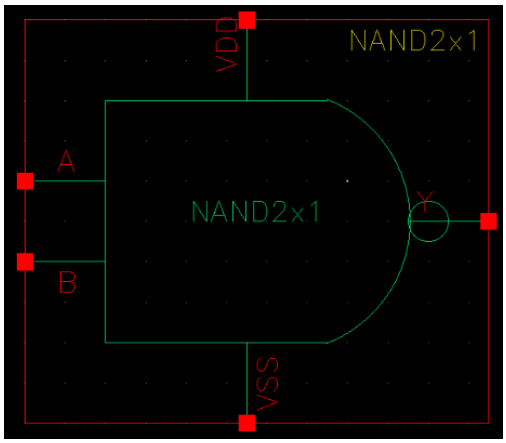



***Figure – 2.4: Symbol of 2 – input NAND Gate***

**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.

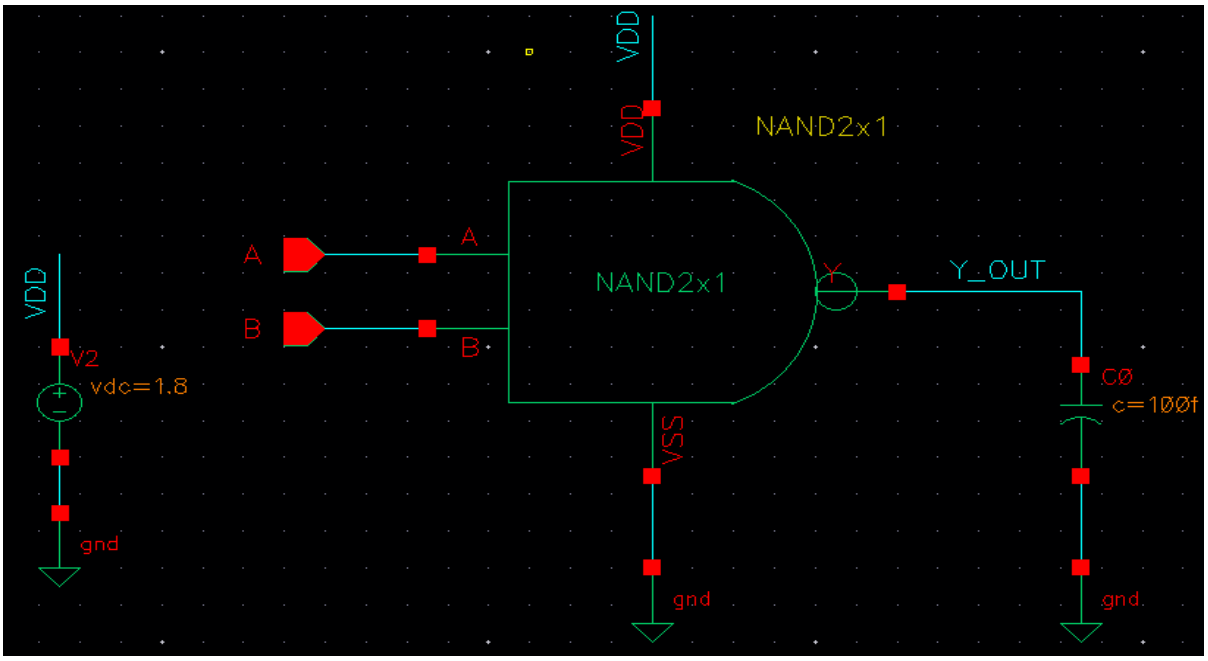



***Figure – 2.5: Test Schematic for 2 – input CMOS NAND Gate***

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

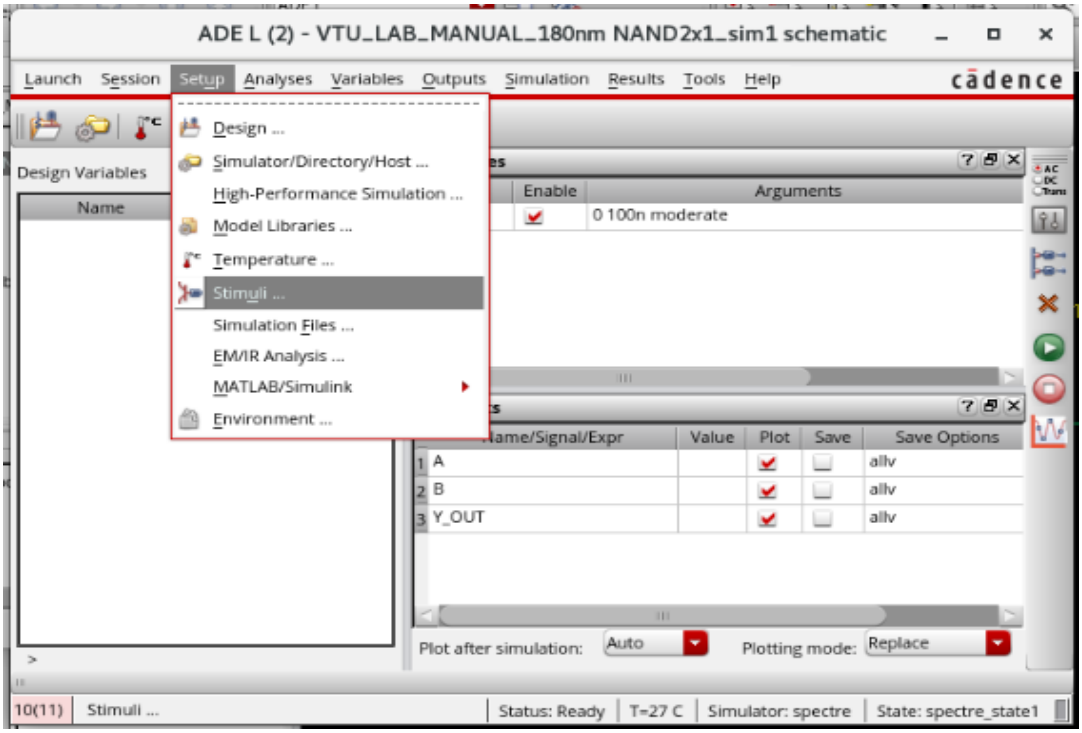



***Figure – 2.6: Setup → Stimuli***

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

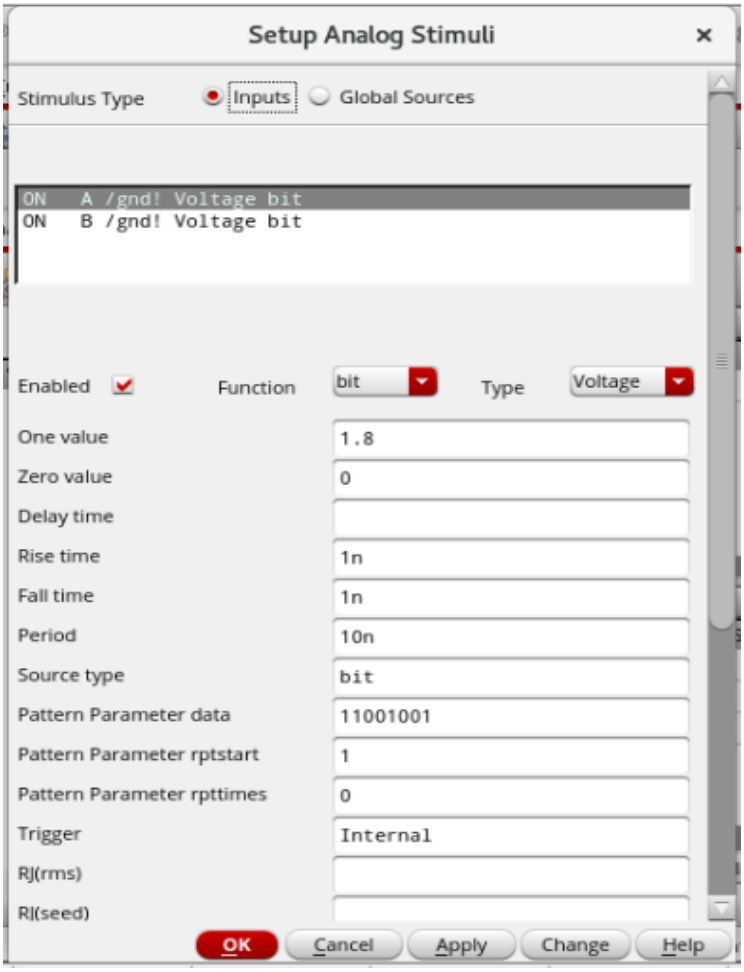



***Figure – 2.7: Setup Analog Stimuli window***

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 "**Zero value → 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.

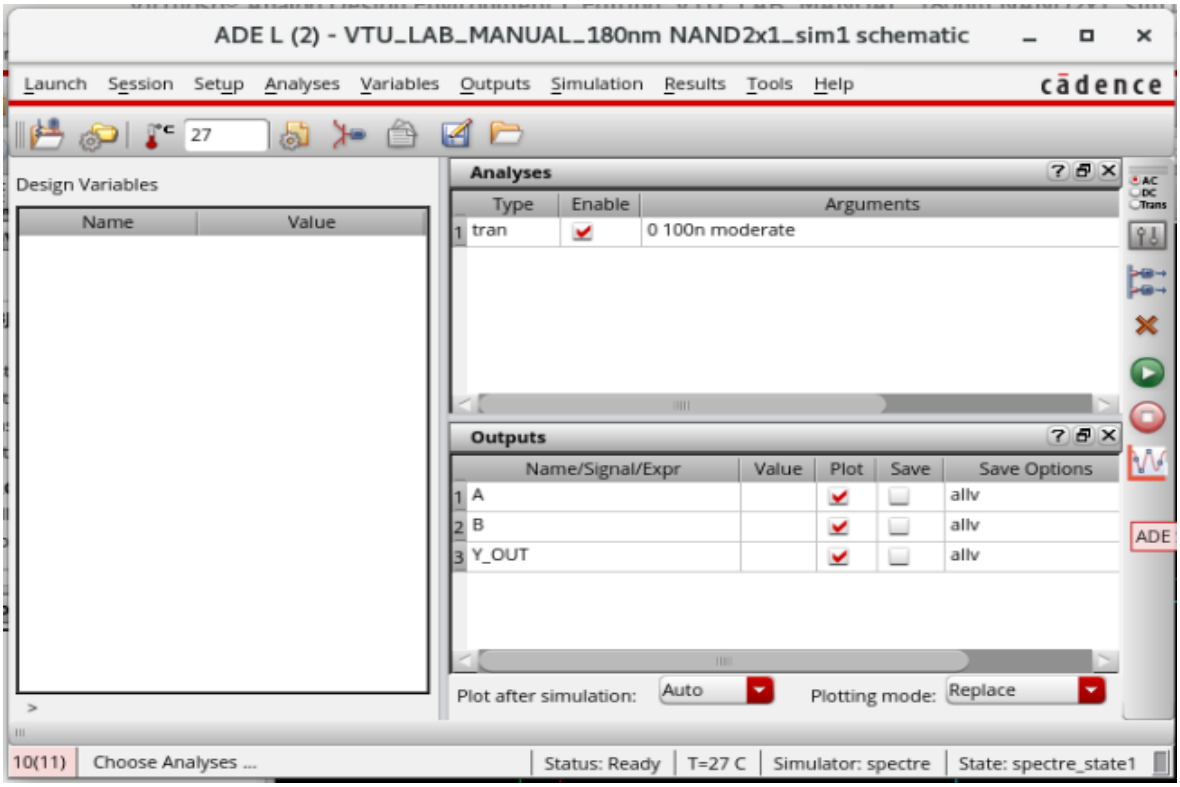



***Figure – 2.8: Updated ADE L window***

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

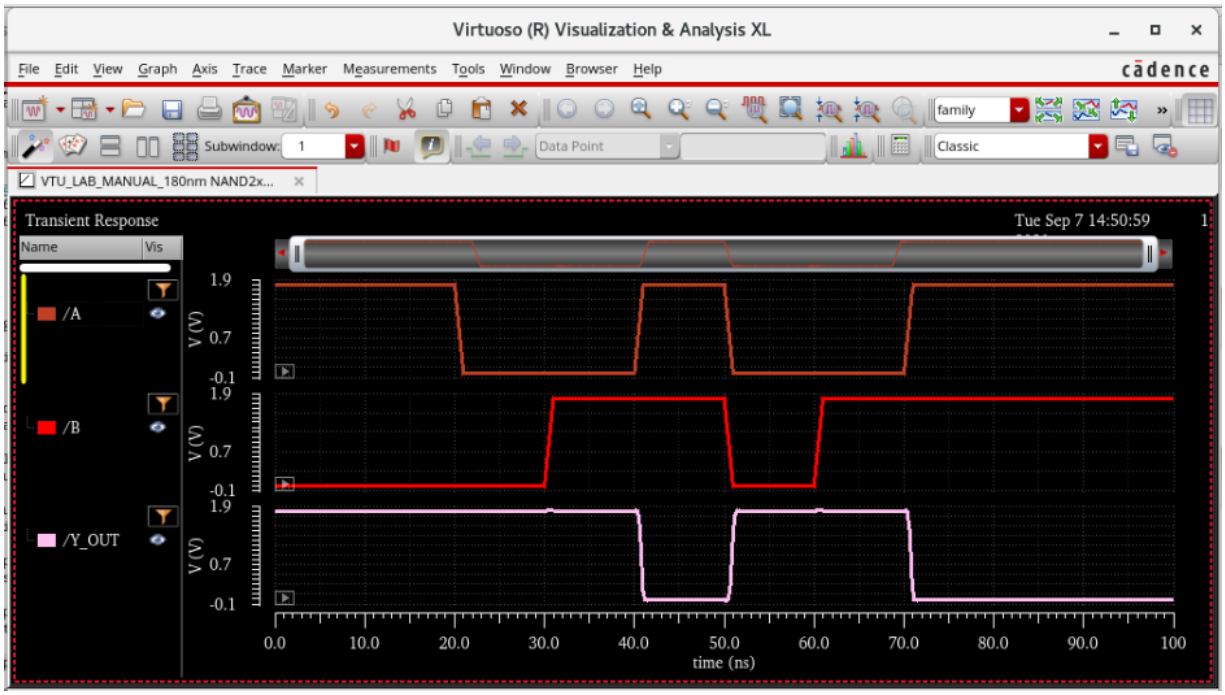



***Figure – 2.9: Transient Analysis of 2 – input CMOS NAND Gate***

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<br>Length, L = 180 n |
| gpdk180 | Pmos | Width, $W_P$ = 40 u<br>Length, L = 180 n |

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

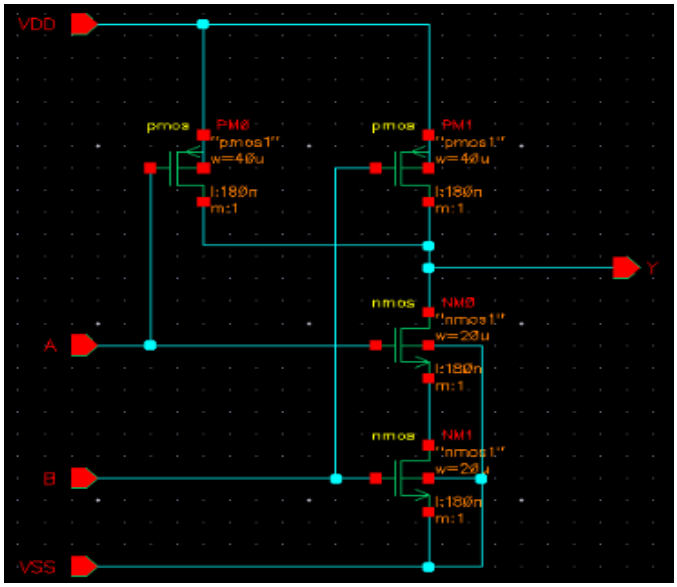



***Figure – 2.10: Schematic for 2 – input CMOS NAND with*** $\frac{W_P}{W_N} = \frac{40}{20}$

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

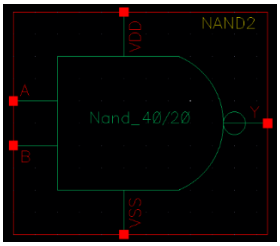



***Figure – 2.11: Symbol for 2 – input CMOS NAND with*** $\frac{W_P}{W_N} = \frac{40}{20}$

**FUNCTIONAL SIMULATION:**

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

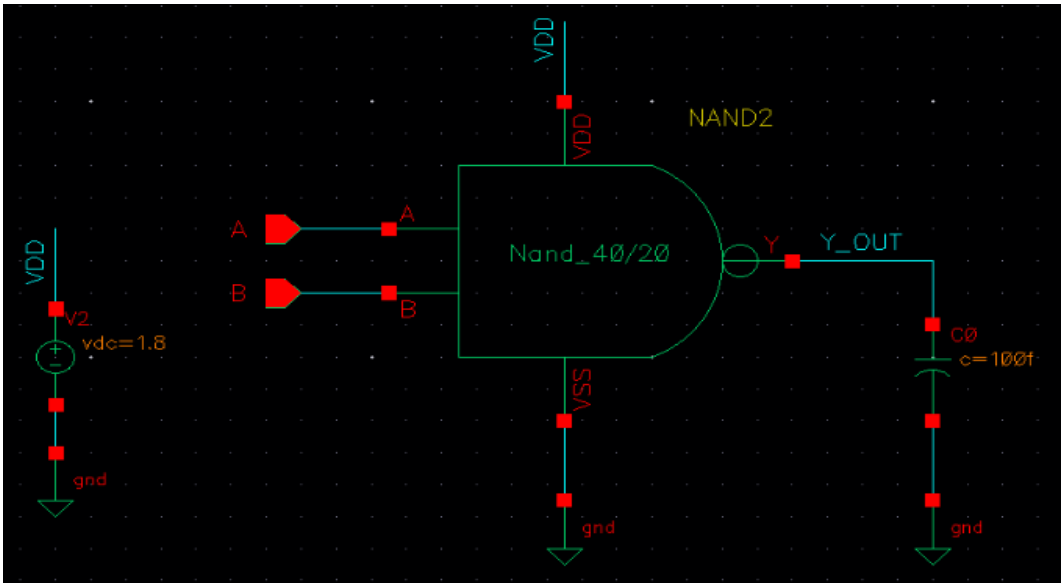



***Figure – 2.12: Test Schematic for 2 – input CMOS NAND with*** $\frac{W_P}{W_N} = \frac{40}{20}$

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

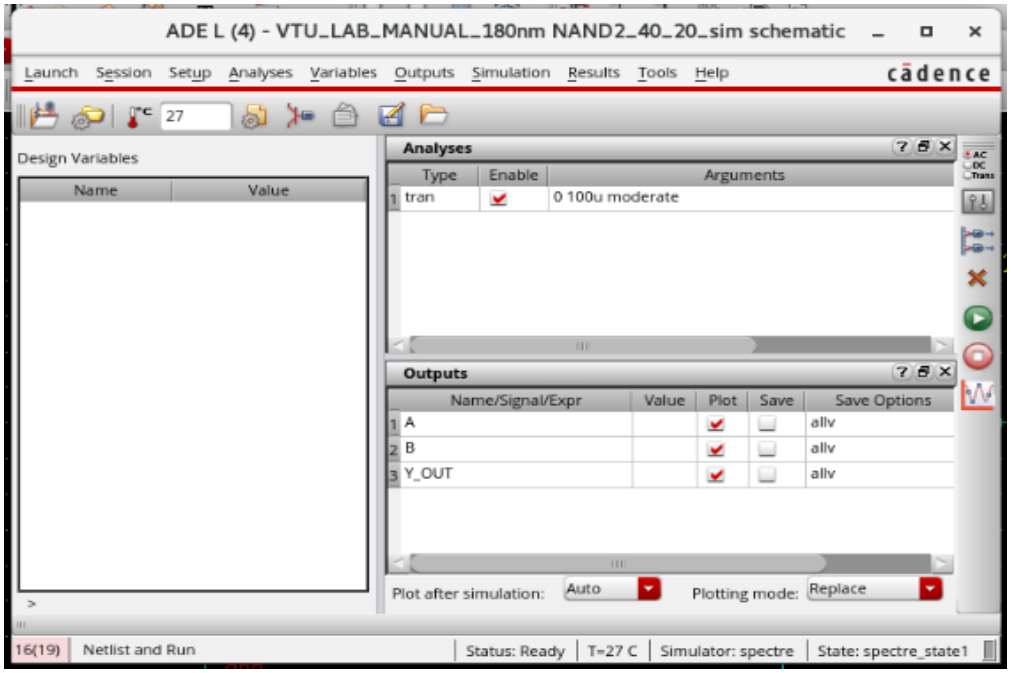



***Figure – 2.13: Updated ADE L window***

The waveforms after simulation is shown in Figure – 2.14.

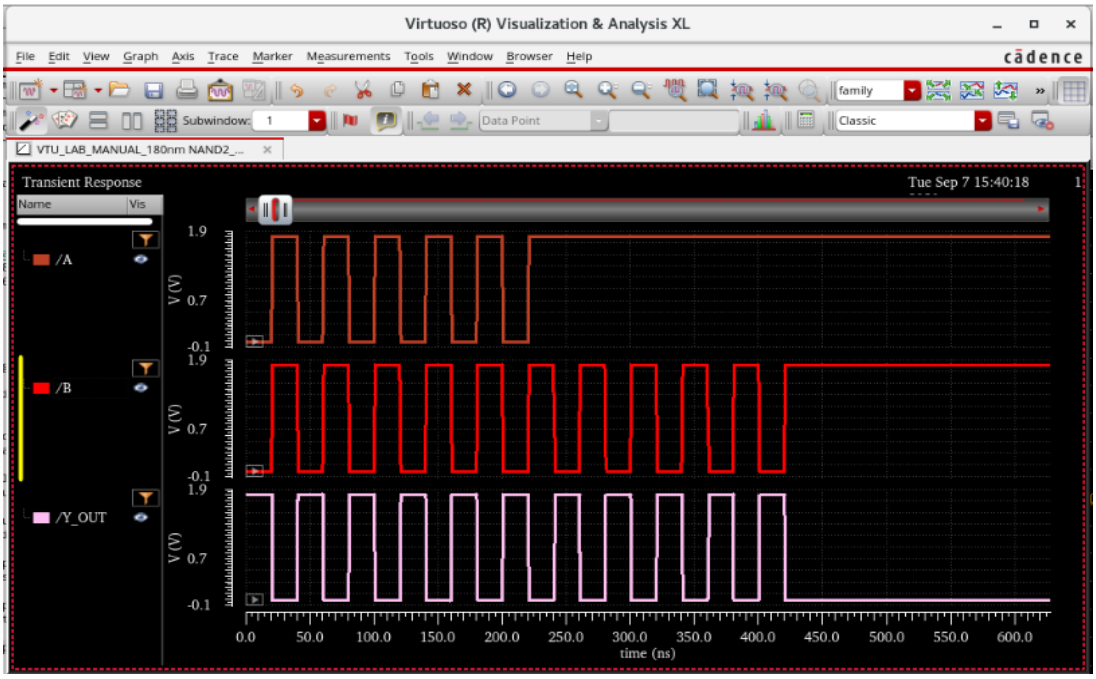



***Figure – 2.14: Simulated Waveforms for 2 – input CMOS NAND with*** $\frac{W_P}{W_N} = \frac{40}{20}$

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

***Table – 14: Delay Elements for 2 – input CMOS NAND Gate with*** $\frac{W_P}{W_N} = \frac{40}{20}$ ***(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.

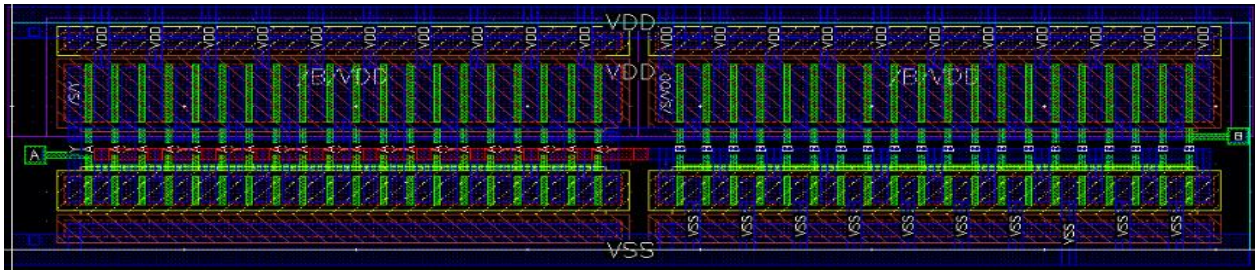



***Figure – 2.15: Layout for 2 – input CMOS NAND Gate with*** $\frac{W_P}{W_N} = \frac{40}{20}$

**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 MIRROR LOAD

**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<br>Length, L = 180 n |
| gpdk180 | Pmos | Width, $W_P$ = 8.85 u<br>Length, L = 180 n |

The completed Schematic is shown in Figure – 3.1.

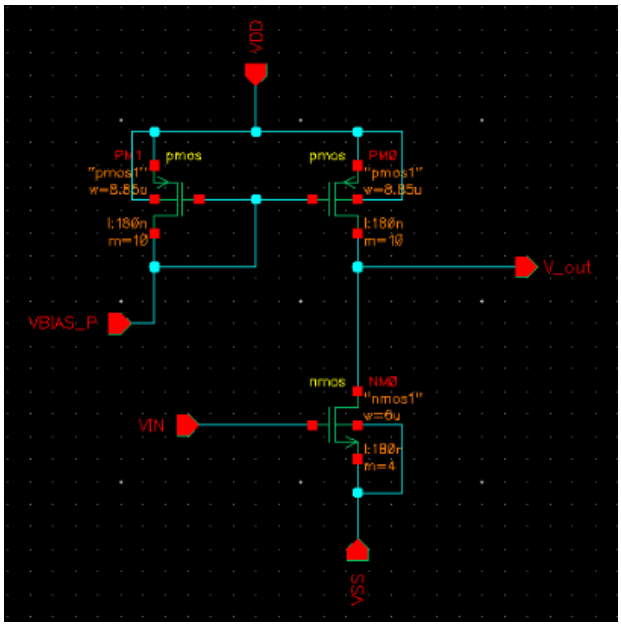



***Figure – 3.1: Schematic of Common Source Amplifier with PMOS Current Mirror Load***

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

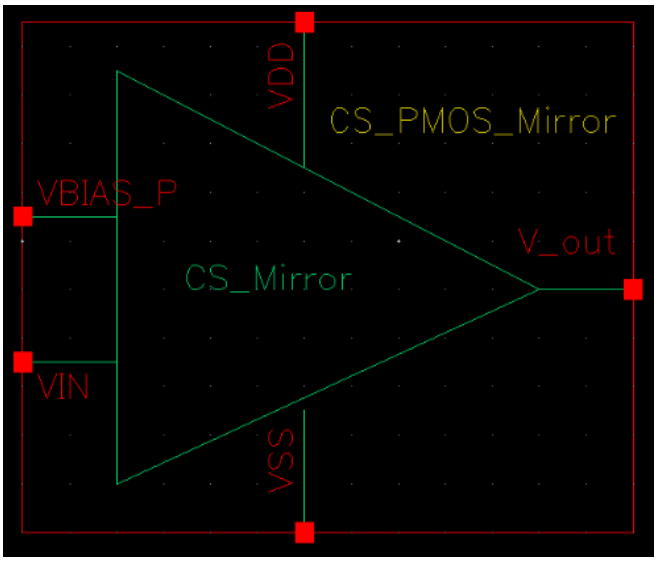



***Figure – 3.2: Symbol of Common Source Amplifier with PMOS Current Mirror Load***

**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.

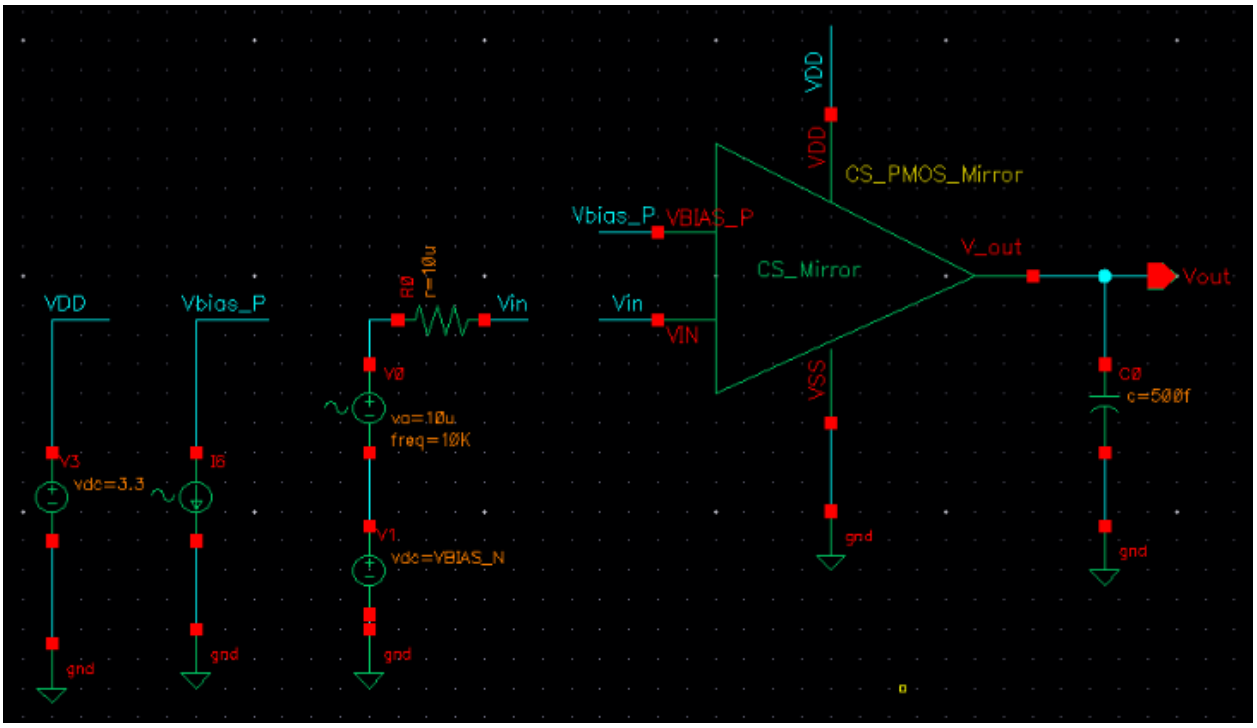



***Figure – 3.3: Test Schematic for 2 – input CMOS NAND Gate***

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.

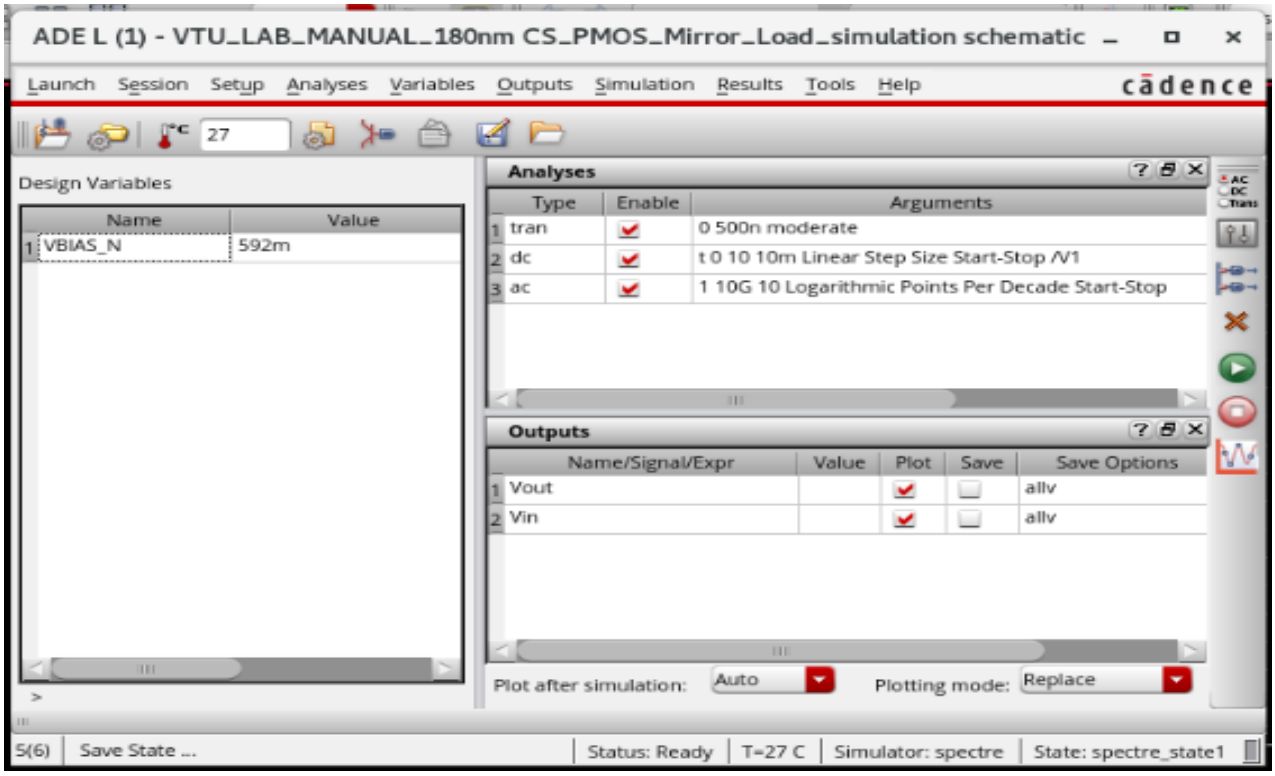



***Figure – 3.4: Updated ADE L window***

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

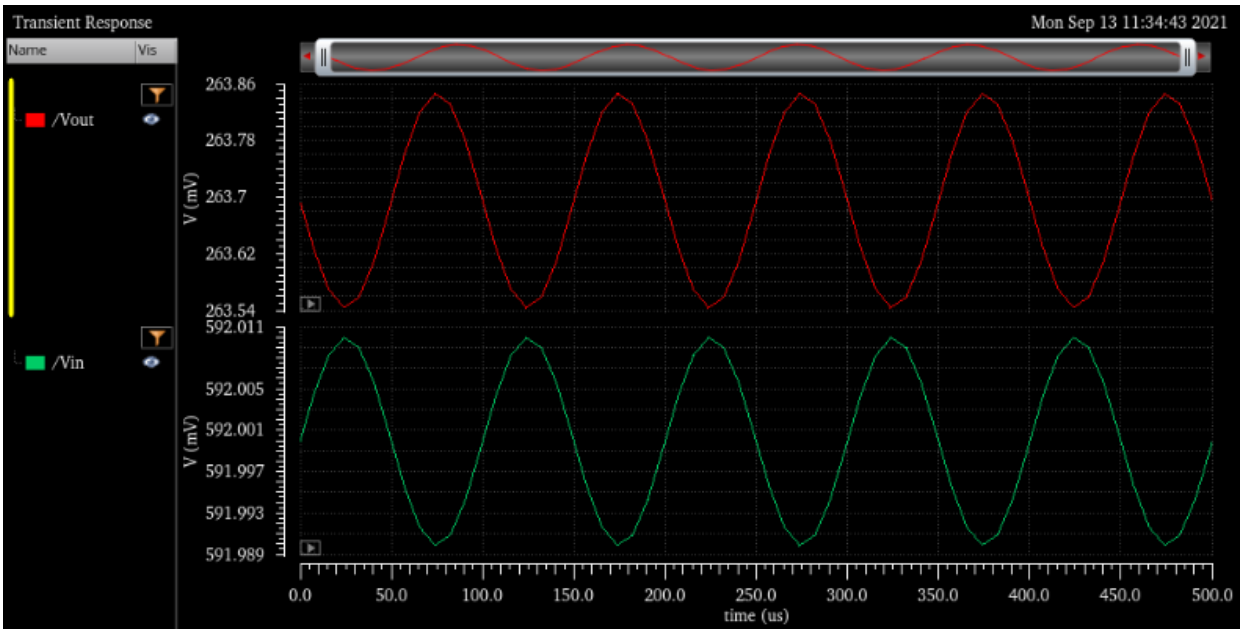



***Figure – 3.5: Transient Analysis***

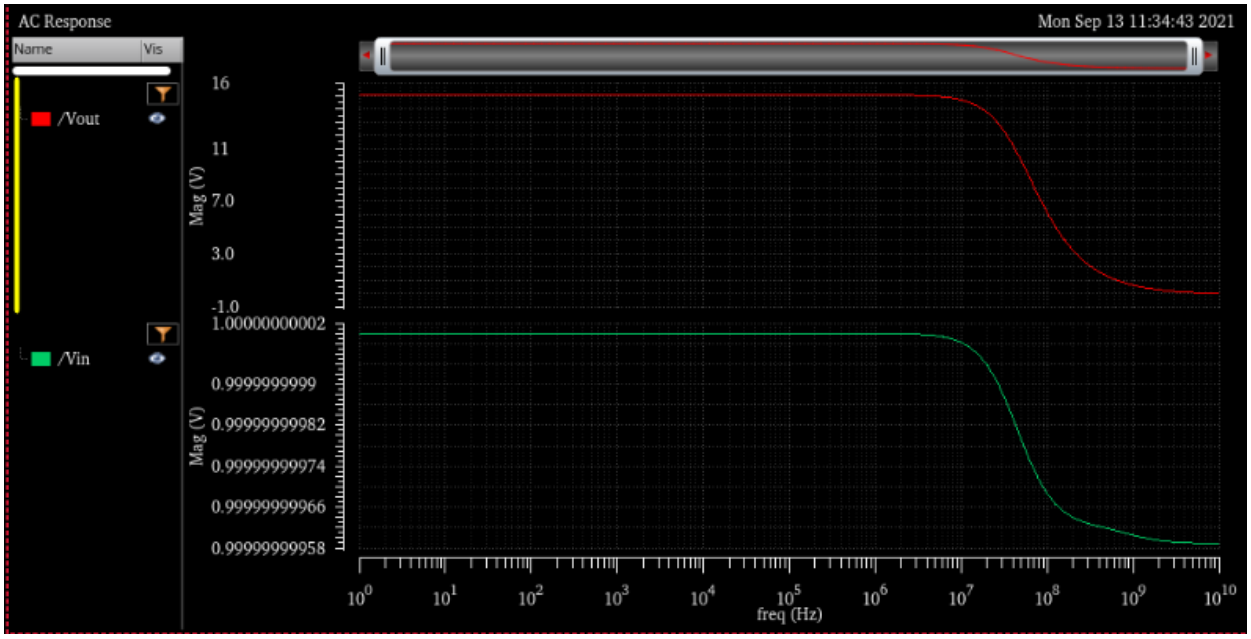



***Figure – 3.6: AC Analysis***

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.

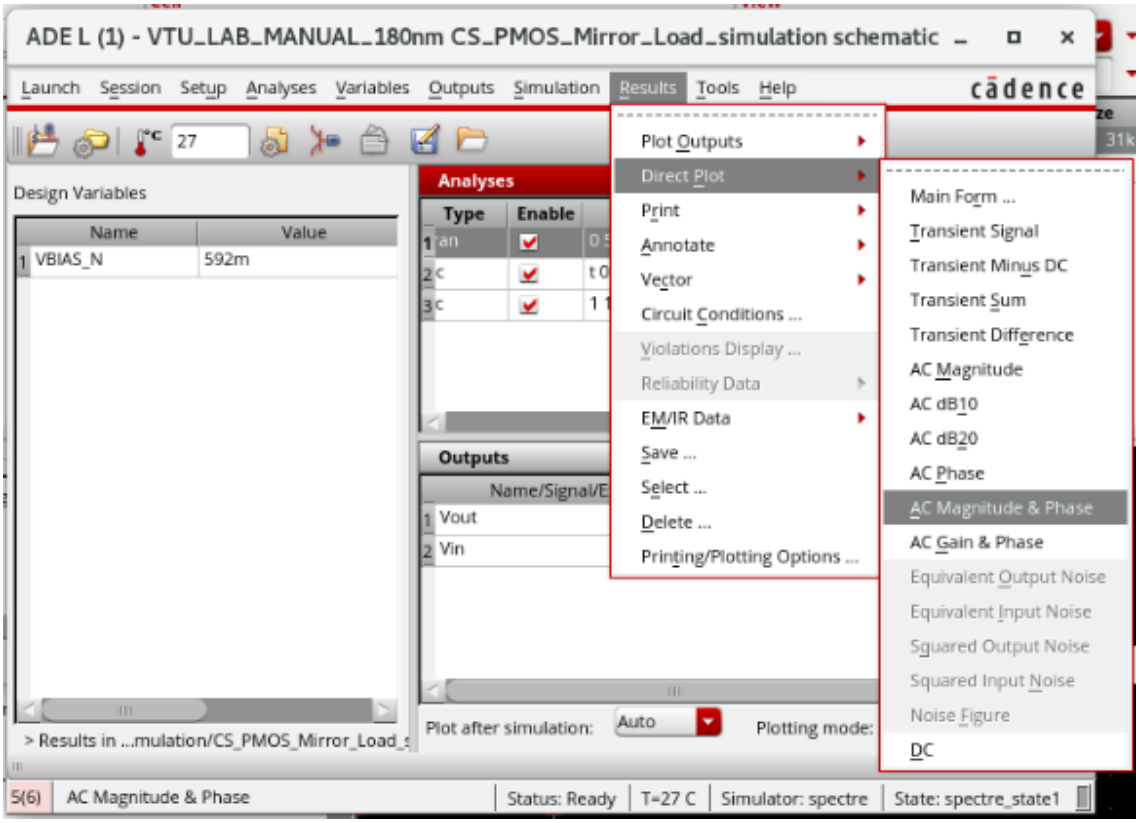



*Figure – 3.7: Results → Direct Plot → AC Magnitude & Phase*

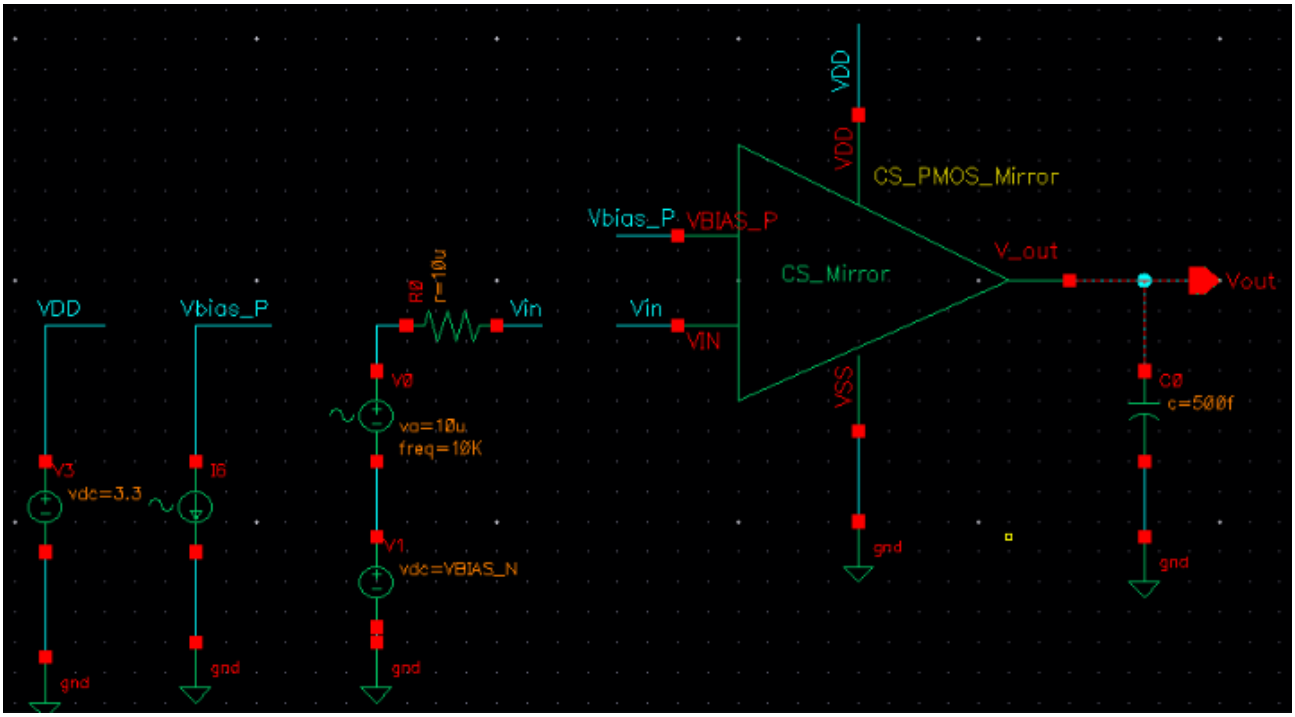



*Figure – 3.8:Selecting Output Net from the Test Schematic*

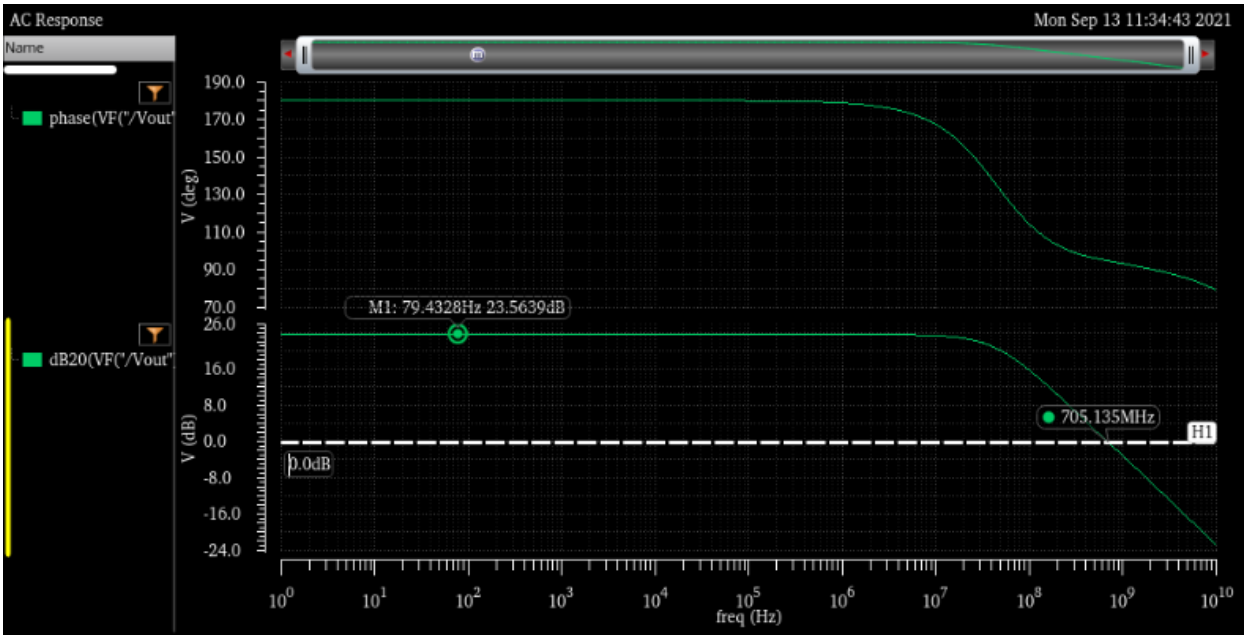



*Figure – 3.9: Gain and Phase plot*

**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.

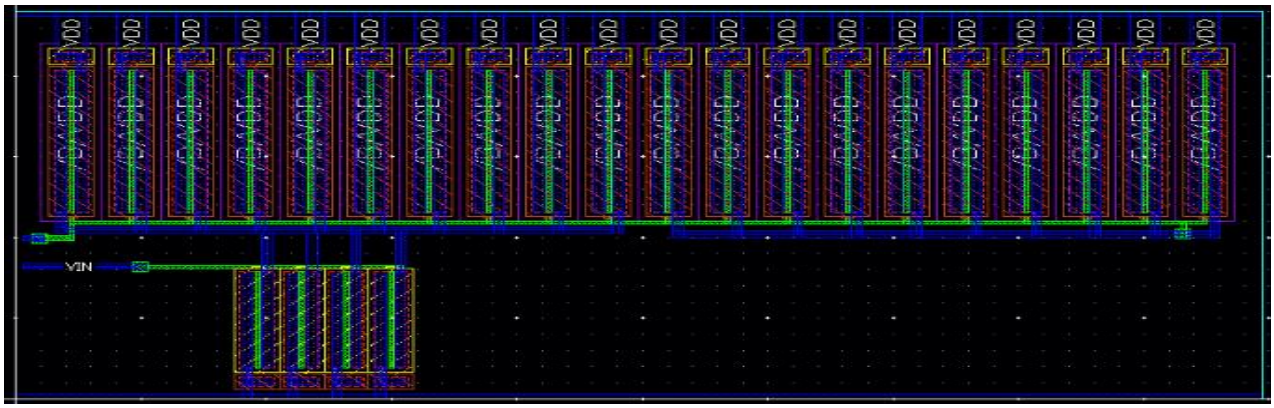



*Figure – 3.10: Layout forCommon Source Amplifier with PMOS Current Mirror Load*

**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.

## 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.

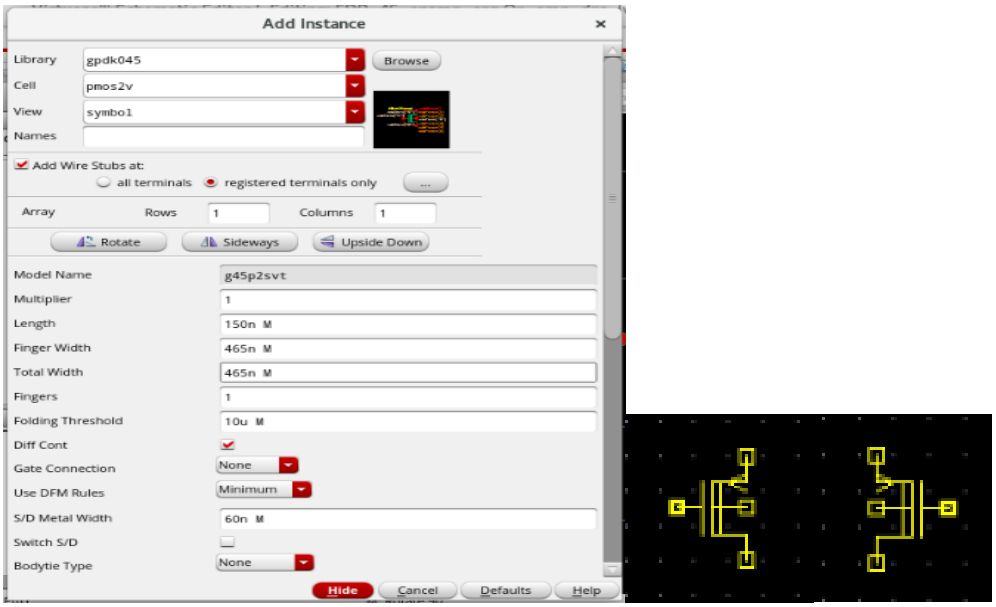



***Figure – 4.1: "Sideways" option to flip the Figure – 4.1(a):Figure – 4.1(b):***

***Transistors Before and After selecting "Sideways"***

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<br>Length, L = 150 n |
| gpdk045 | M3, M4 | nmos2v | Width, W = 490 n<br>Length, L = 150 n |
| gpdk045 | M5, M7 | nmos2v | Width, W = 1.09 u<br>Length, L = 150 n |
| gpdk045 | M2 | pmos2v | Width, W = 10 u<br>Length, L = 150 n |
| gpdk045 | M6 | nmos2v | Width, W = 6.88 u<br>Length, L = 150 n |
| gpdk045 | M8 | pmoscap2v | Calculated Parameter = Capacitance<br>Capacitance = 250.043f |

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

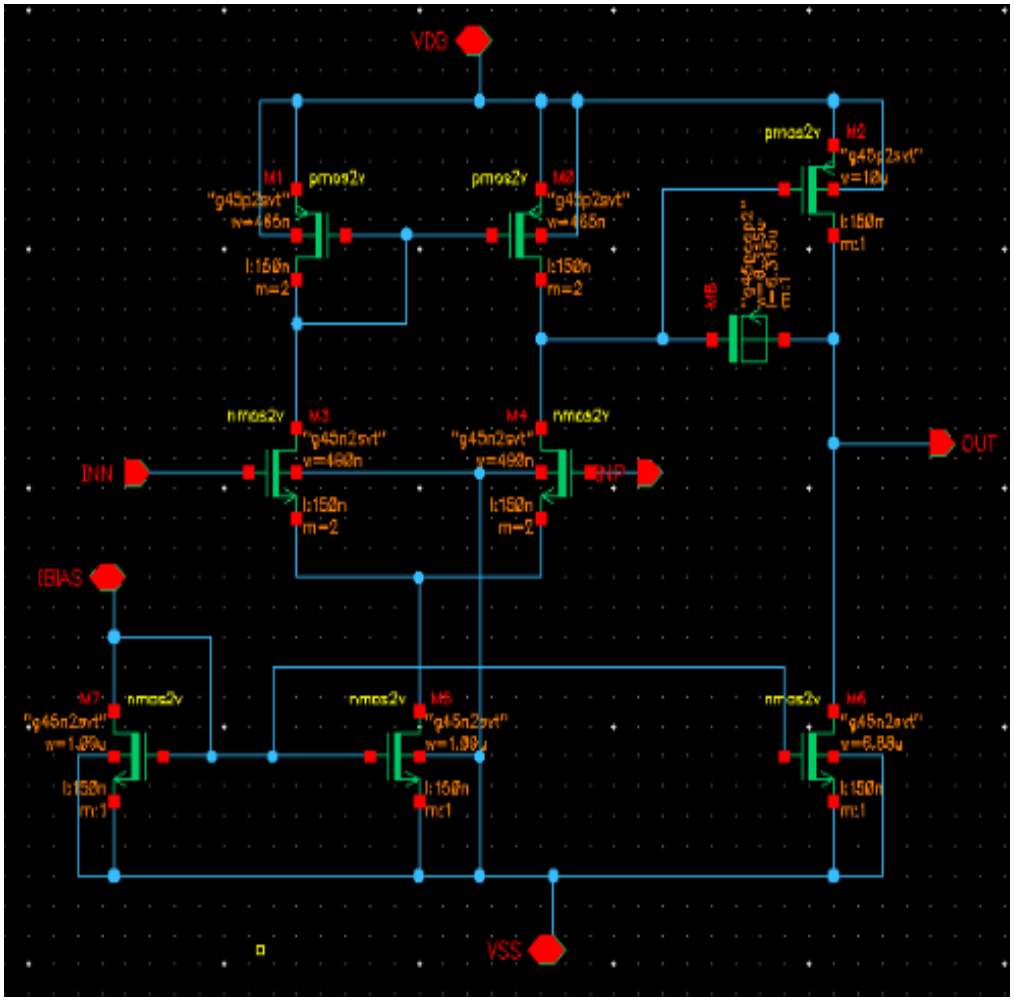



*Figure – 4.2: Schematic of 2 – Stage Operational Amplifier*

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

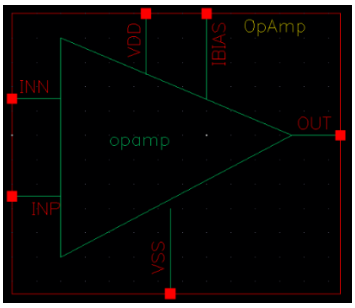



*Figure – 4.3: Symbol for 2 – Stage Operational Amplifier*

**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.

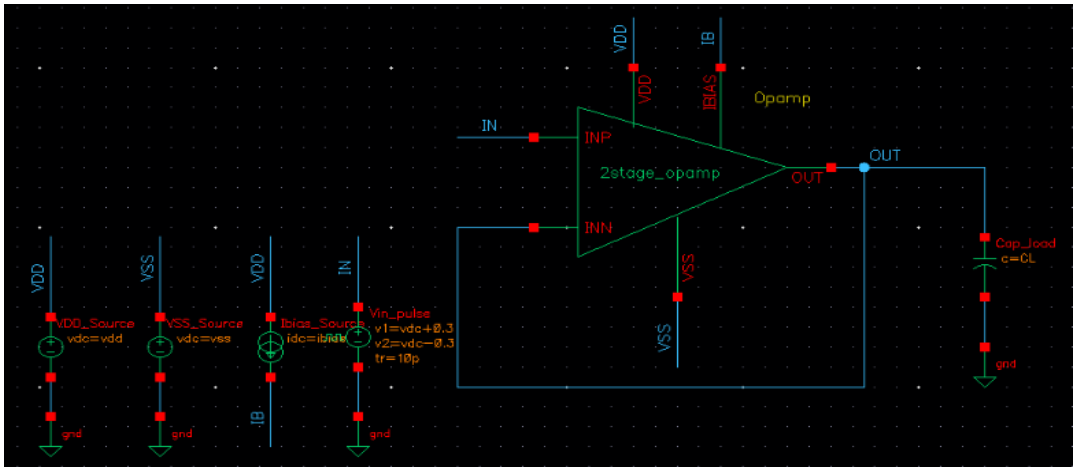



*Figure – 4.4: Test Schematic for 2 – Stage Operational Amplifier*

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.

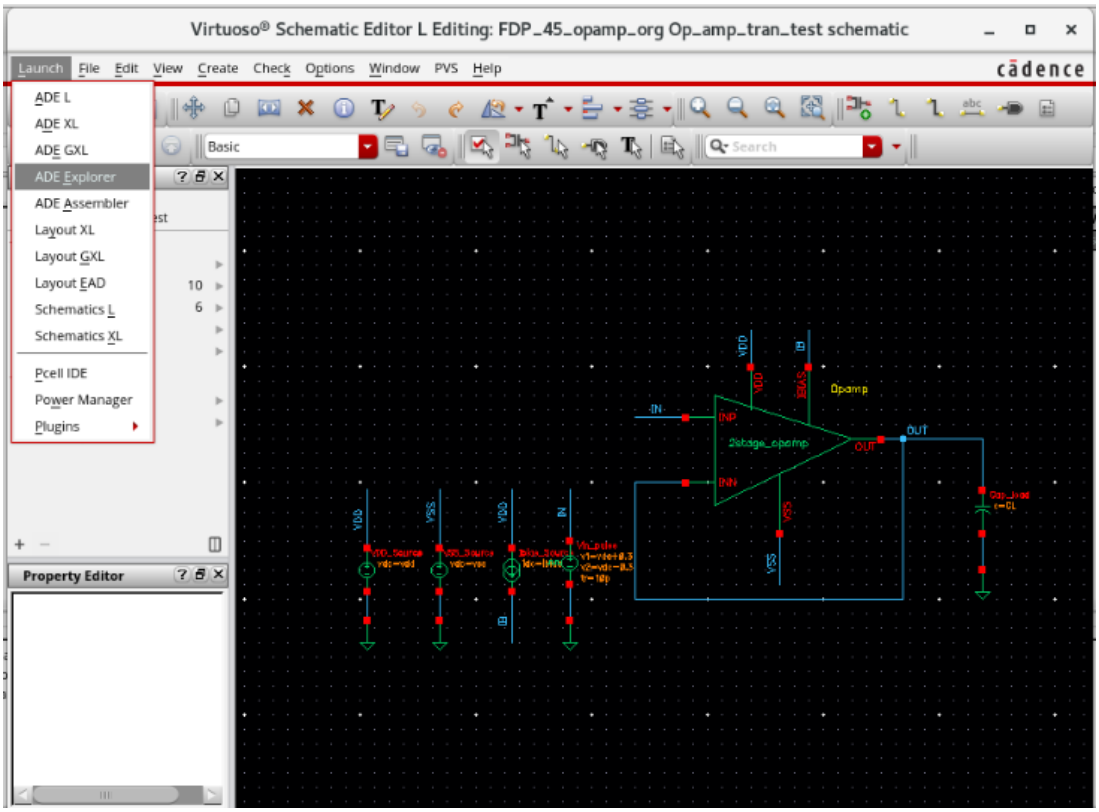



***Figure – 4.5: Launch → ADE Explorer***

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

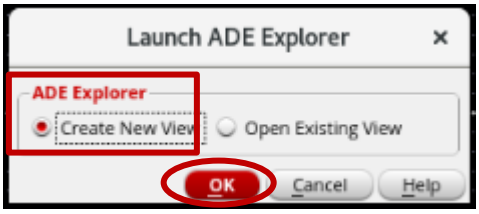



***Figure – 4.6: "Launch ADE Explorer" window***

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**".

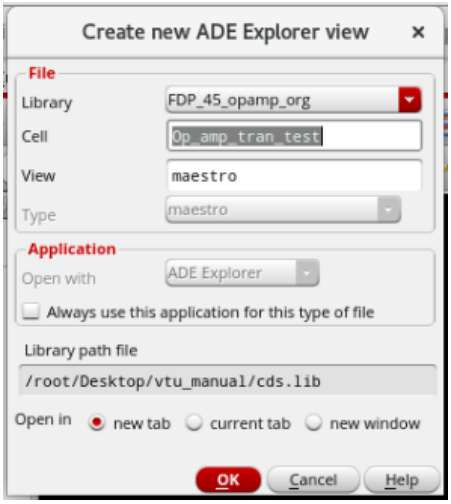



***Figure – 4.7: "Create new ADE Explorer view" window***

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

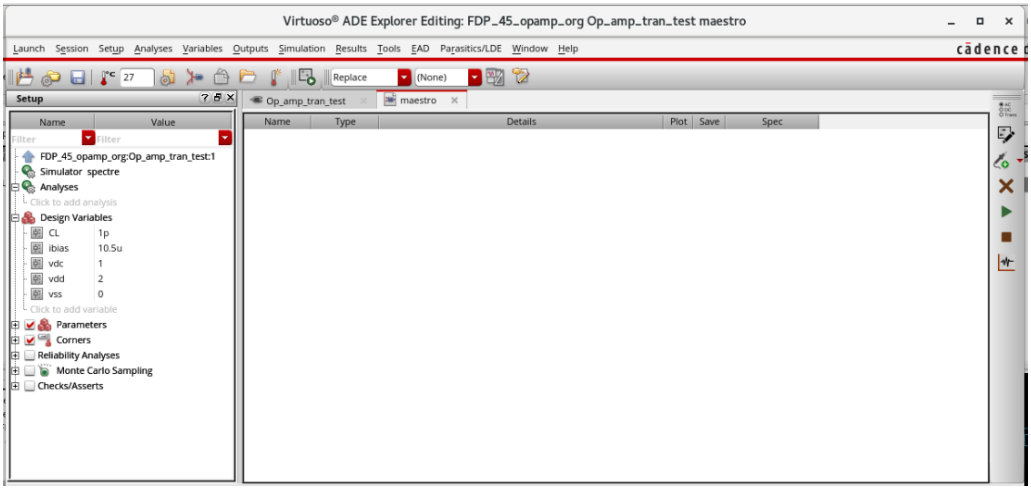



***Figure – 4.8: "Virtuoso ADE Explorer Editing" window***

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

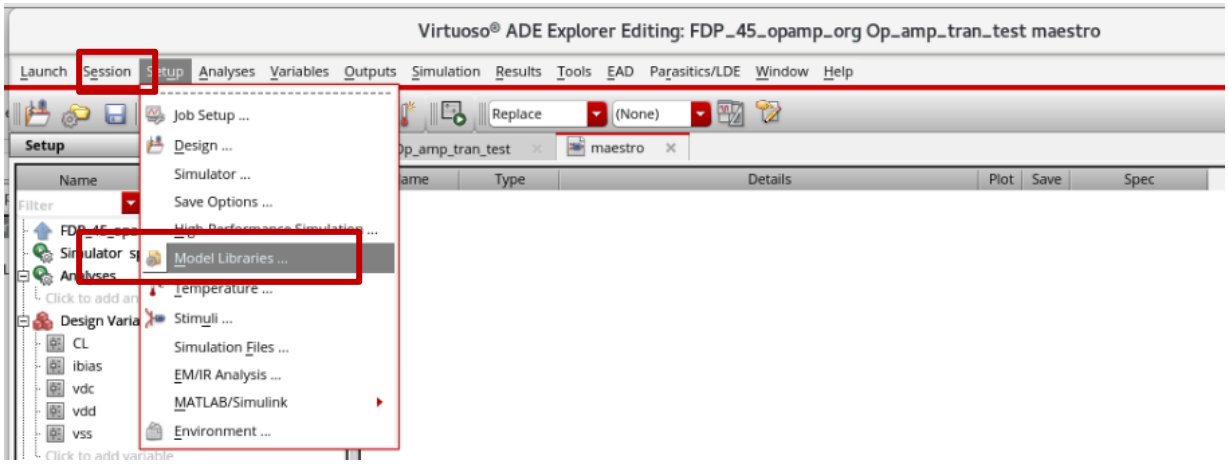



***Figure – 4.9: Setup → Model Libraries***

The "**spectre1: Model Library Setup**" window pops up as shown in Figure – 4.10. Select the respective "**.scs**" file and the process corner as "**tt**". Click on "**OK**"

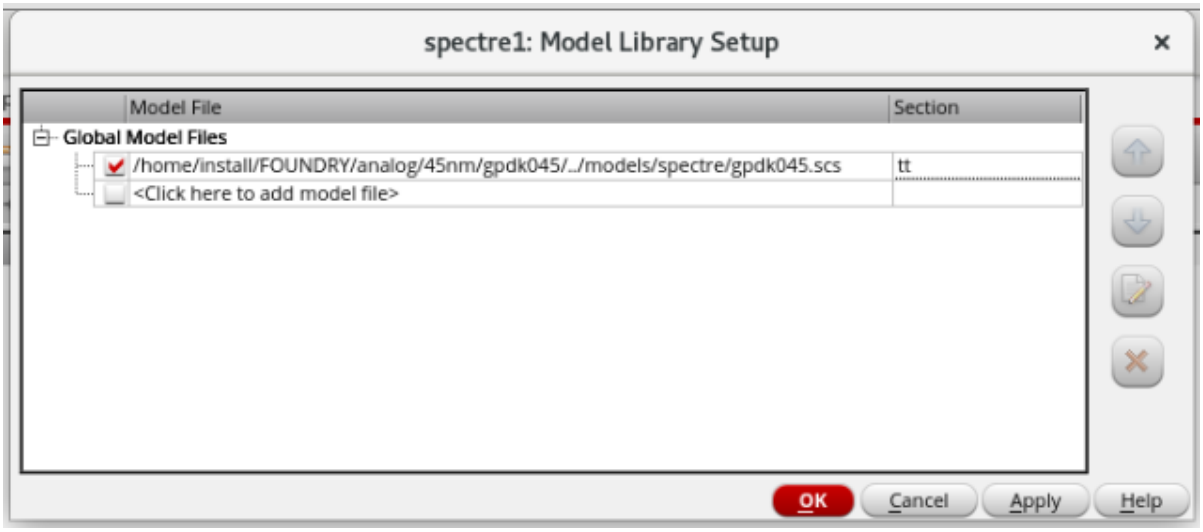



***Figure – 4.10: "spectre1: Model Library Setup" window***

To analyze the circuit through Transient Analysis and AC Analysis, select "**Click to add analysis**" just below the "**Analyses**" option in the "**Setup**" window as shown in Figure – 4.11.

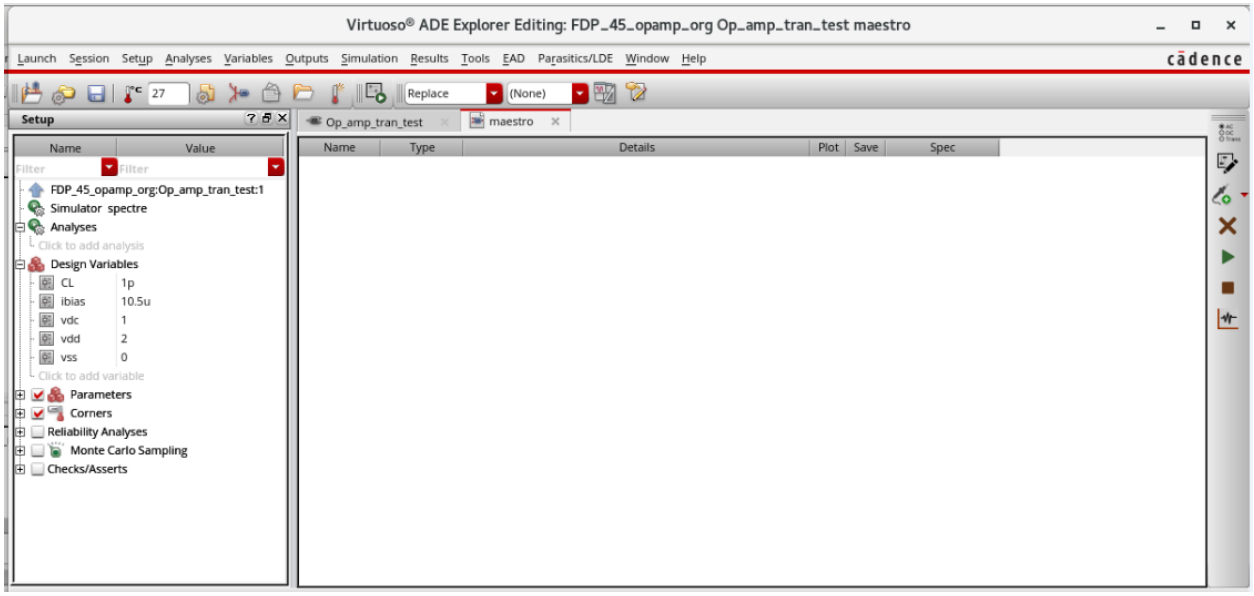



***Figure – 4.11: Analyses → Click to add analysis***

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.

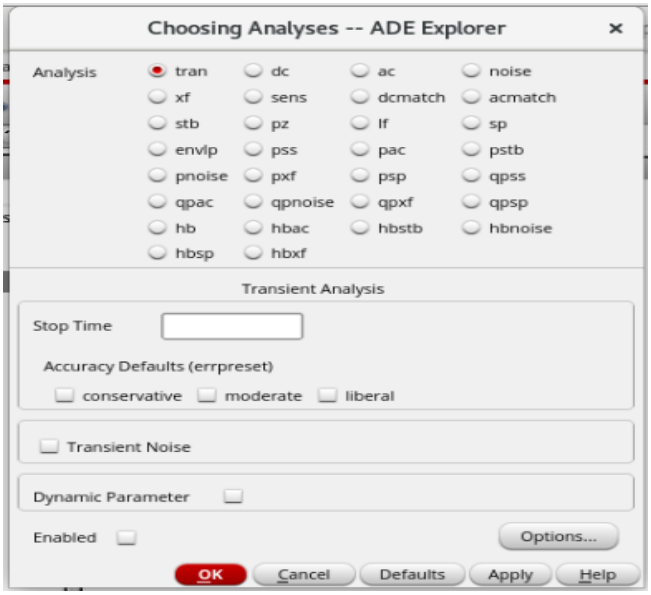



***Figure – 4.12: "Choosing Analyses – ADE Explorer" window***

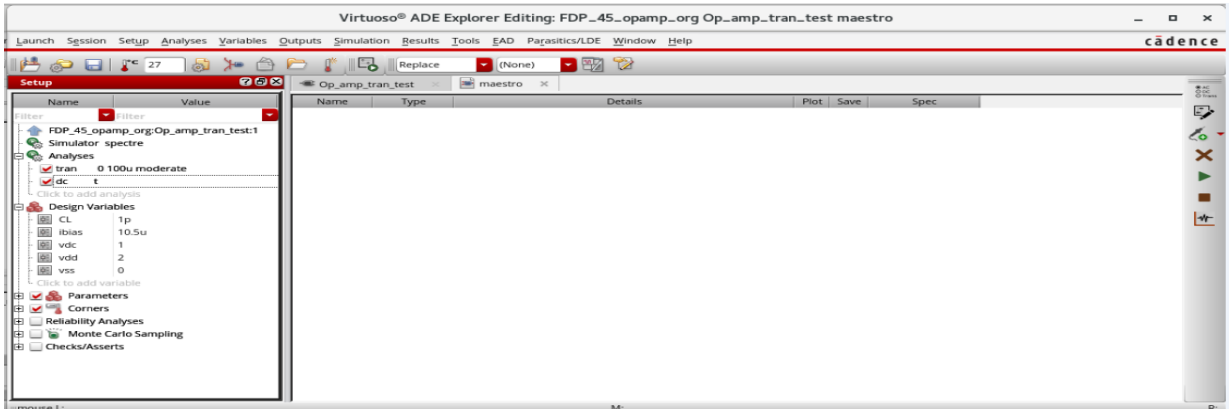



***Figure – 4.13: Updated ADE Explorer***

| 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.

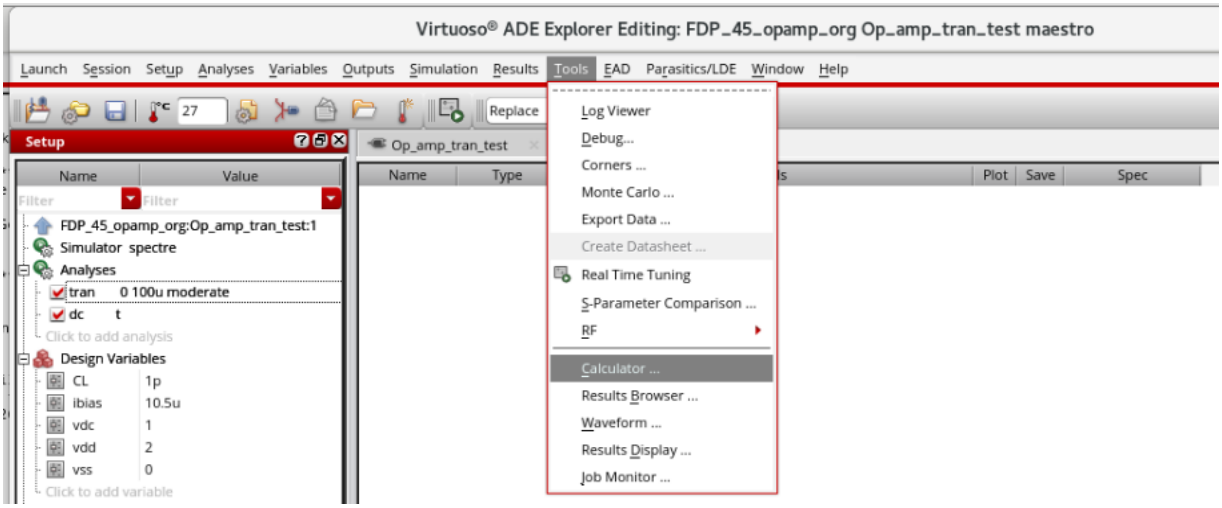



***Figure – 4.14: Tools → Calculator***

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

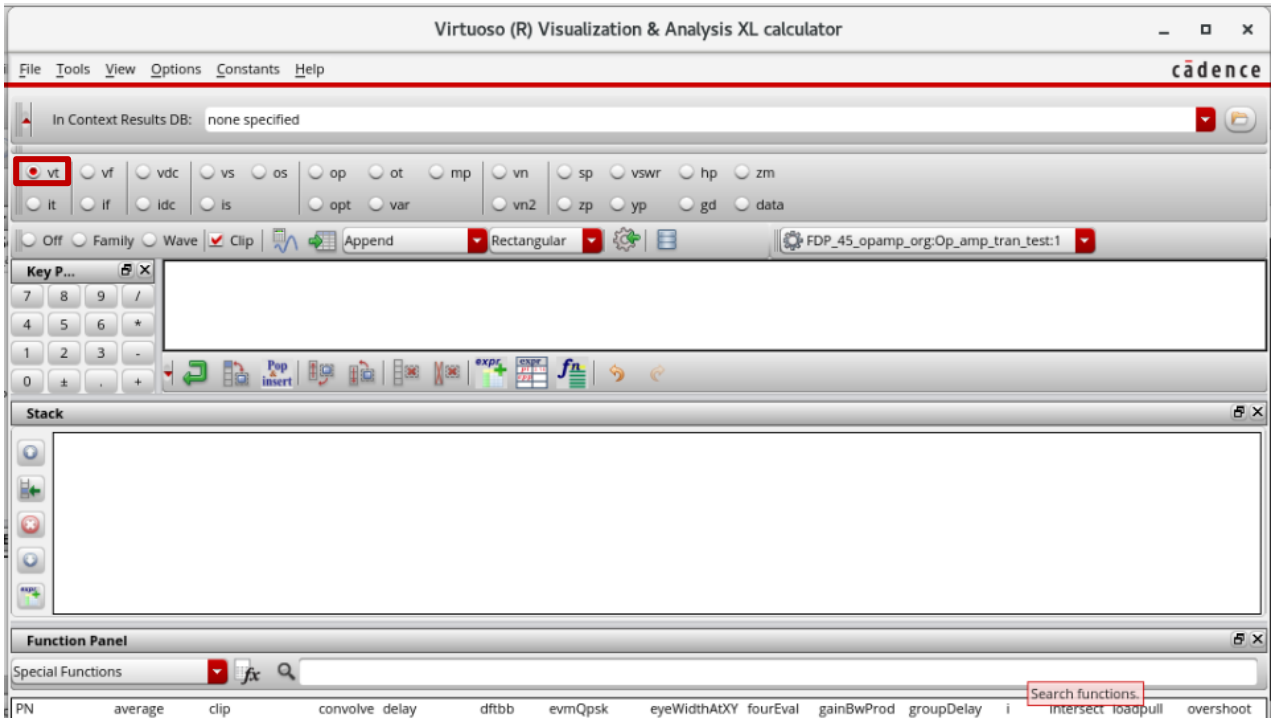



***Figure – 4.15: "Virtuoso Visualization & Analysis XL calculator" window***

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.

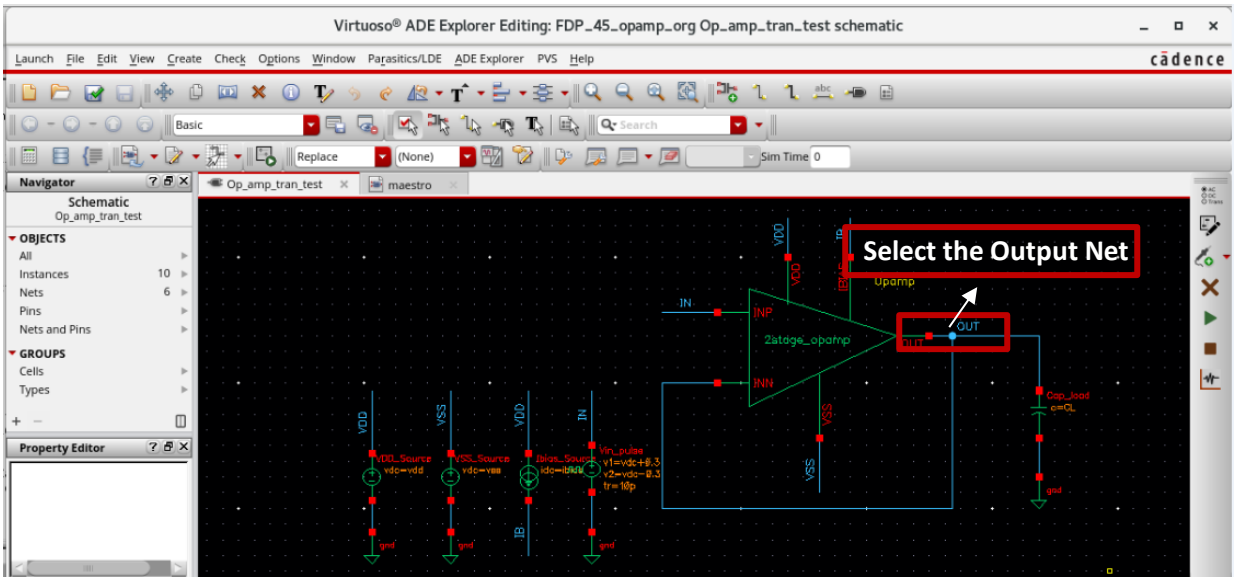



*Figure – 4.16: Test Schematic*

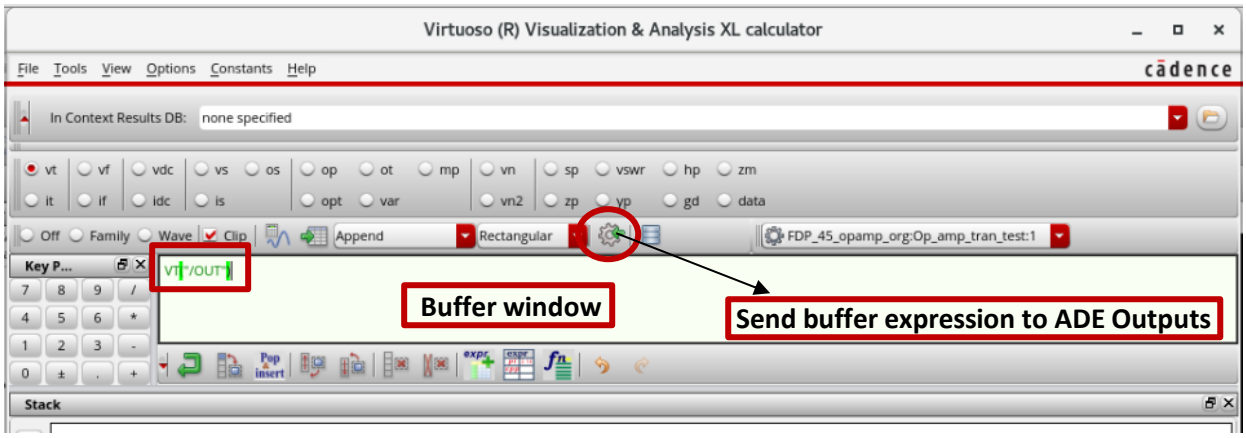



*Figure – 4.17: Updated Buffer window*

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.

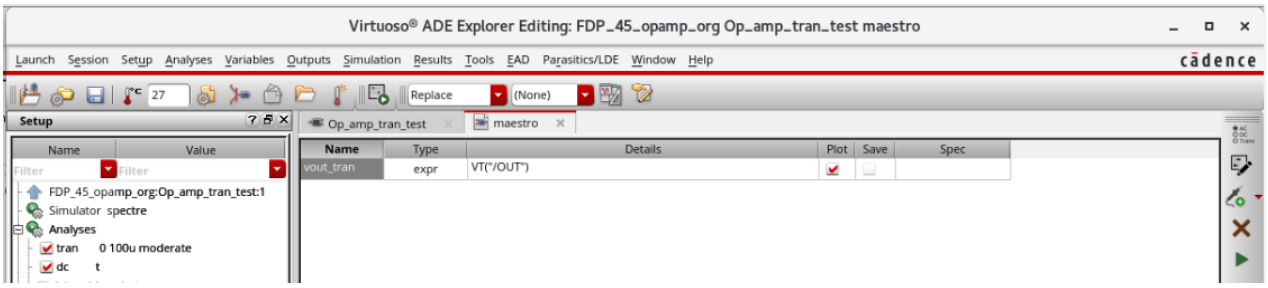



*Figure – 4.18: Updated ADE Explorer*

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.

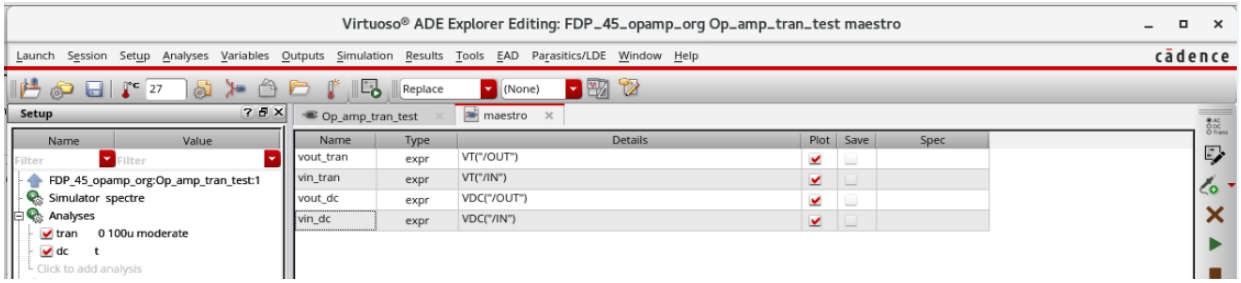



***Figure – 4.19: Updated ADE Explorer***

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.

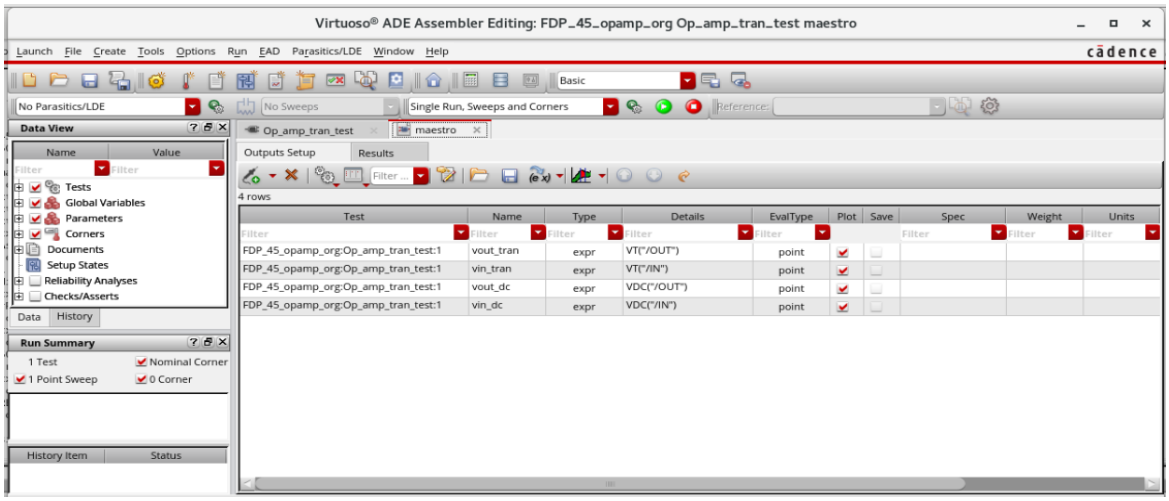



***Figure – 4.20: ADE Assembler invoked***

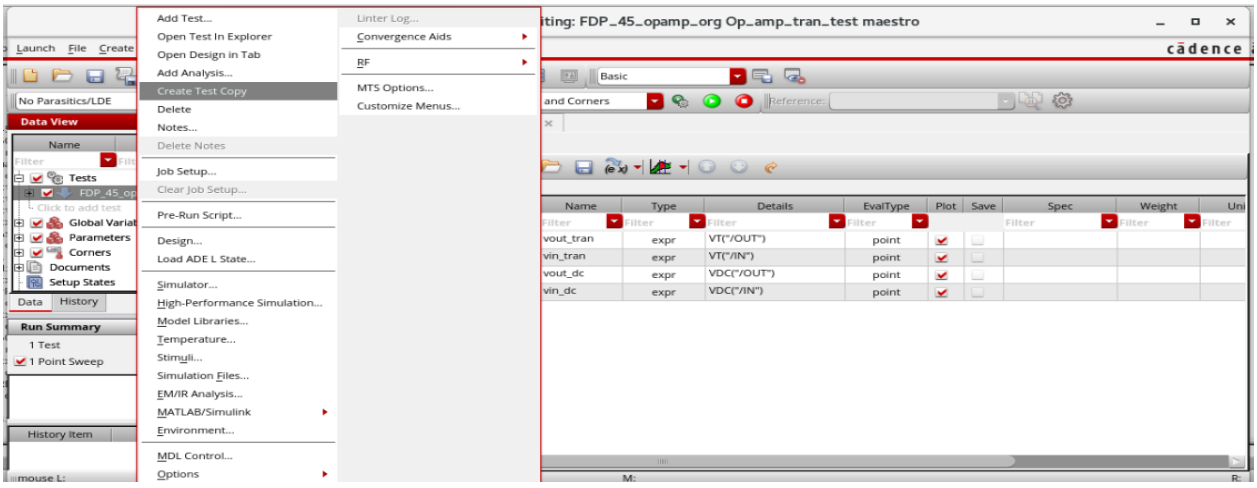



***Figure – 4.21: Create Test Copy***

Expand "**Tests**" anduse 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.

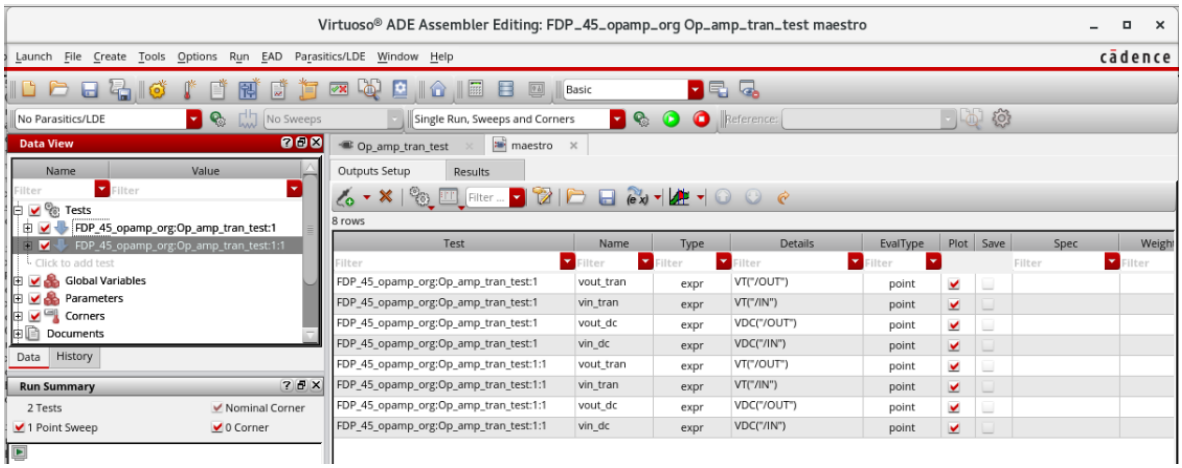



***Figure – 4.22: Copied Test***

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

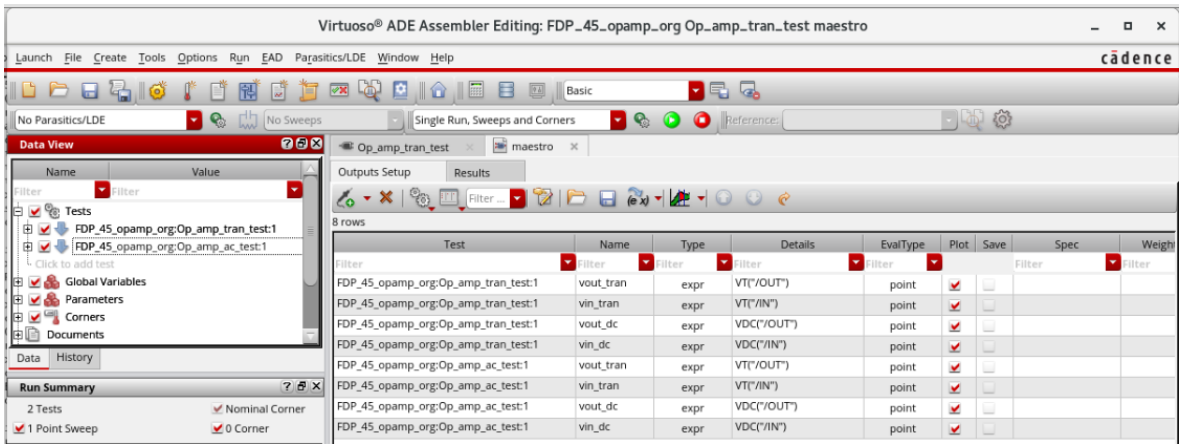



***Figure – 4.23: Renamed Test***

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.

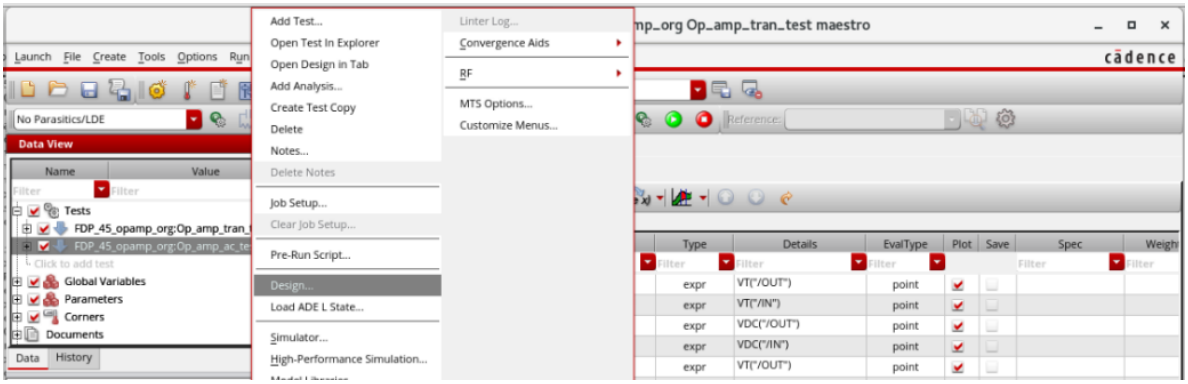



*Figure – 4.24: Select "Design" option*

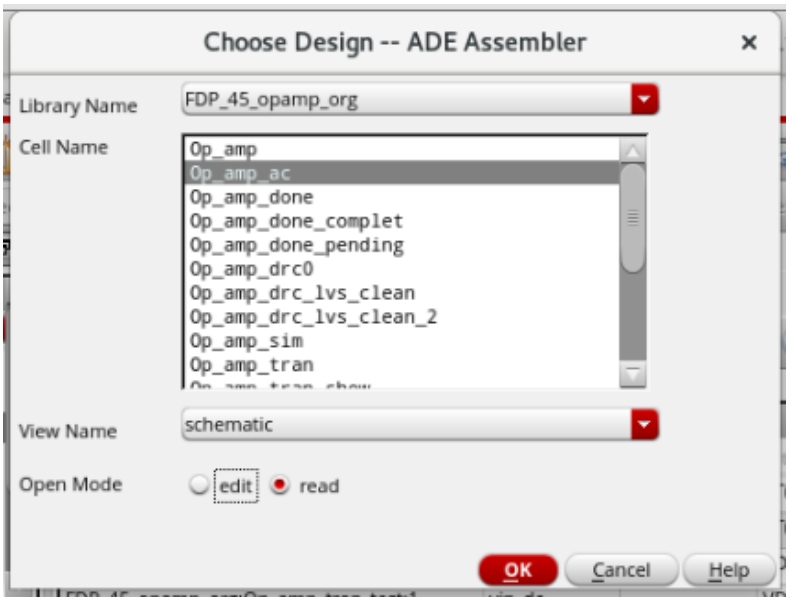



*Figure – 4.25: Select the "Op_amp_ac" Cell Name*

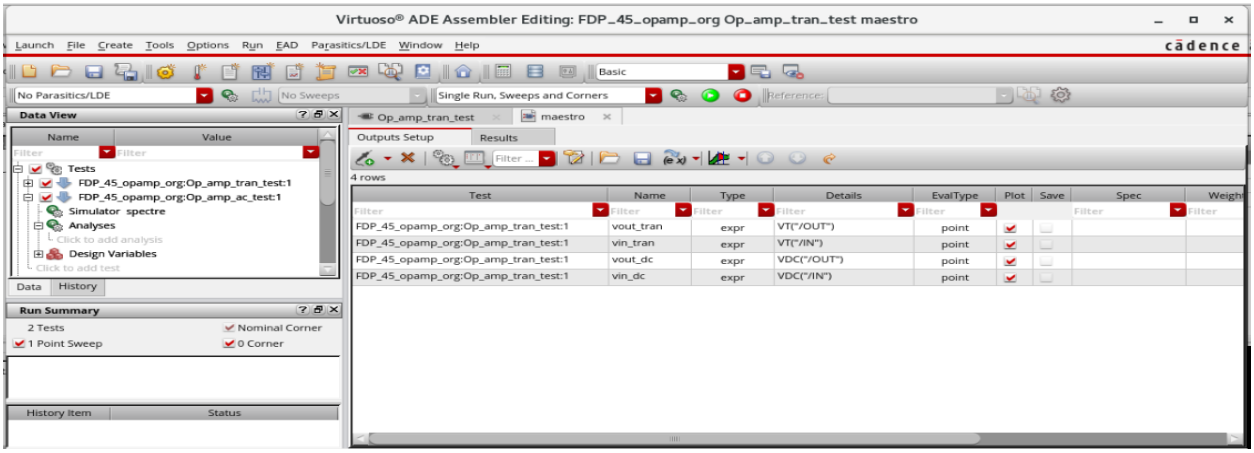



*Figure – 4.26: Updated ADE Assembler*

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.

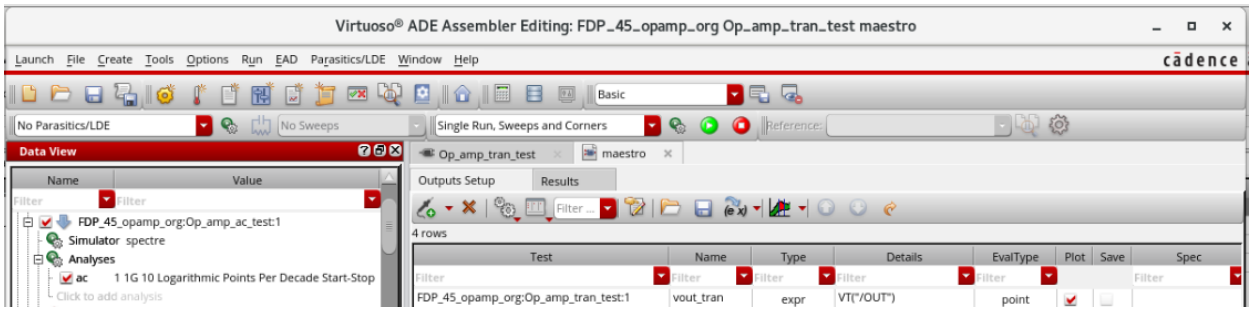



***Figure – 4.28: Updated ADE Assembler with "ac" analysis***

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.

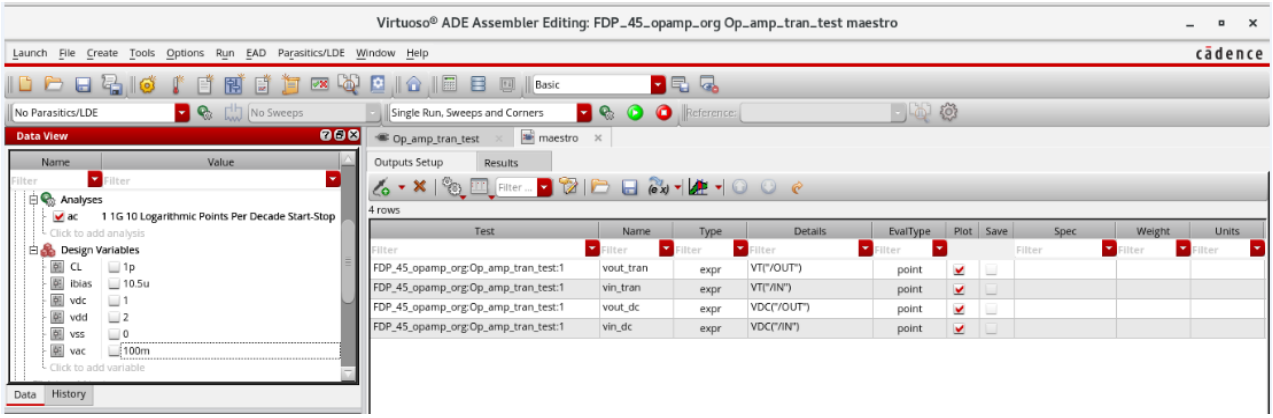



***Figure – 4.29: Updated Design Variables***

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

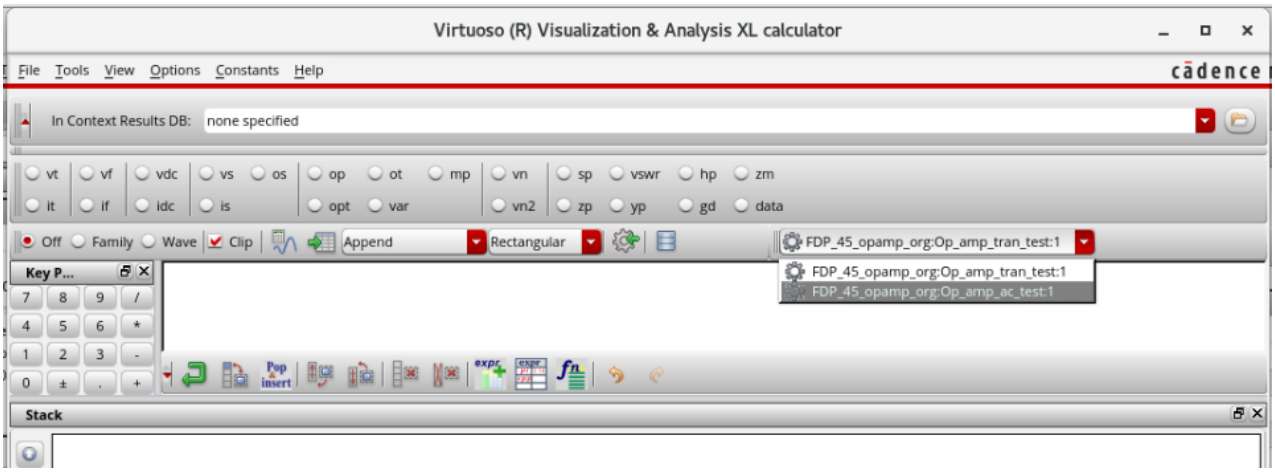



***Figure – 4.30: Selecting "ac" test from calculator***

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.

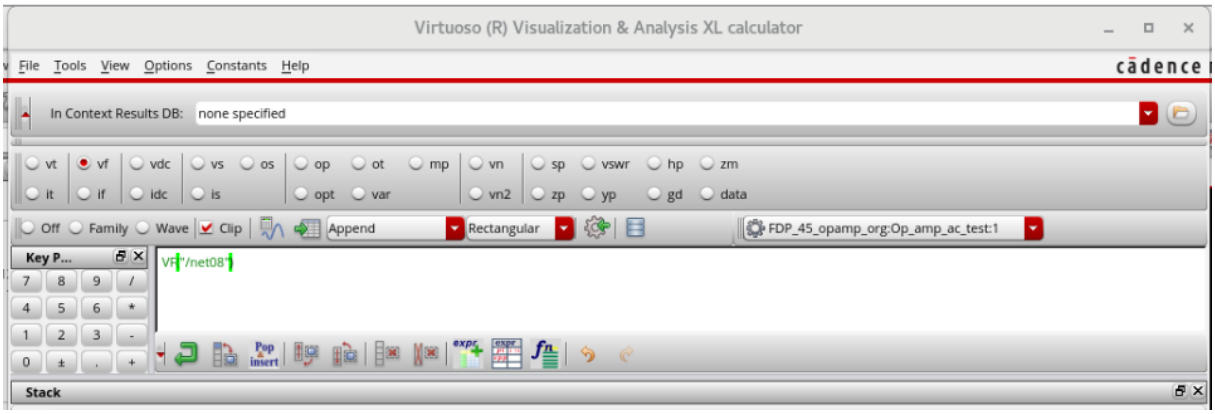



*Figure – 4.31: Updated Buffer after "vf" and output net selection*

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

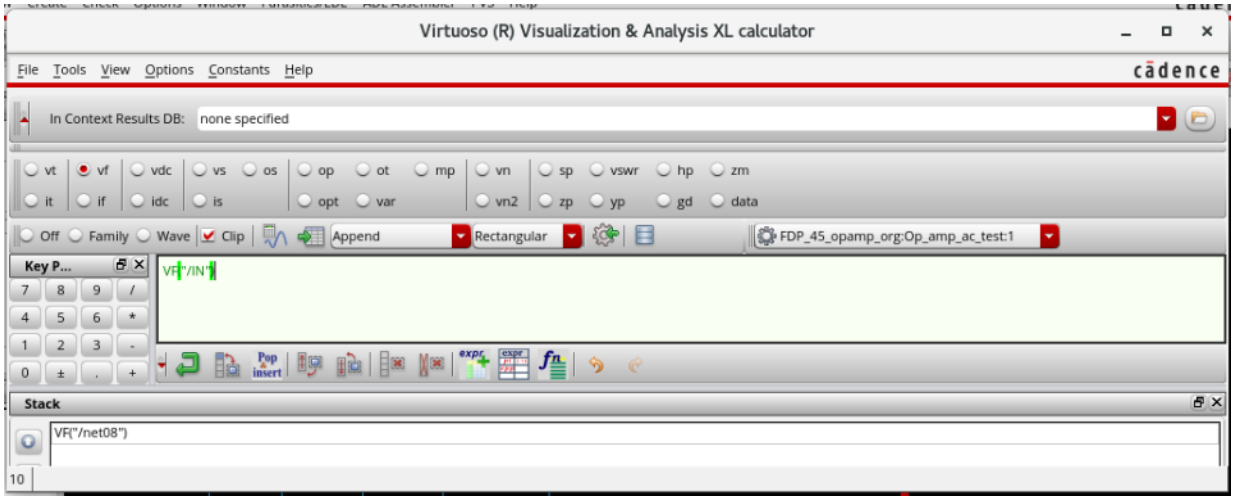



*Figure – 4.32: Updated Buffer and Stack*

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

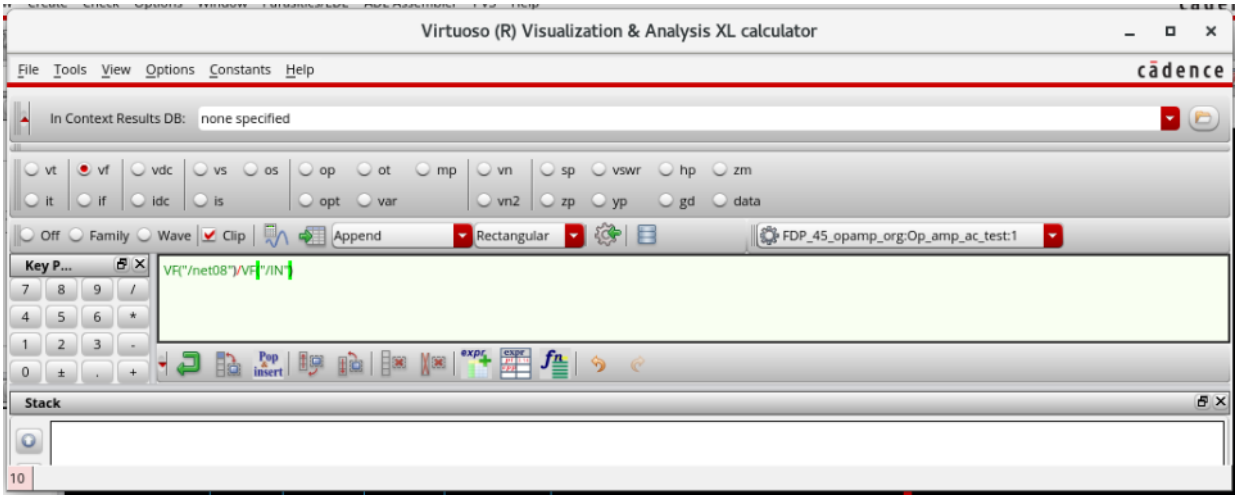


*Figure – 4.33: Updated Buffer and Stack*

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

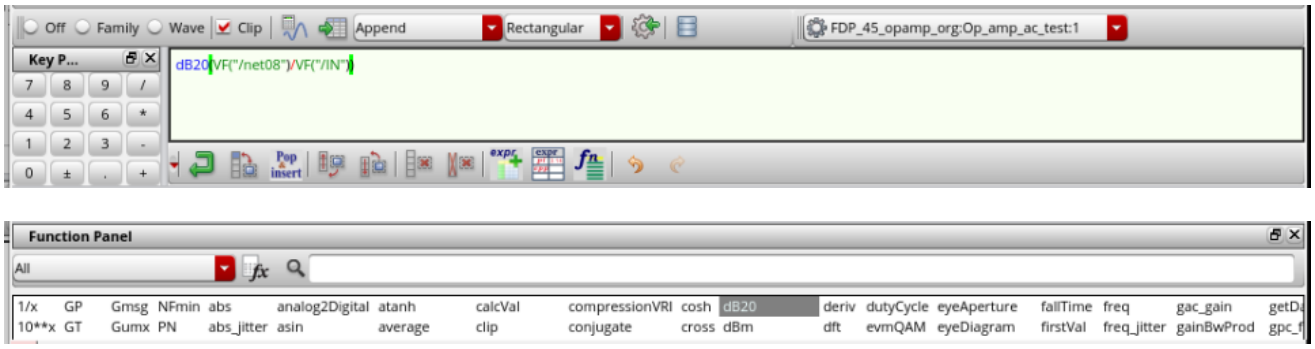



*Figure – 4.34: Updated Buffer after selecting "dB20"*

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.

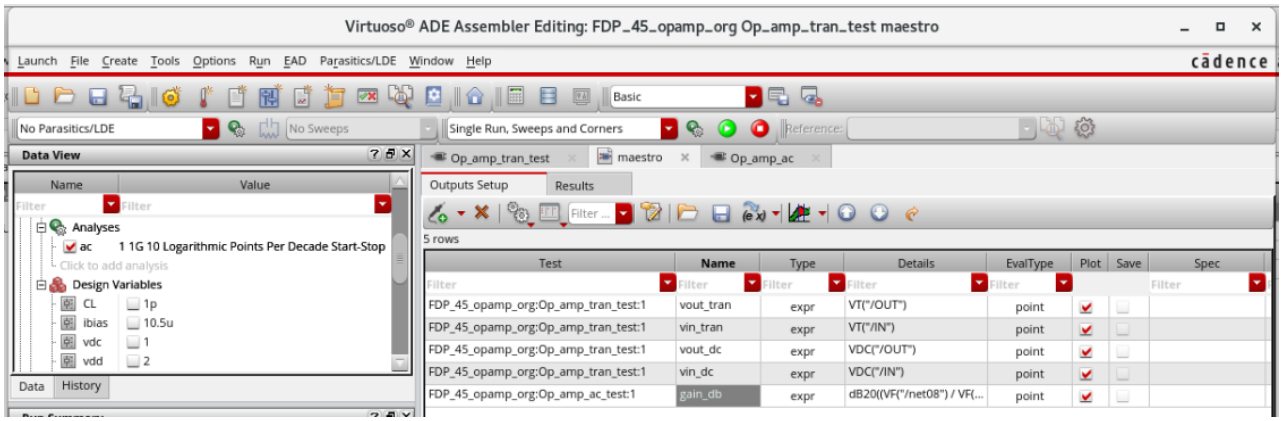



*Figure – 4.35: Expression from the Calculator*

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

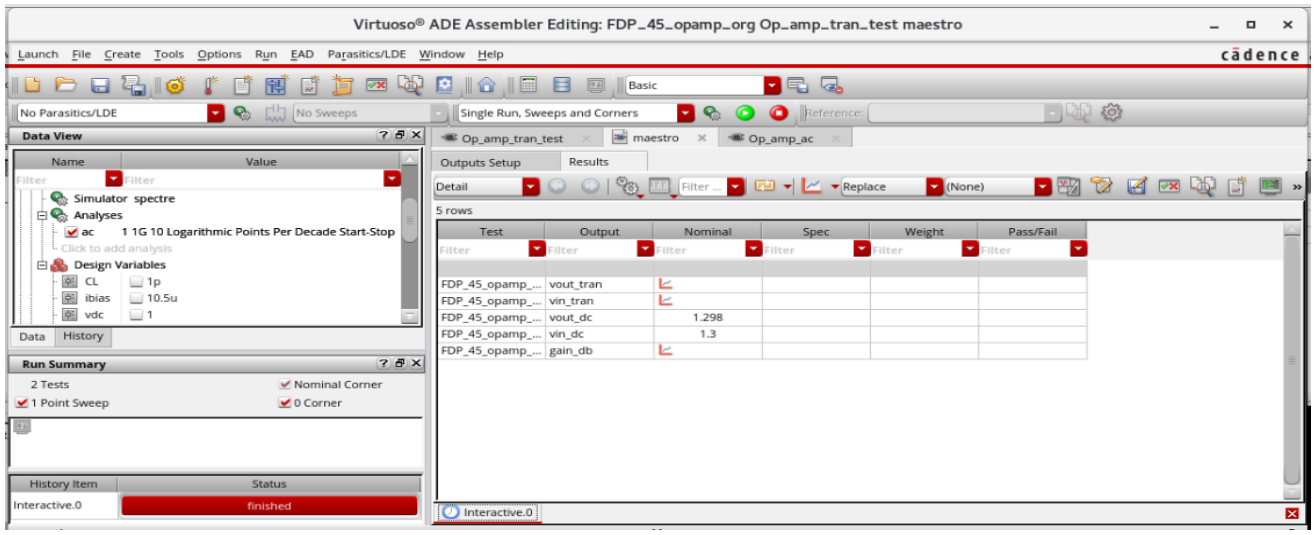



***Figure – 4.36: ADE Assembler after simulation***

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.

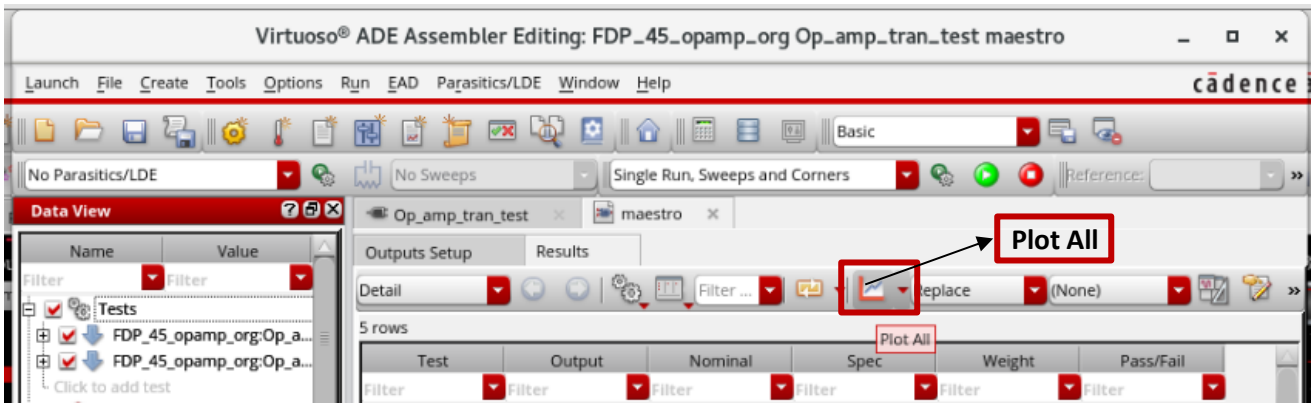


*Figure – 4.39: "Plot All" option*

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

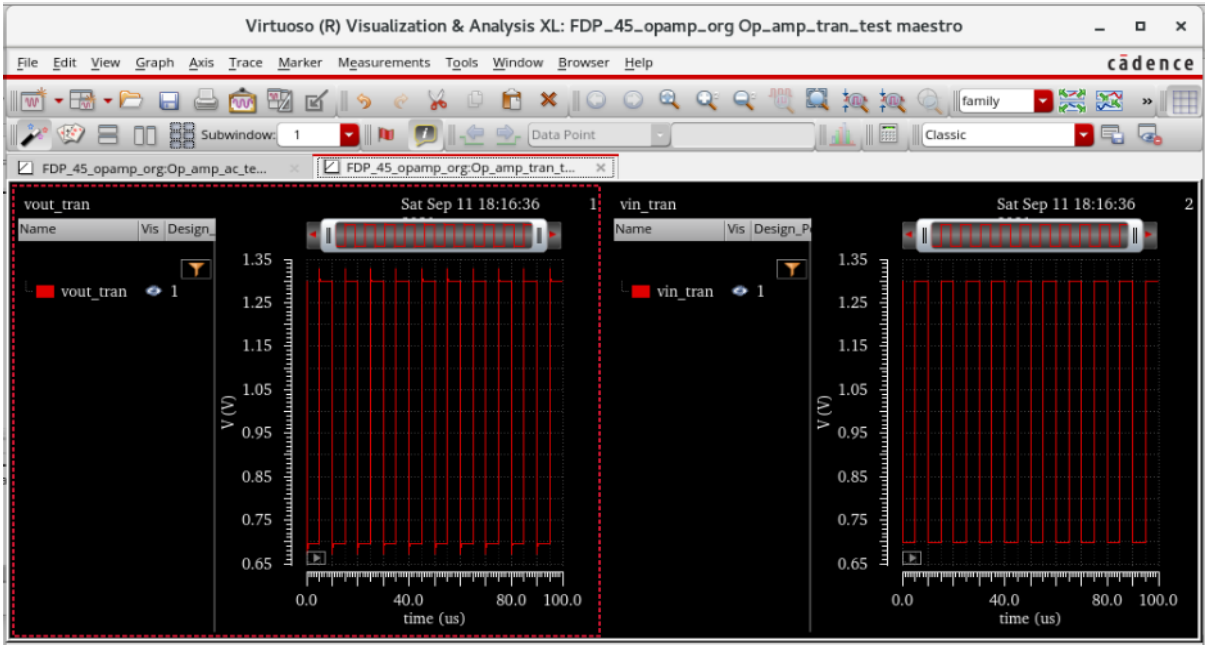



*Figure – 4.40: Plotted Waveforms*

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

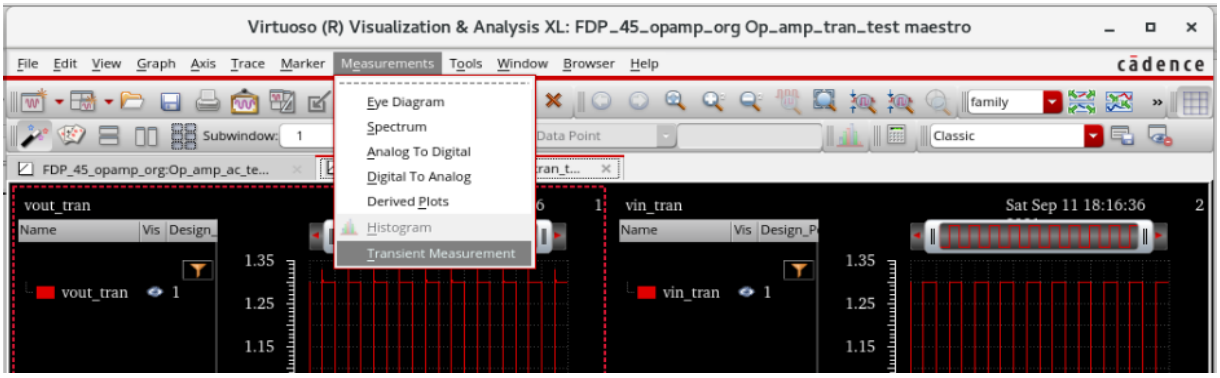



*Figure – 4.41: Measurements → Transient Measurement*

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.

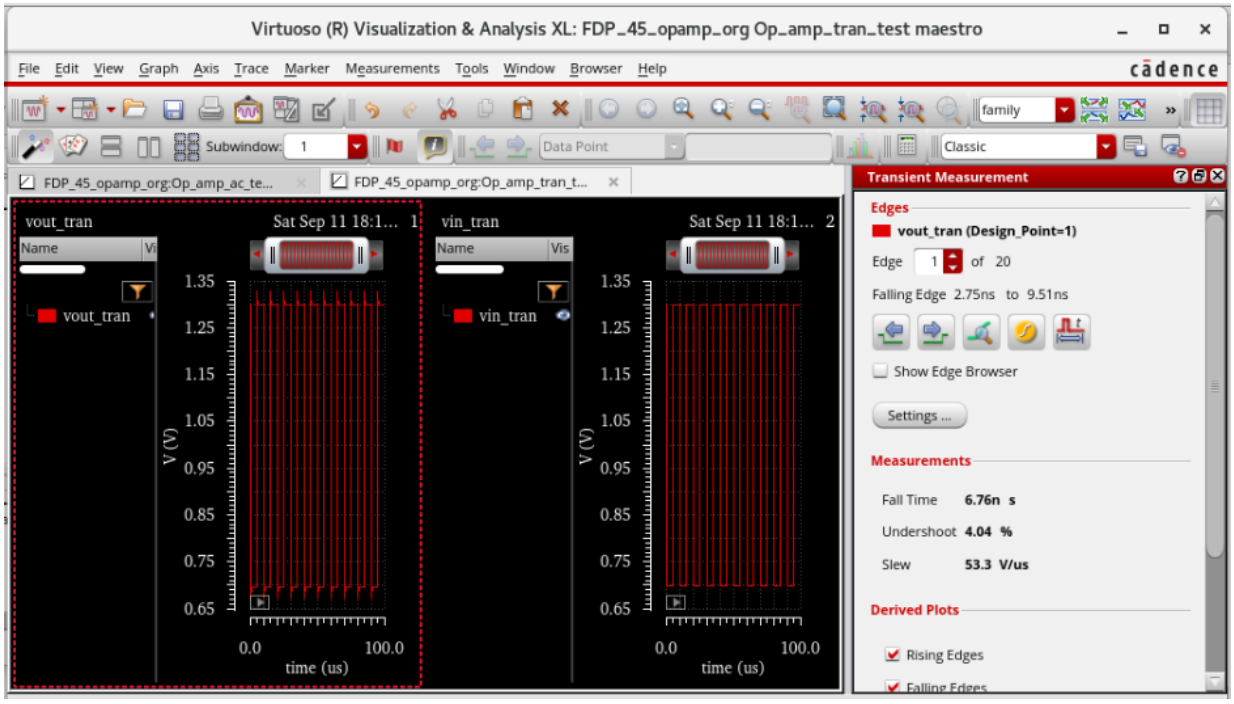



***Figure – 4.42: Transient Measurement tab***

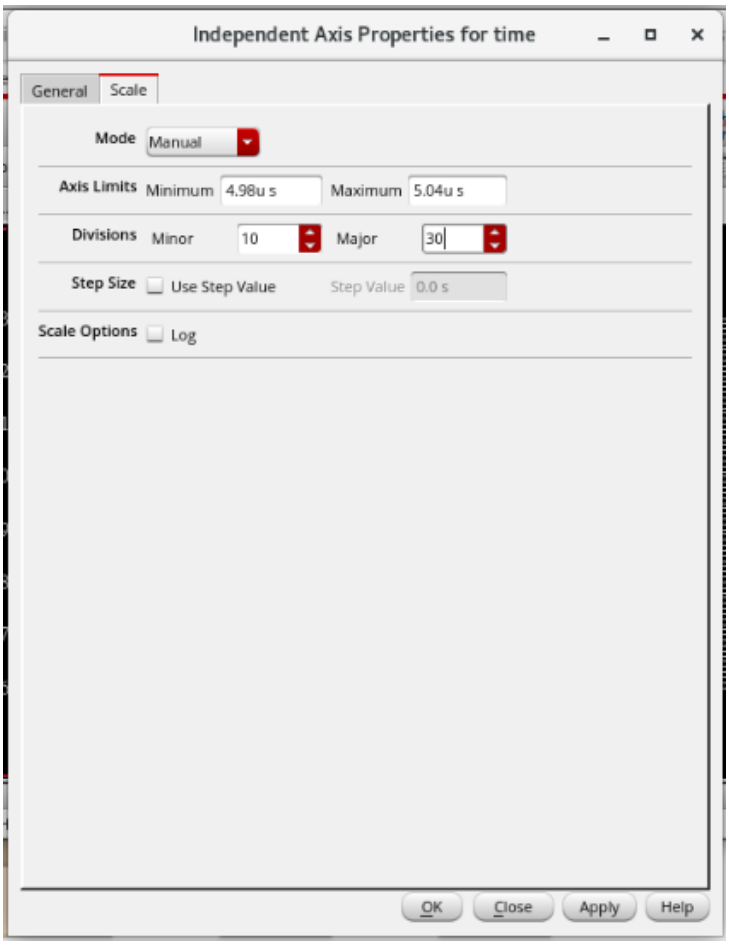



***Figure – 4.43: "Independent Axis Properties for time" window***

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.

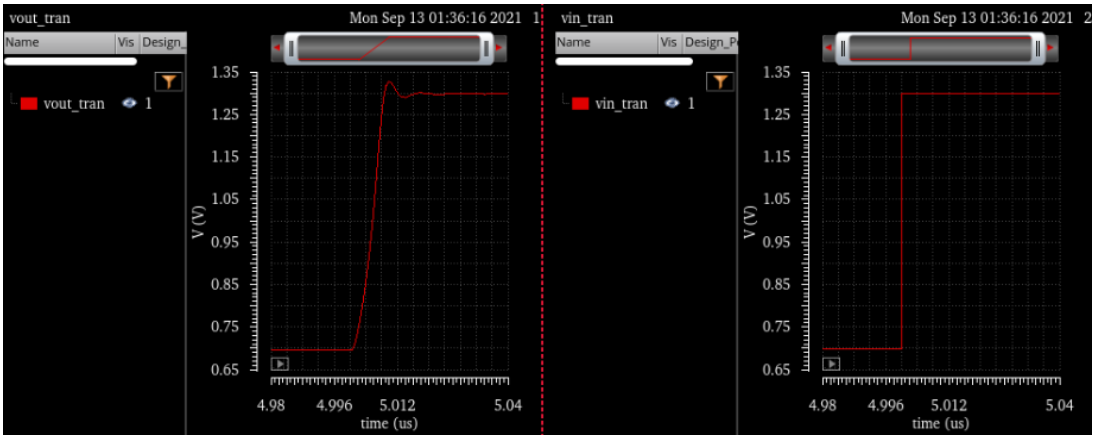



***Figure – 4.44: Isolated Waveforms***

Use left mouse click and drag and drop to combine the waveforms as shown in Figure – 4.45.

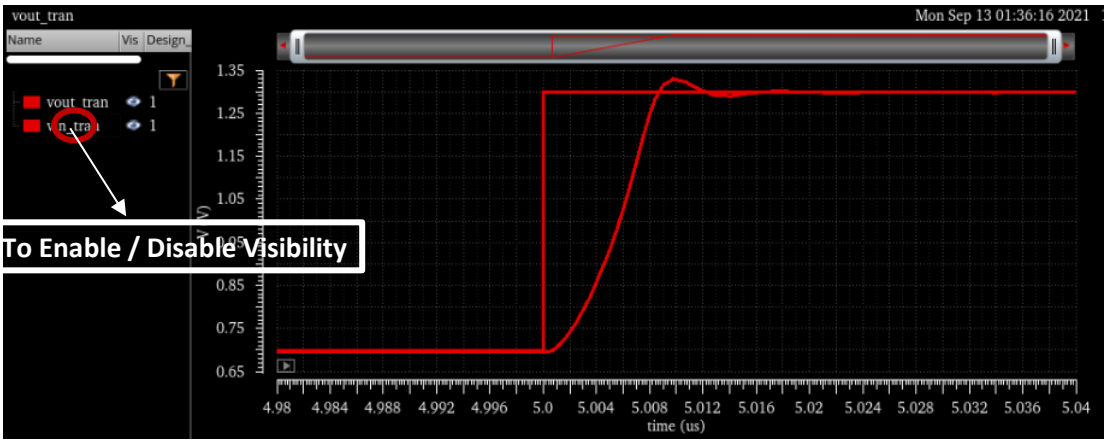



***Figure – 4.45: Combined Waveforms***

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

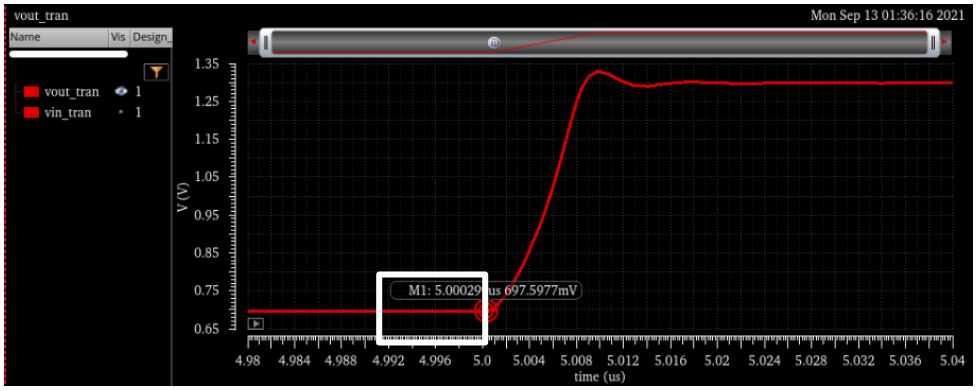



***Figure – 4.46: Waveform with Marker***

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

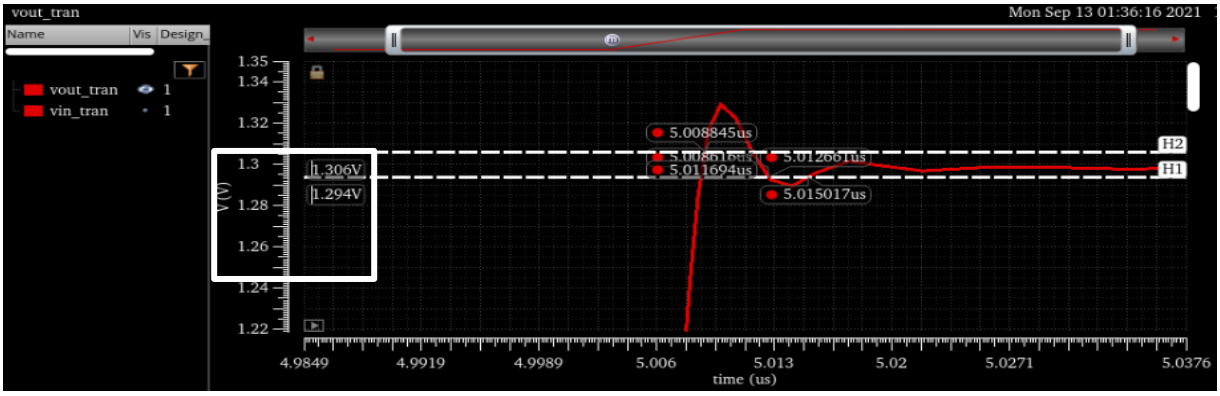



*Figure – 4.47: Horizontal cursors at 1.294 V and 1.306 V*

Use the zooming options to zoom-in and zoom-out as and when required. Setup a marker on the lower horizontal cursor as shown in Figure – 4.48.

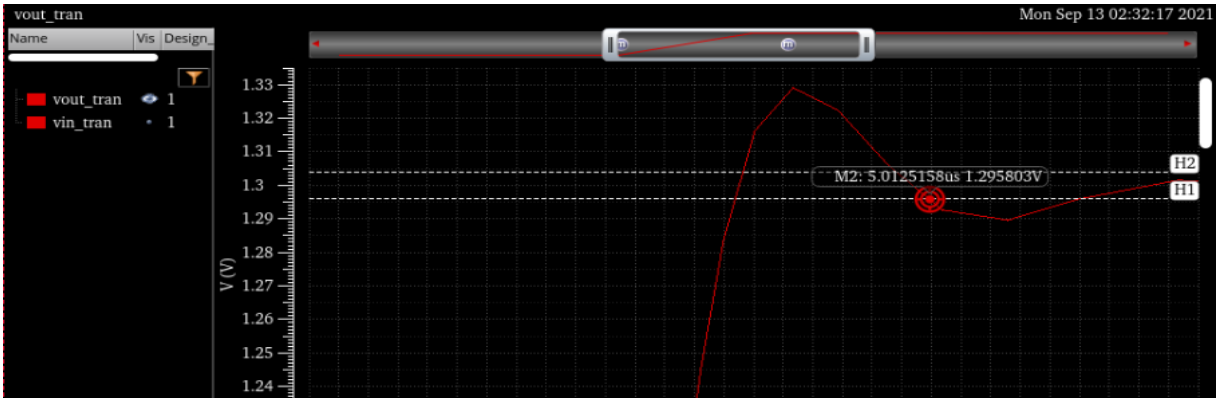



*Figure – 4.48: Marker on the 2nd horizontal cursor*

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 as shown 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 the cursor with the AC Analysis curve gives the Unity Gain Bandwidth.

The **Unity Gain Bandwidth** is measured as**84.51M Hz**.

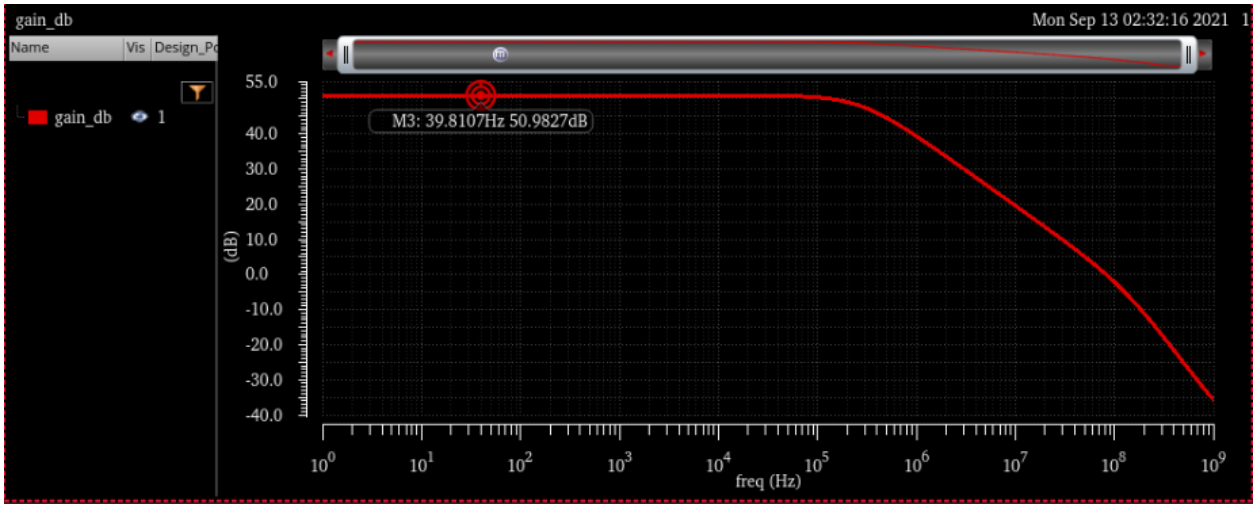



*Figure – 4.49: DC Open Loop Gain – 50.98 dB*

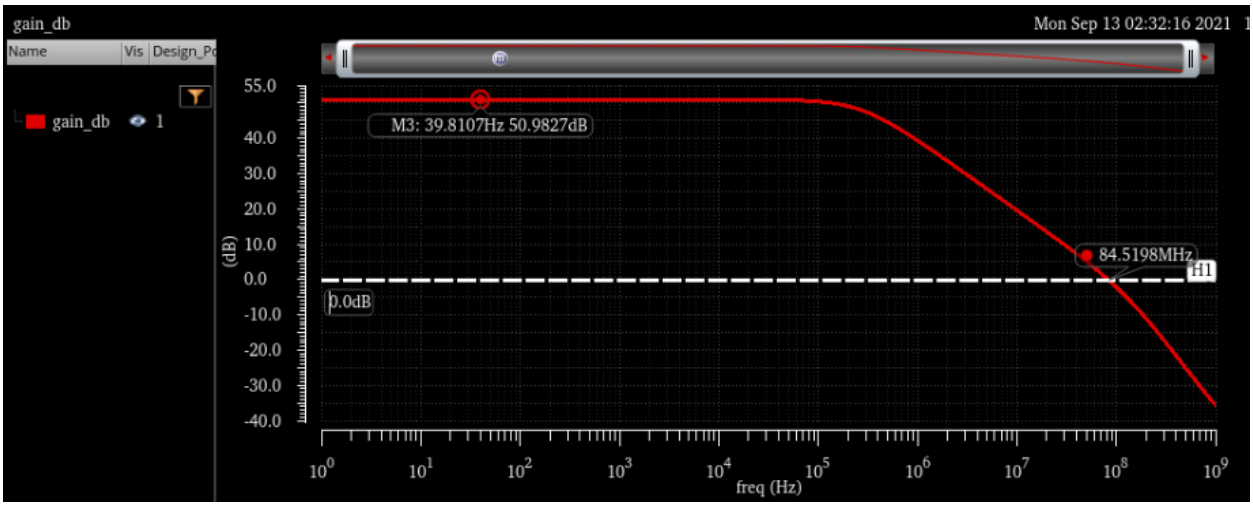



*Figure – 4.50: Unity Gain Bandwidth – 84.51M Hz*

## GENERATING THE EXPRESSIONS:

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

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

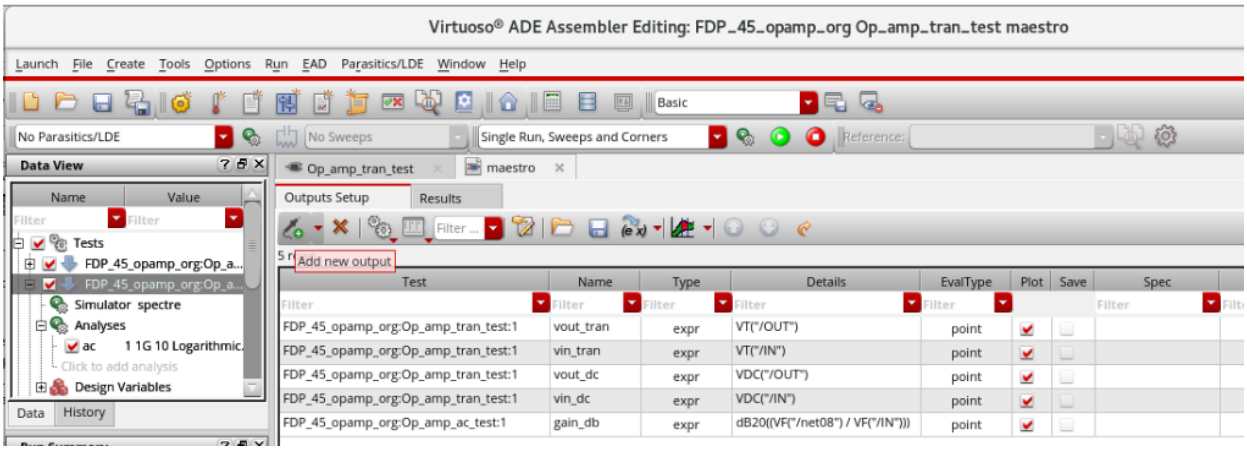



***Figure – 4.51: "Add new output" option***

Select "**FDP_45_opamp_org:Op_amp_tran_test → Expression**" as shown in Figure – 4.52.

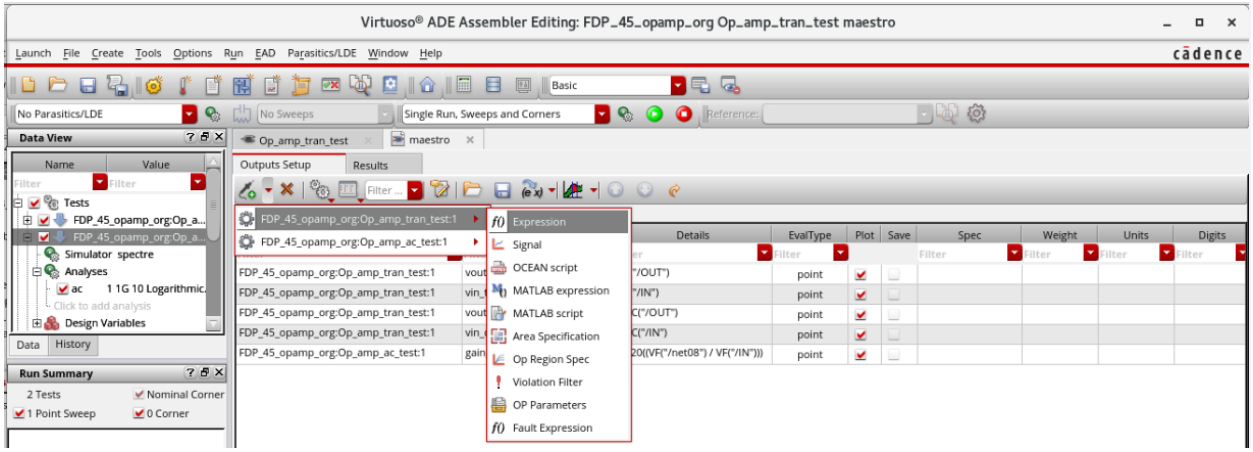



***Figure – 4.52: FDP_45_opamp_org:Op_amp_tran_test → Expression***

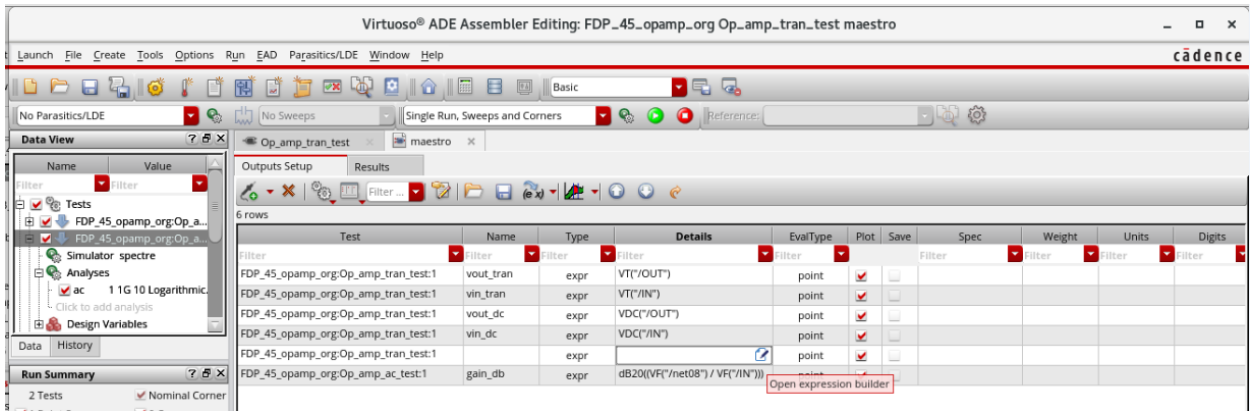



***Figure – 4.53: Open expression builder***

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.

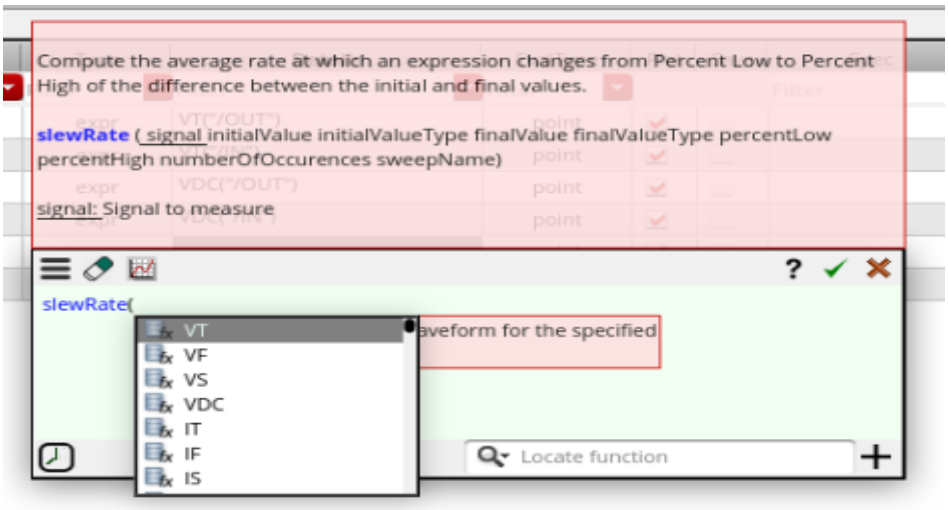



***Figure – 4.54: "slewRate" auto-completion***

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.

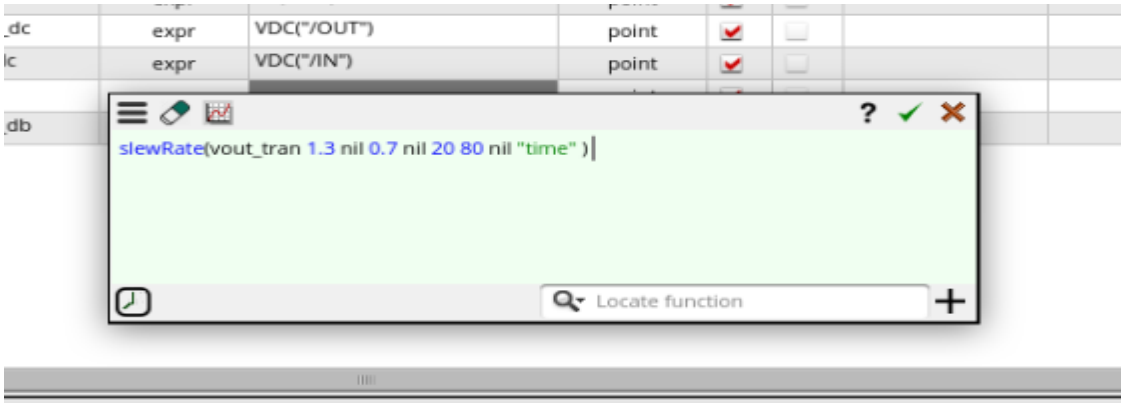



***Figure – 4.55: Completed expression***

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.

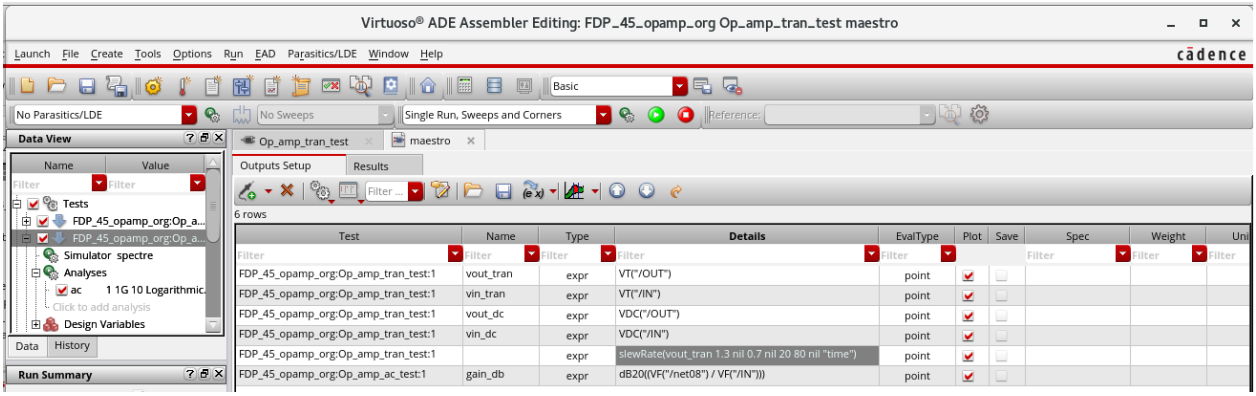



***Figure – 4.56: Expression updated in ADE Assembler***

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

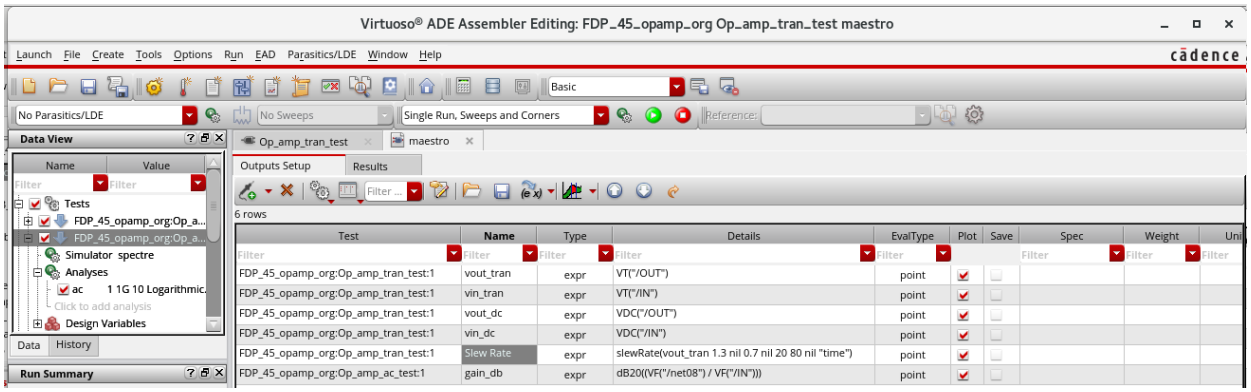



***Figure – 4.57: Naming the expression***

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"))) | 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**".


***Figure – 4.60: Dissipated Power***

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 and evaluate the data.

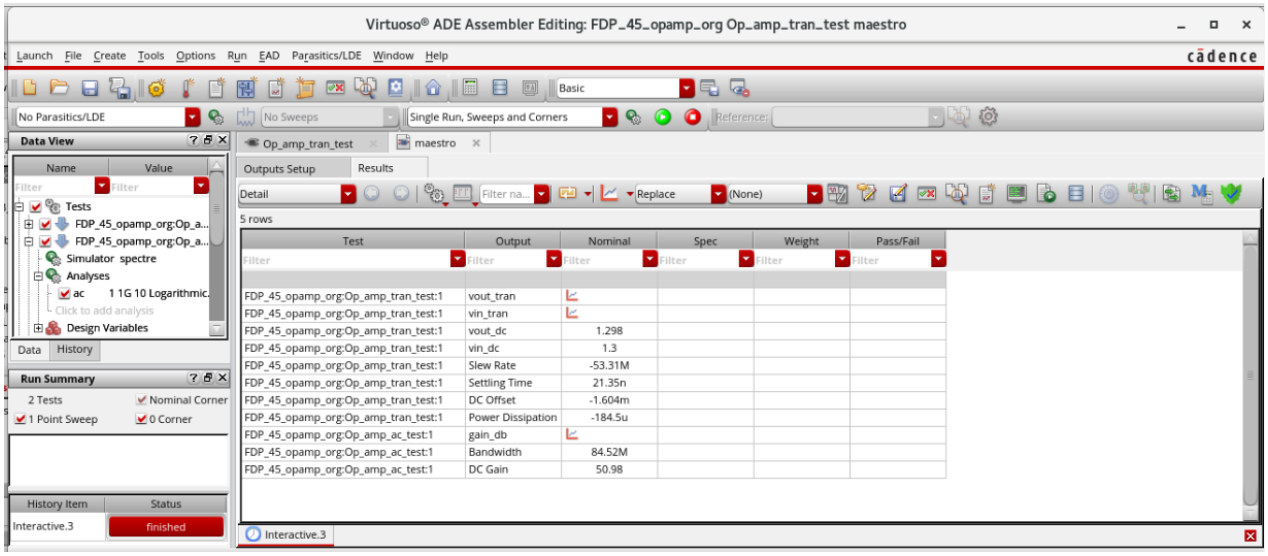



***Figure – 4.64: Evaluated Results after re-simulation***

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.

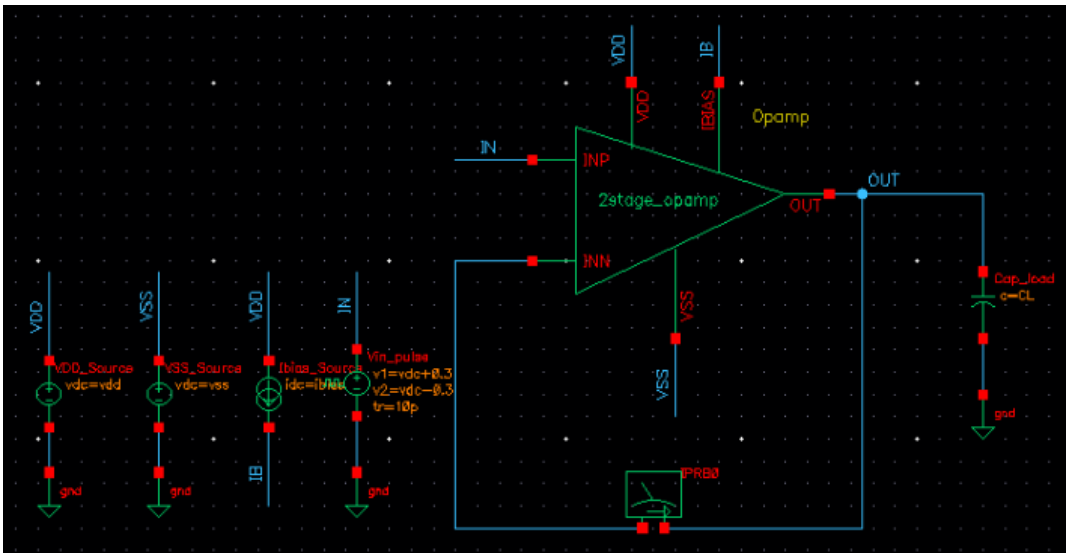



***Figure – 4.68: Schematic for Gain Margin and Phase Margin measurement***

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.

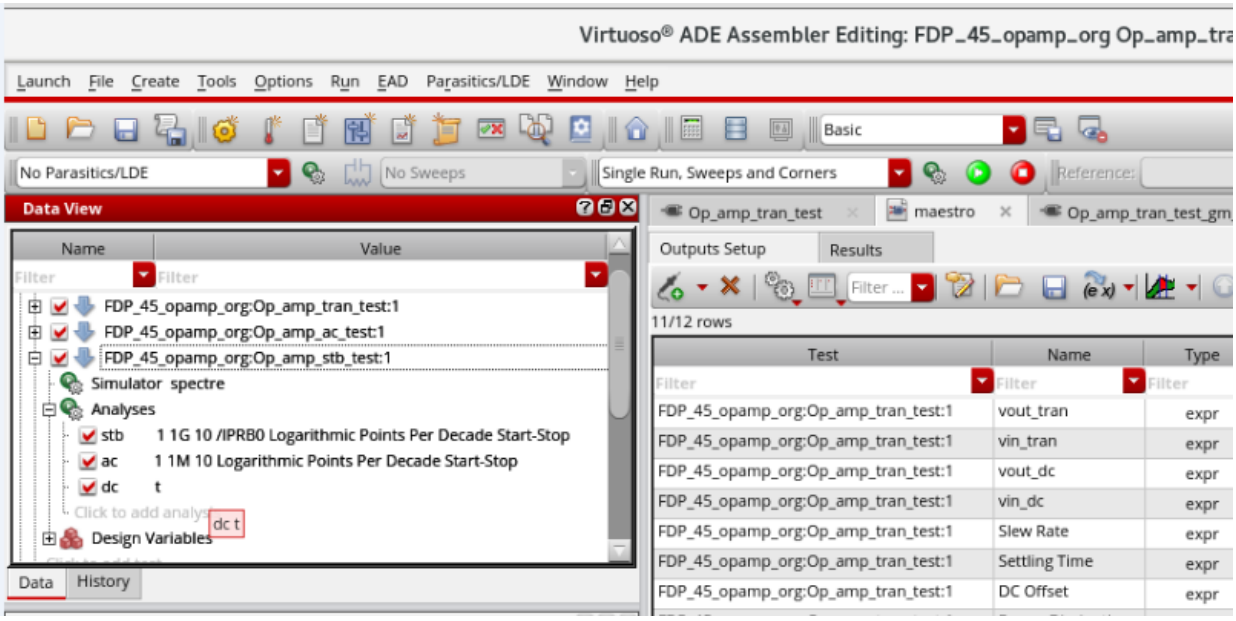



***Figure – 4.69: Test Creation for Stability Analysis***

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

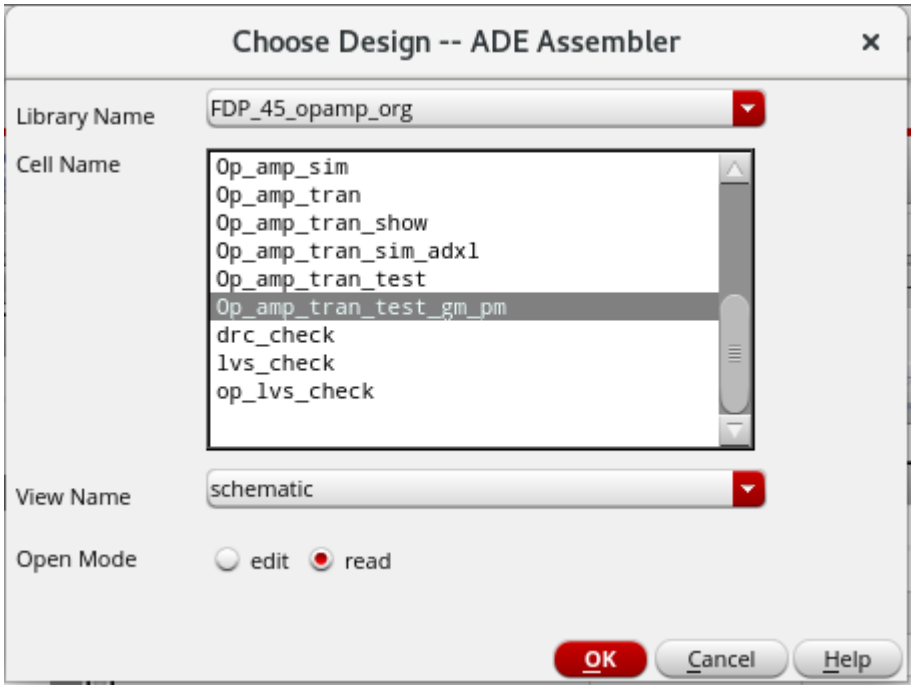



***Figure – 4.70: Design Selection***

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

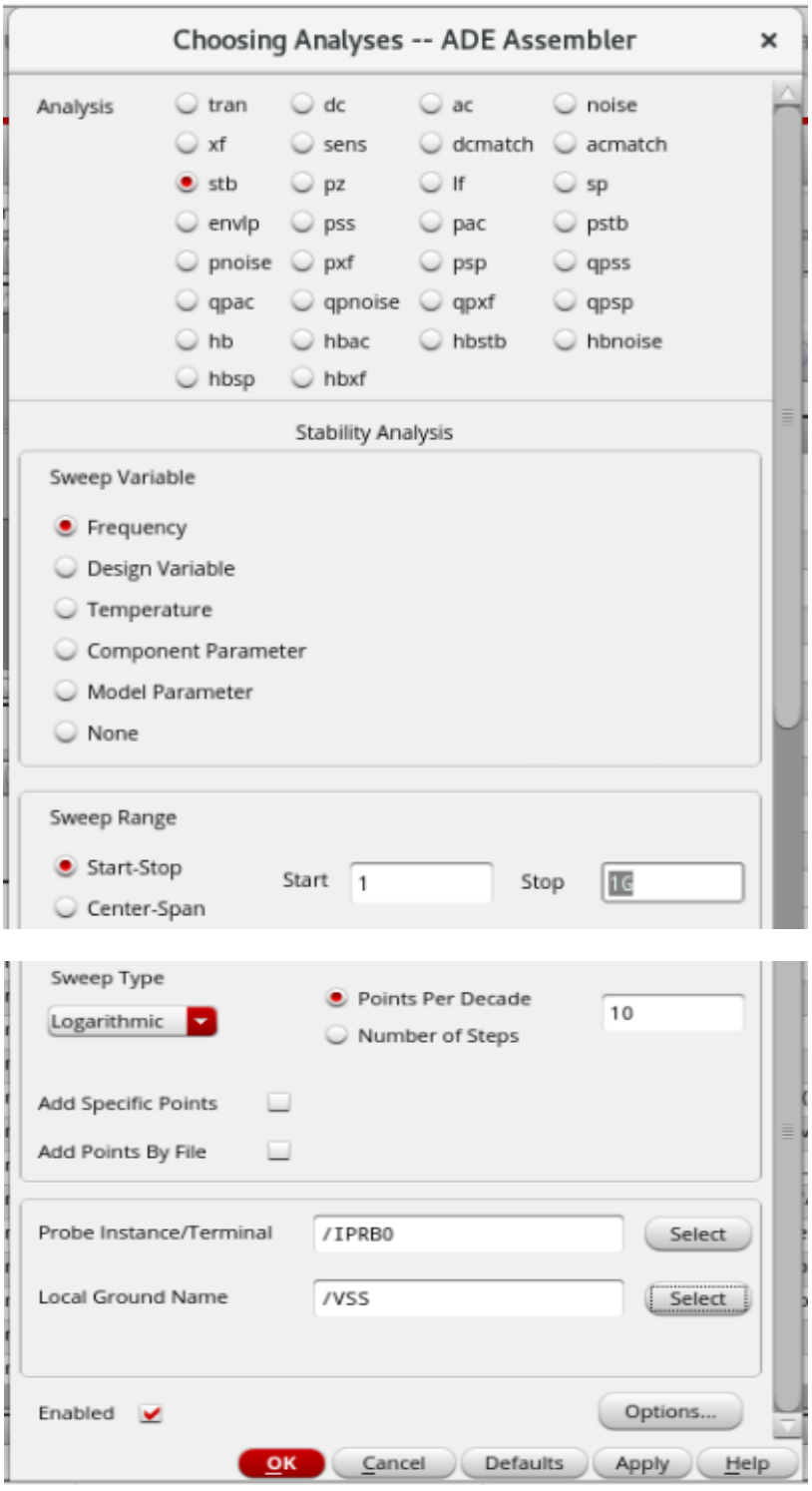



***Figure – 4.71: "stb" selection and its parameters***

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

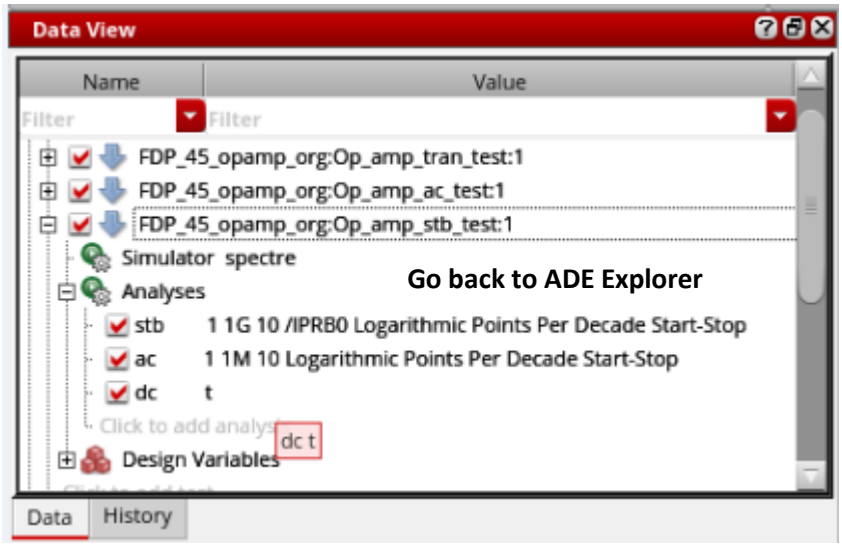



*Figure – 4.72: Click on downward arrow*

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

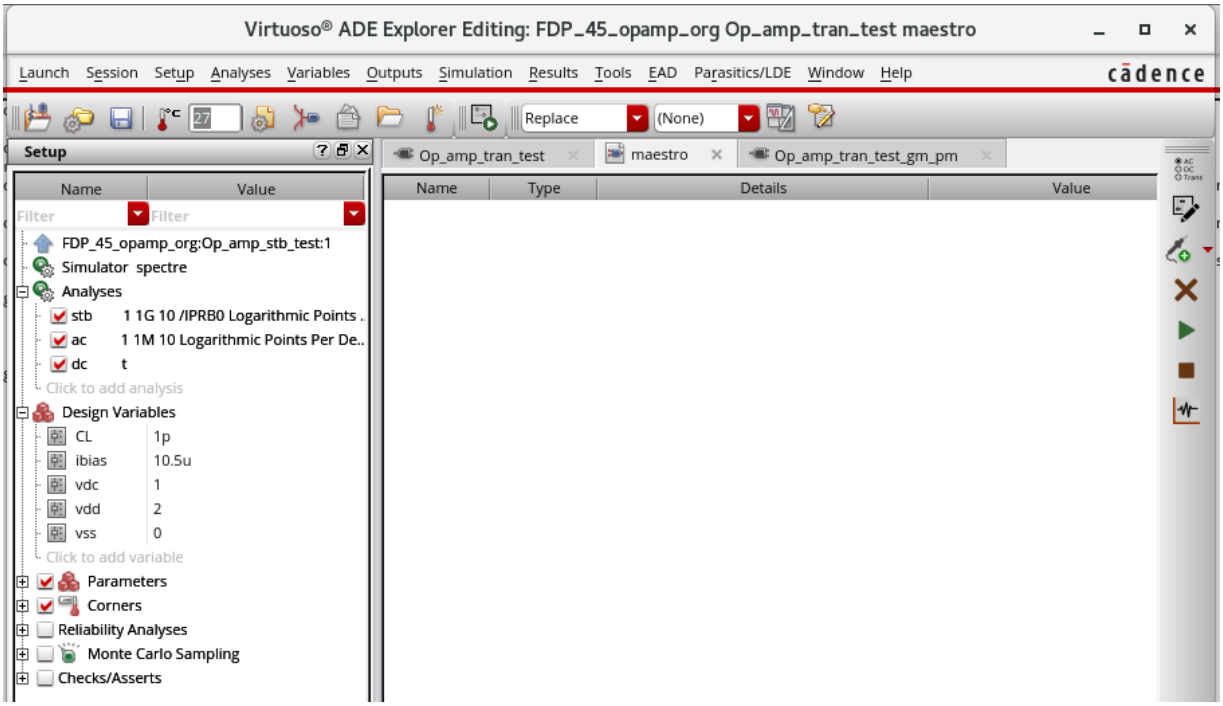



*Figure – 4.73: ADE Explorer*

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.

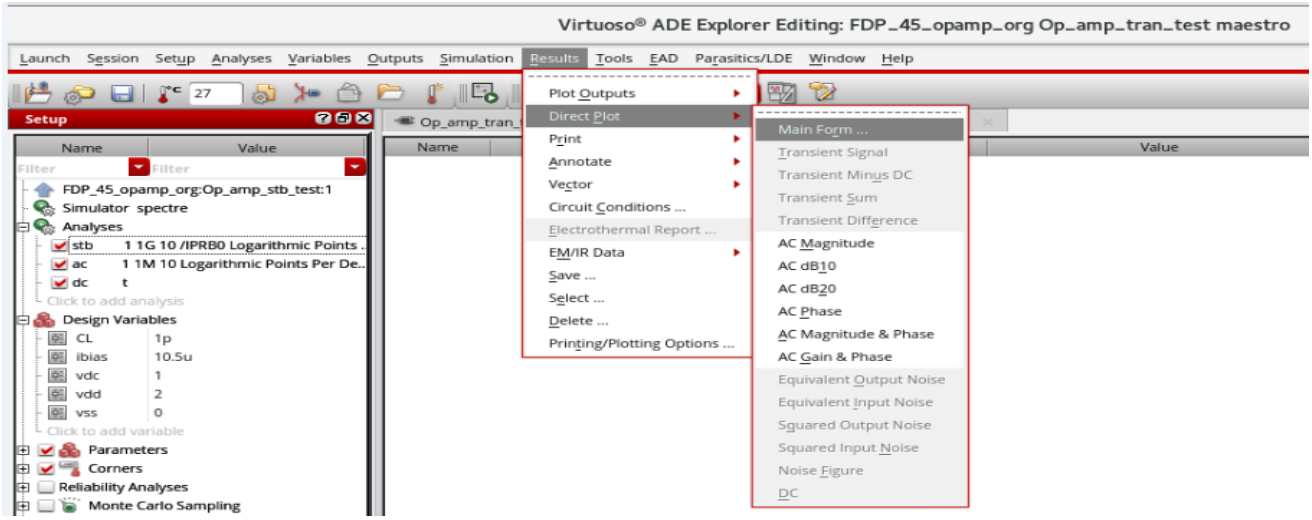



*Figure – 4.74: Results → Direct Plot → Main Form*

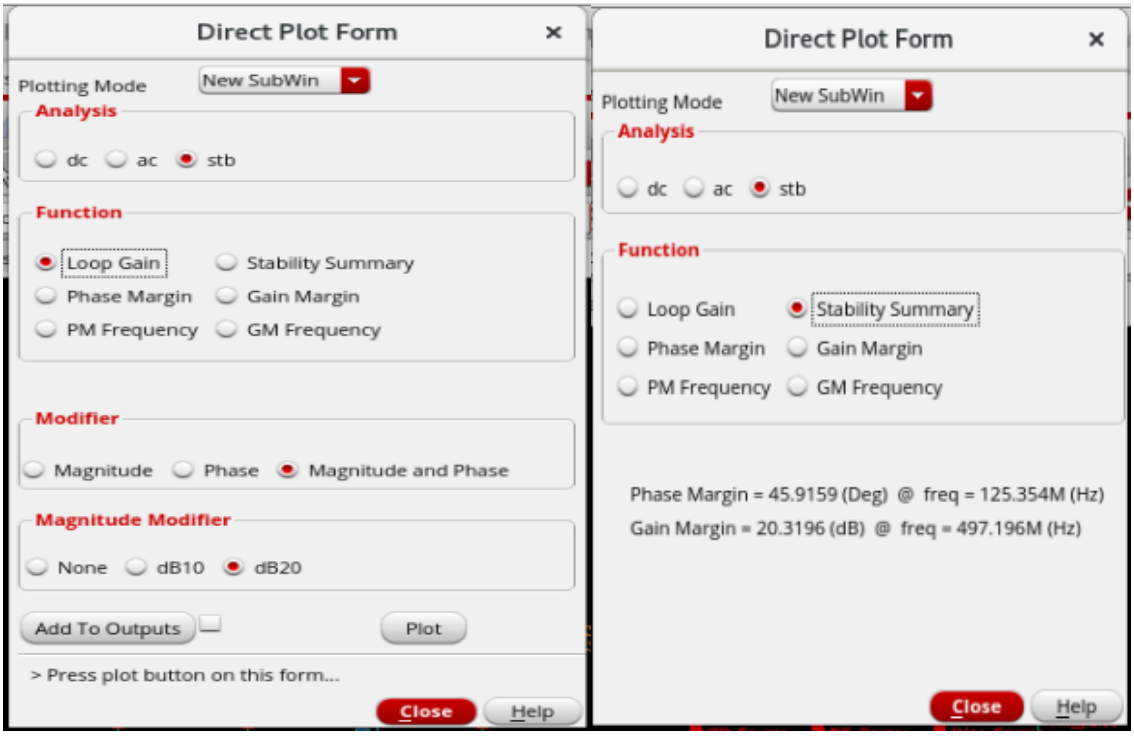



*Figure – 4.75: Direct Plot Form and Stability Summary with Gain Margin and Phase Margin*

**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.

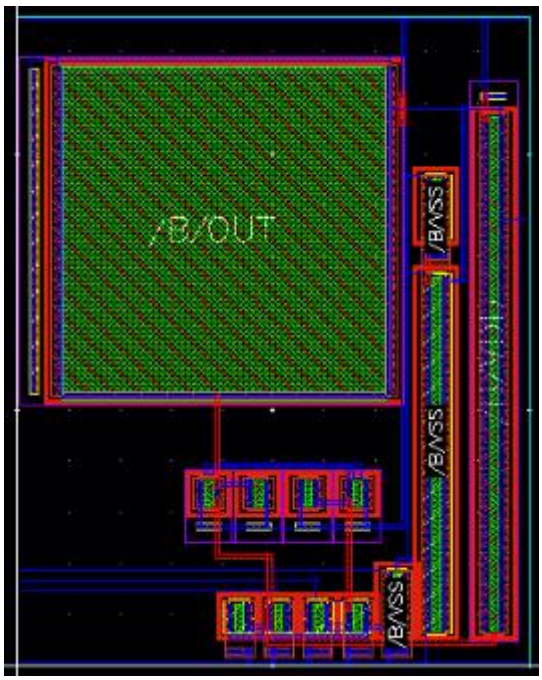



***Figure – 4.76: Layout for 2 – Stage Operational Amplifier***

**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 SCHEMATIC EDITOR**

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


***Figure – a: Options → User Preferences***

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.

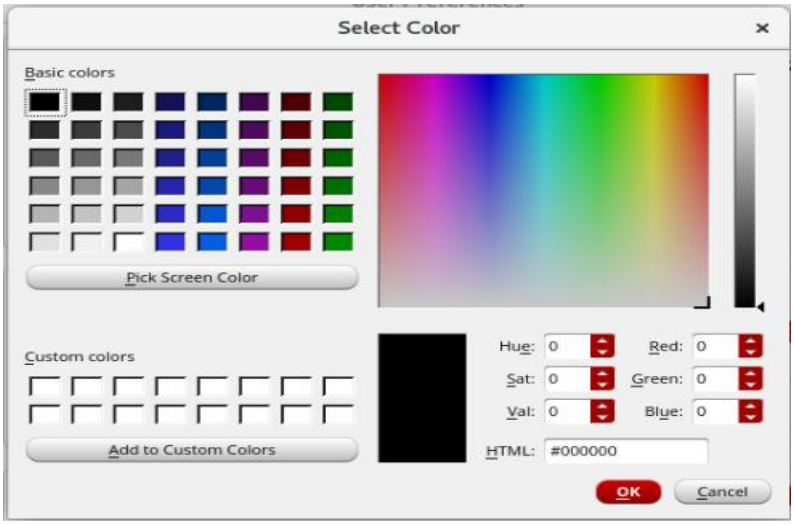



***Figure – c: Select Color window***

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.

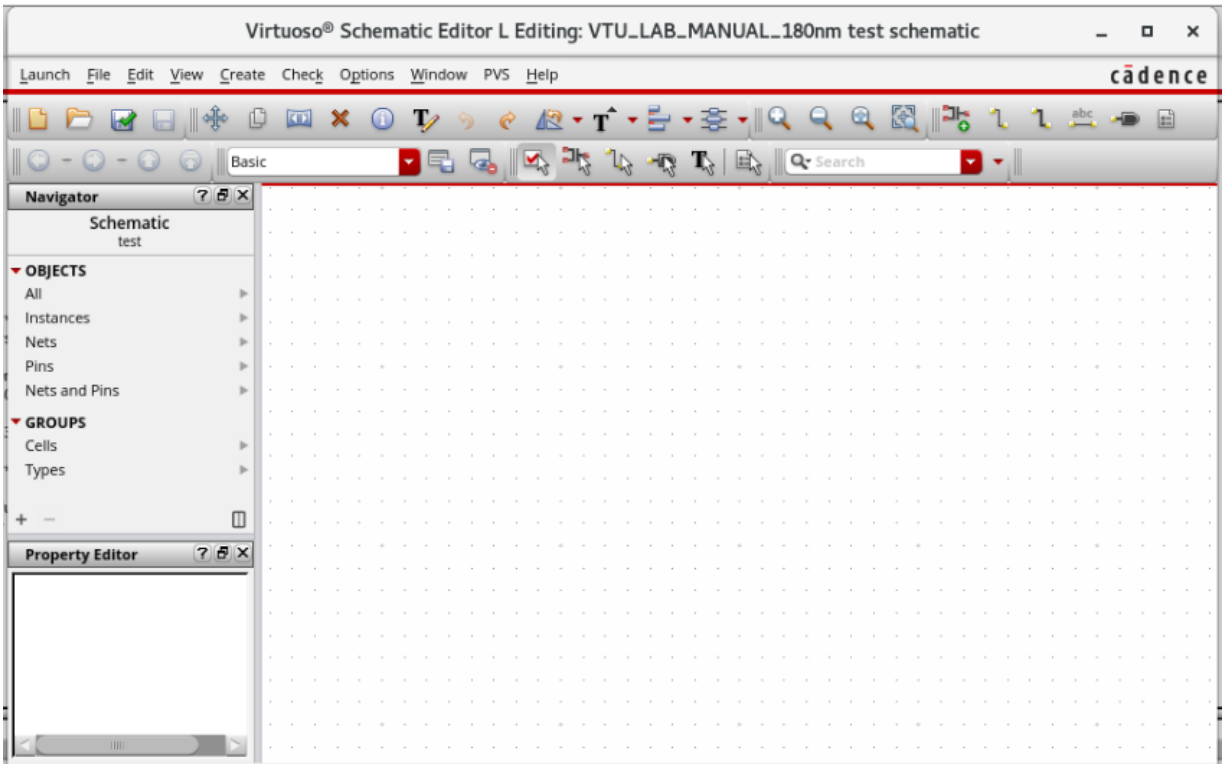



***Figure – e: Updated Virtuoso Schematic Editor L Editing window***

# PART-B
# DIGITAL DESIGN

## Lab 1 : 4-bit Up/Down Asynchronous Counter

**Objective :**

a) Verify the Functionality using Test bench
b) Synthesize the design with constraints and analyse reports, critical path.
c) Objectives a,b for 32-bit counter.

**Creating a Work space :**

 Create a Directory (New Folder) for yourself and name it your roll Number. Create a new sub Directory for the Design and open a terminal from the Sub-Directory.

### a) Verifying the Functionality using Test Bench

Step 1 : Creating Source Code

 In the Terminal, type gedit <filename>.v or <filename>.vhdl depending on the HDL Language you are to use.
 A Blank Document opens up into which the following source code can be typed down.

**4-Bit Up-Down Asynchronous Counter** :-

```
`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 reg [3:0]count;                //Defining 4-bit Output as Reg type
always@(posedge clk or negedge rst) //The Block is executed when
begin                                 //EITHER of positive edge of clock
if(!rst)                              //or Neg Edge of Rst arrives
count=0;                              // Both are independent events
if(m)                                 //i.e., Asynchronous
count=count+1;
else
count=count-1;
end
endmodule
```

 Use Save option or Ctrl+S to save the code and close the text file.

Step 2 : Creating Test bench

 Similarly, create your test bench using gedit <filename_tb>.v or <filename_tb>.vhdl to open a new blank document.

**Test Bench :**

```
`timescale 1ns/1ps                          //Creating Time Scale as in Source Code
module counter_test;                        //Defining Module Name without Port List
reg clk, rst,m;                             //Defining I/P as Registers [to Hold Values]
wire [3:0] count;                           //Defining O/P as Wires [To Probe Waveforms]
initial
begin
clk=0;                                      //Initializing Clock and Reset
rst=0;#25;                                  //All O/P is 4'b0000 from t=0 to t=25ns.
rst=1;                                      //Up-Down counting is allowed at posedge clk
end
initial
begin
m=1;                                        //Condition for Up-Count
#600 m=0;                                   //Condition for Down-Count
rst=0;#25;
rst=1;
#500 m=0;
End
counter counter1(clk,m,rst, count);         //Instantiation of Source Code
always #5 clk=~clk;                         //Inverting Clk every 5ns
initial
#1400 $finish;                              //Finishing Simulation at t=1400ns
Endmodule
```

 Save and Close the file

Step 3 : Compilation and Simulation

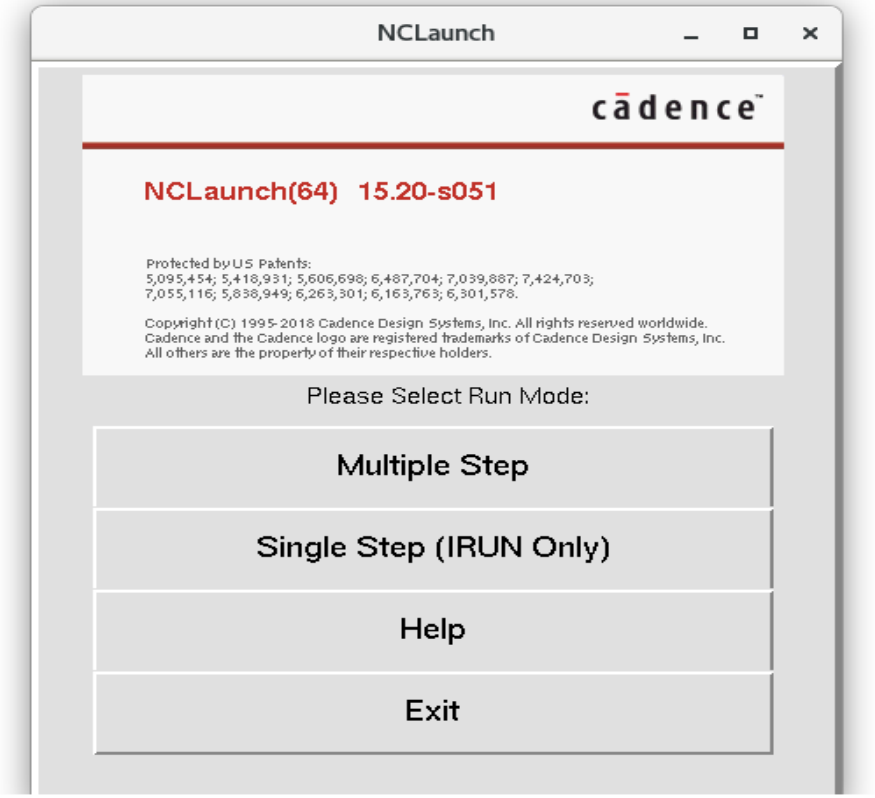
 The Tool used for Simulation is Incisive Enterprise and the command used to invoke the tool is "nclaunch -new"
 The -new switch is to be used only for the first time for a particular Design workspace.
 To Invoke the tool, type the following Three commands in your terminal.

```
1) csh                                      //Invokes C-Shell
2) source /home/install/cshrc               //Tool Resources are loaded
```

(The path of cshrc could vary depending on the installation destination as /home/cad/ or /cad etc.)
- A New Page on the terminal appears with the following line,
"Welcome to Cadence Tools Suite"

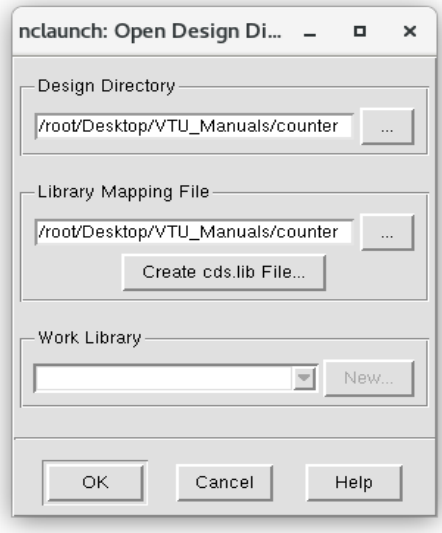
```
3) nclaunch -new                            //Invokes Incisive Tool
```

```
Welcome to Cadence Tools Suite

[root@krishna counter]# nclaunch -new
nclaunch(64): 15.20-s051: (c) Copyright 1995-2018 Cadence Design Systems, Inc.
```

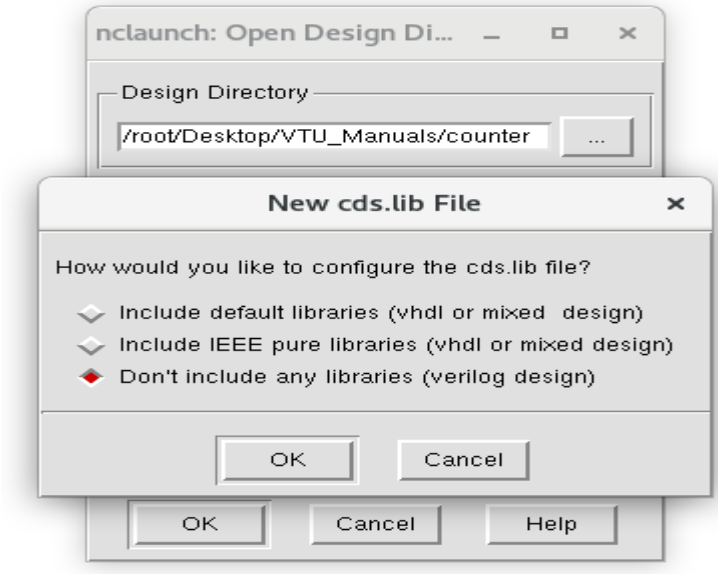



 Select Multiple Step and then select “Create cds.lib File”

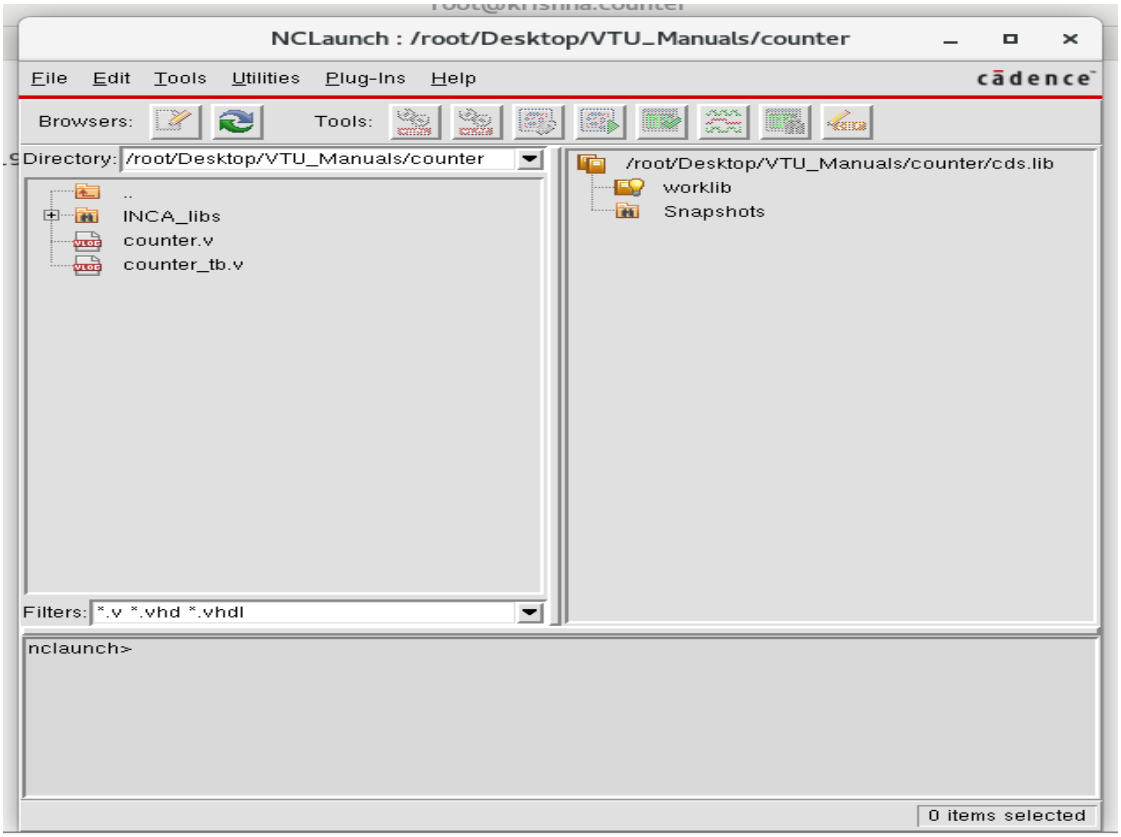
```
Welcome to Cadence Tools Suite

[root@krishna counter]# nclaunch -new
nclaunch(64): 15.20-s051: (c) Copyright 1995-2018 Cadence Design Systems, Inc.
```



 Save cds.lib file and select the correct option for cds.lib file format based on the HDL Language and Libraries used.



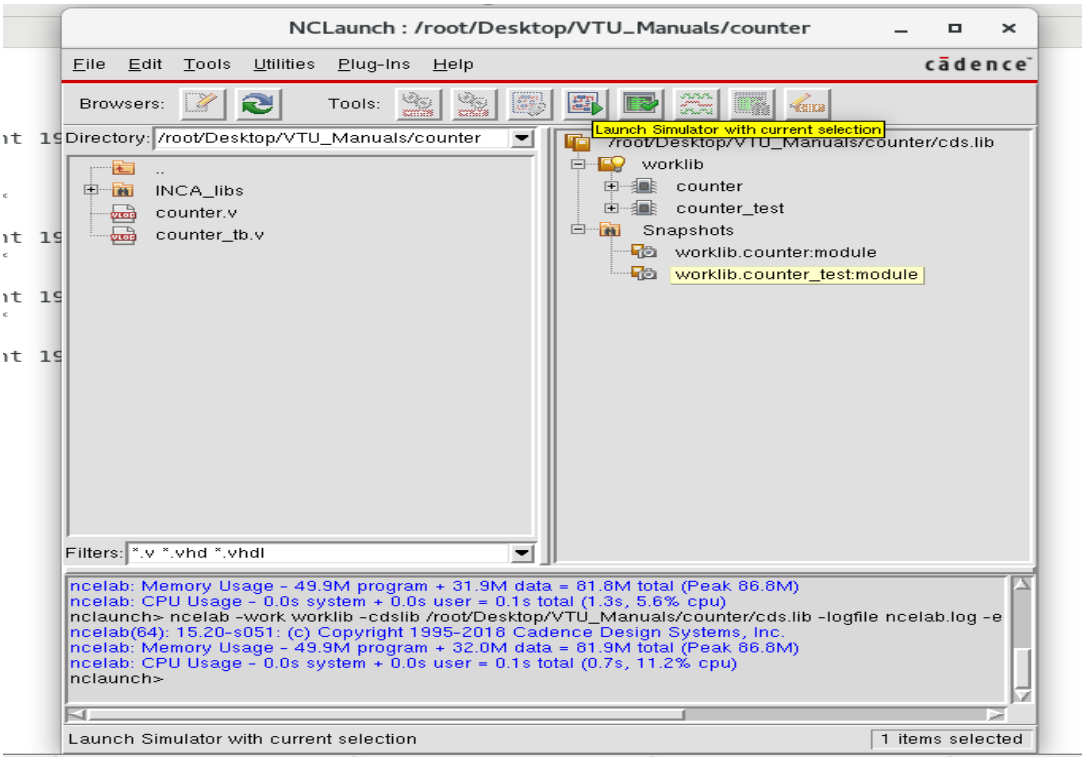

- By Default, Verilog Designs uses Third Option of "Don't include any libraries".
- Nclaunch window pops up where the Compilation, Elaboration and Simulation would be done, with list of .v or .vhdl files on the left.



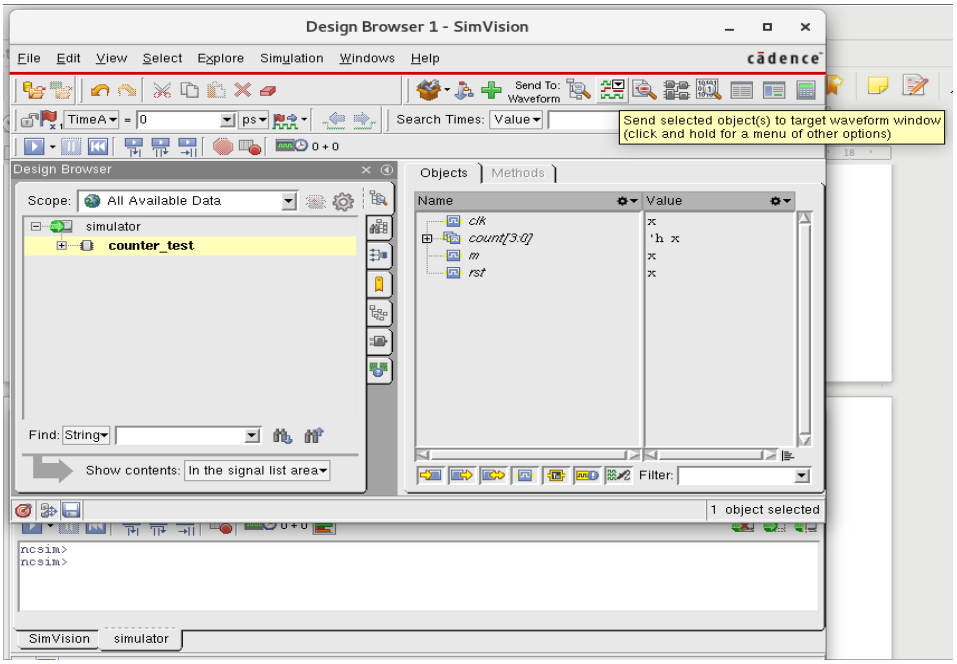
The horizontal list of icons at the top under “Tools” are as follows,

1. VHDL Compiler - Compiles VHDL Codes
2. Verilog Compiler - Compiles Verilog Codes
3. Elaborator - Elaborates/Details the Compiled Codes
4. Simulator - Simulates the Test Benches

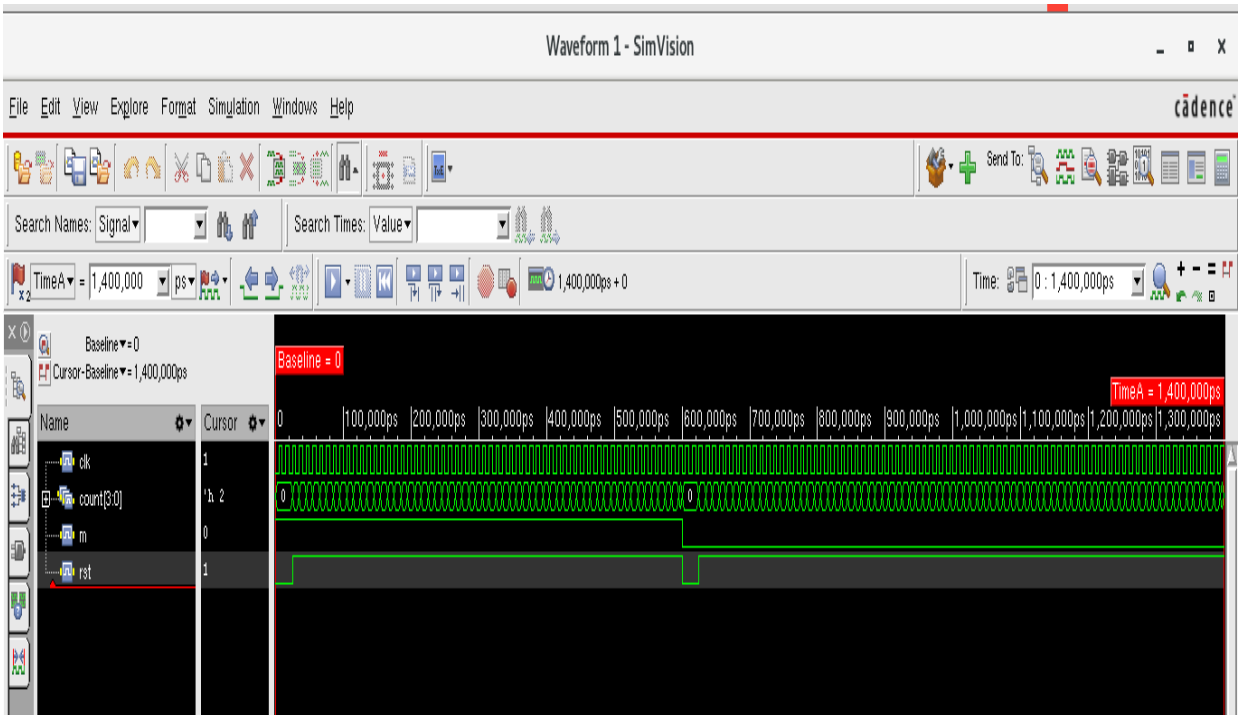
- Compiling the Codes :
  - Select the counter.v file on left and Verilog Compiler under Tools.
  - Similarly, select counter_test.v on left and VHDL Compiler under Tools.
- Elaborating the Design :
  - Expand the Worklib to find the compiled designs with module names.
  - Select each and click on “Launch Elaborator”.
- Simulating the Design :
  - Expand Snapshots and select the “worklib:counter_test:module” and click on “Launch Simulator” to verify Functionality of design




- A Design Browser window opens with the test bench module name on the left. Select it to see the I/O pins on the right.




- Make a right click on counter_test → Send to waveform Window.
- In the Waveform Window, select Simulation (On Top) → Run (or F2) to see the simulation waveform. Click on the "=" on the top right to fit the waveform to screen.
- You can click anywhere on the waveform and use +/- to Zoom In/Out



**b) Synthesize the design using Constraints and analyse 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/
- The Available technology nodes are 180nm ,90nm and 45nm.
- In the terminal, initialize 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 counter.v

3. elaborate
4. read_sdc constraints_top.sdc //Reading Top Level SDC
5. synthesize -to_mapped -effort medium
   //Performing Synthesis Mapping and Optimisation
6. report_timing > counter_timing.rep
   //Generates Timing report for worst datapath and dumps into file
7. report_area > counter_area.rep
   //Generates Synthesis Area report and dumps into a file
8. report_power > counter_power.rep
   //Generates Power Report [Pre-Layout]
9. write_hdl > counter_netlist.v //Creates readable Netlist File
10. 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) report_timing gives you the path with highest failing slack where Setup Slack = Required Time – Arrival Time.

2) Worst Setup Slack ==> Highest Arrival time ==> Highest Propagation Delay.

3) Maximum Clock Frequency = 1/ (Max Data Path Delay – Min Clock Path Delay + Tsetup)

All the Information can be gathered from report_timing.

4) The Cells given in the netlist can be checked in the .lib files for their properties.

**b)** Compilation, Simulation and Synthesis of 32-bit Up/Down Counter.

Source Code :

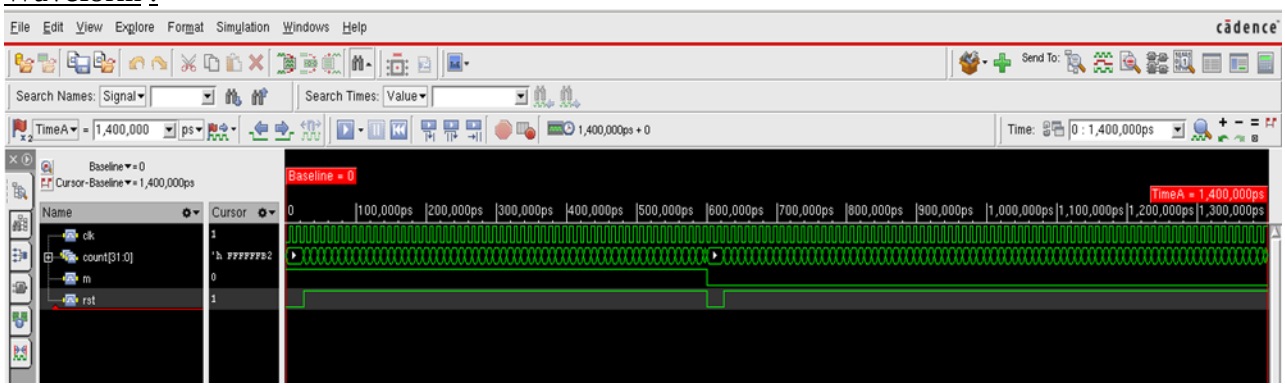
```
`timescale 1ns/1ps                          //Defining a Timescale for
Precision module counter(clk,rst,m,count);  //Defining Module and Port List
input clk,rst,m;                            //Defining Inputs
output reg [31:0]count;//Defining 4-bit Output as Reg type
 always@(posedge clk or negedge rst)  //The Block is executed when
 begin                                      //EITHER of positive edge of
clock
if(!rst)                                    //or Neg Edge of Rst arrives
count=0;                                    // Both are independent
events if(m)
count=count+1;
else
count=count-1;
```

```
end
endmdule
```

Test Bench :

```
`timescale 1ns/1ps                  //Creating Time Scale as in Source Code
module counter_test;                //Defining Module Name without Port List
reg clk, rst,m;                     //Defining I/P as Registers [to Hold Values]
wire [31:0] count;                  //Defining O/P as Wires [To Probe Waveforms]
initial
begin

        clk=0;                                  //Initializing Clock and Reset


 rst=0;#25;                         //All O/P is 4'b0000 from t=0 to t=25ns.


 rst=1;                                         //Up-Down counting is allowed at posedge clk
 end


initial
begin                               //Condition for Up-Count
m=1;

 #600 m=0;                          //Condition for Down-Count
 rst=0;#25;
 rst=1; #500
 m=0;
 end


counter counter1(clk,m,rst, count);             //Instantiation of Source
Code always #5 clk=~clk;                        //Inverting Clk every 5ns
initial
#1400 $finish                       //Finishing Simulation at t=1400ns endmodule
```

Waveform :



# Lab 2 : 4-Bit Adder

**Objectives :**

a) Verify the functionality using Test bench
b) Synthesize, Analyse Reports and Netlist, Critical Path and Max Operating Frequency.

**Creating a Work space :**

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

## a) Verify the Functionality

- Three Codes shall be written for implementation of 4-bit Adder as follows,
  - 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

Source Code – fa.v :-

```
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
```

Source Code – fa_4bit.v :-

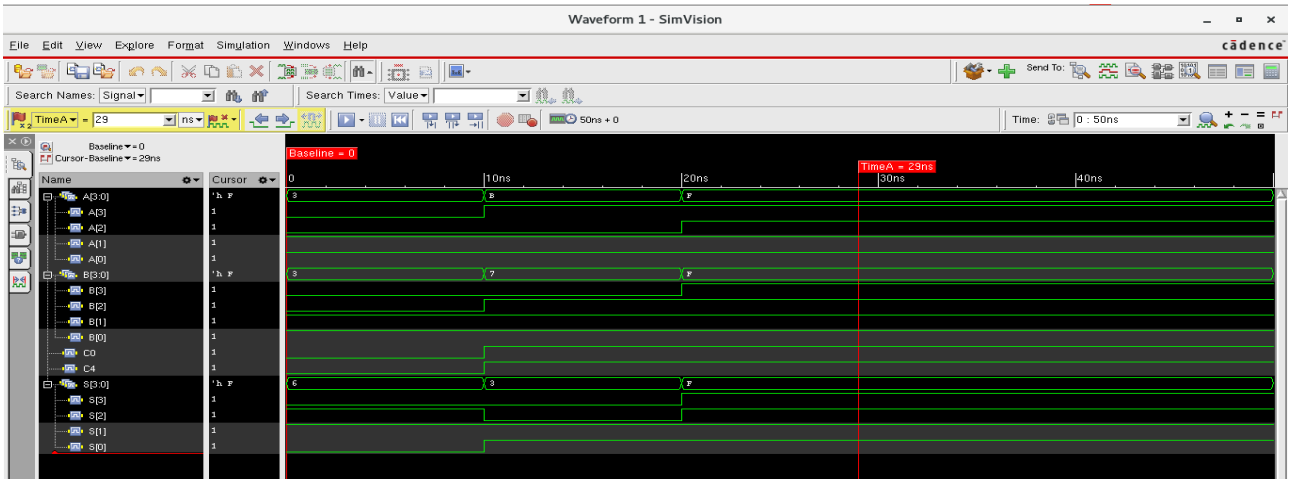
```
module four_bit_adder(A,B,C0,S,C4);
 input [3:0] A,[3:0] B,C0;
output [3:0] S,C4; wire
C1,C2,C3;

                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,

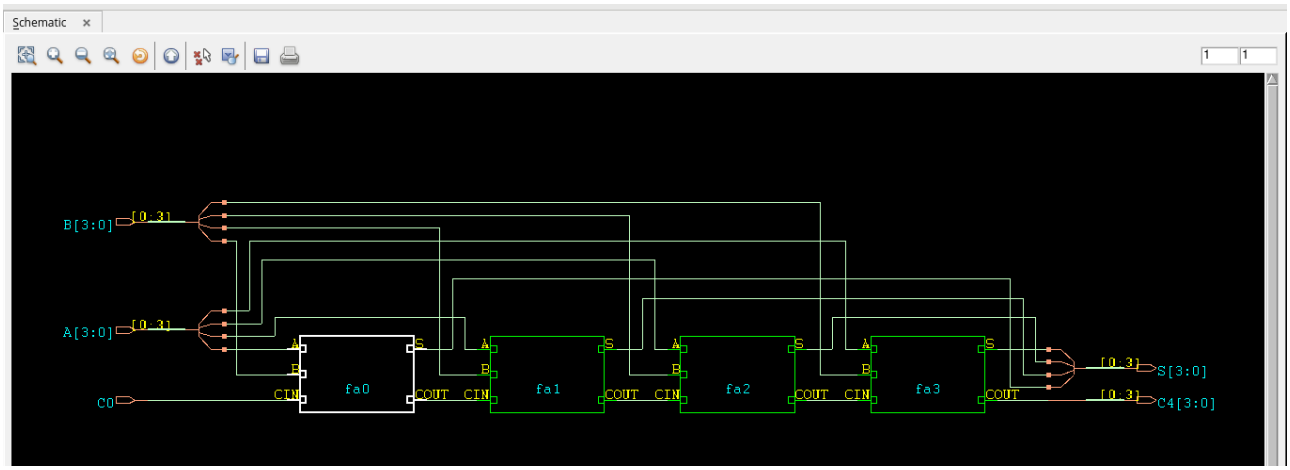
1. read_libs /home/install/FOUNDRY/digital/90nm/dig/lib/slow.lib
2. read_hdl {fa.v fa_4bit.v} //Reading multiple Verilog Files
3. elaborate
4. set_top_module four_bit_adder //Differentiating Top & Sub Module
5. set_dont_use *XL //Dont Use Cells with High Driving Strength
6. synthesize -to_mapped -effort medium
   //Performing Synthesis Mapping and Optimisation
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 reference

### 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. You can 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.

# Lab 3 : UART

**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

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
 #(parameterCLKS_PER_BIT = 217)
  (
  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 IDLE         = 3'b000;
parameter TX_START_BIT = 3'b001;
parameter TX_DATA_BITS = 3'b010;
parameter TX_STOP_BIT = 3'b011;
parameter 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)
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 <= TX_START_BIT;
             end
          else
       r_SM_Main <= IDLE;
       end // case: IDLE
// Send out Start Bit. Start bit = 0
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  <= TX_START_BIT;
            end
            else
            begin
            r_Clock_Count <= 0;
            r_SM_Main     <= TX_DATA_BITS;
             end
        end // case: TX_START_BIT
        // Wait CLKS_PER_BIT-1 clock cycles for data bits to finish
       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          <= 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 <= TX_DATA_BITS;
 end
else
begin
r_Bit_Index <= 0;
r_SM_Main <= TX_STOP_BIT;
 end
end
end // case: TX_DATA_BITS
// Send out Stop bit. Stop bit = 1
 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    <= TX_STOP_BIT;
end
else
begin
 r_TX_Done    <= 1'b1;
r_Clock_Count <= 0;
r_SM_Main     <= CLEANUP;
 r_TX_Active <= 1'b0;
end
end // case: TX_STOP_BIT
// Stay here 1 clock
                                    CLEANUP :
begin
r_TX_Done <= 1'b1;
r_SM_Main <= IDLE;
end
default :
r_SM_Main <= 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 = 217)
 (
  input       i_Clock,
  input       i_RX_Serial,
  output     o_RX_DV,
  output [7:0] o_RX_Byte
  );
  parameter IDLE = 3'b000;
   parameter RX_START_BIT = 3'b001;
   parameter RX_DATA_BITS = 3'b010;
  parameter RX_STOP_BIT = 3'b011;
  parameter CLEANUP    = 3'b100;
 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: Control RX state machine
always @(posedge i_Clock)
begin
case (r_SM_Main)
IDLE :
     begin
      r_RX_DV       <= 1'b0;
      r_Clock_Count <= 0;
      r_Bit_Index <= 0;

      if (i_RX_Serial == 1'b0)          // Start bit detected
       r_SM_Main <= RX_START_BIT;
      else
      r_SM_Main <= IDLE;
       end
      // Check middle of start bit to make sure it's still low
      RX_START_BIT :
Begin
          if (r_Clock_Count == (CLKS_PER_BIT-1)/2)
                                                  begin
                                        if (i_RX_Serial == 1'b0)
                                                        begin
          r_Clock_Count <= 0; // reset counter,
          found the middle
```

```
                r_SM_Main     <= RX_DATA_BITS;
end
else
r_SM_Main <= IDLE;
end
else
 begin
r_Clock_Count <= r_Clock_Count + 1;
r_SM_Main<= RX_START_BIT;
end
end // case: RX_START_BIT
// Wait CLKS_PER_BIT-1 clock cycles to sample serial data
RX_DATA_BITS :
     begin
if (r_Clock_Count < CLKS_PER_BIT-1)
 begin
r_Clock_Count <= r_Clock_Count + 1;
r_SM_Main<= RX_DATA_BITS;
end
else
begin
r_Clock_Count<= 0;
 r_RX_Byte[r_Bit_Index] <= i_RX_Serial;
if (r_Bit_Index < 7)
 begin                  // Check if we have received all bits
```

```
r_Bit_Index <= r_Bit_Index + 1;
 r_SM_Main <= RX_DATA_BITS;
end
else
begin
r_Bit_Index <= 0;
r_SM_Main <= RX_STOP_BIT;
end
end
end // case: RX_DATA_BITS
// Receive Stop bit. Stop bit = 1
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 <= RX_STOP_BIT;
        end
        else
        begin
        r_RX_DV    <= 1'b1;
```

```
r_Clock_Count <= 0;
r_SM_Main<= CLEANUP;
end

end // case: RX_STOP_BIT

// Stay here 1 clock

CLEANUP :

Begin

r_SM_Main <= IDLE;
r_RX_DV <= 1'b0;

end

default :

r_SM_Main <= IDLE;

endcase

 end

assign o_RX_DV = r_RX_DV;

assign o_RX_Byte = r_RX_Byte;

endmodule // UART_RX
```

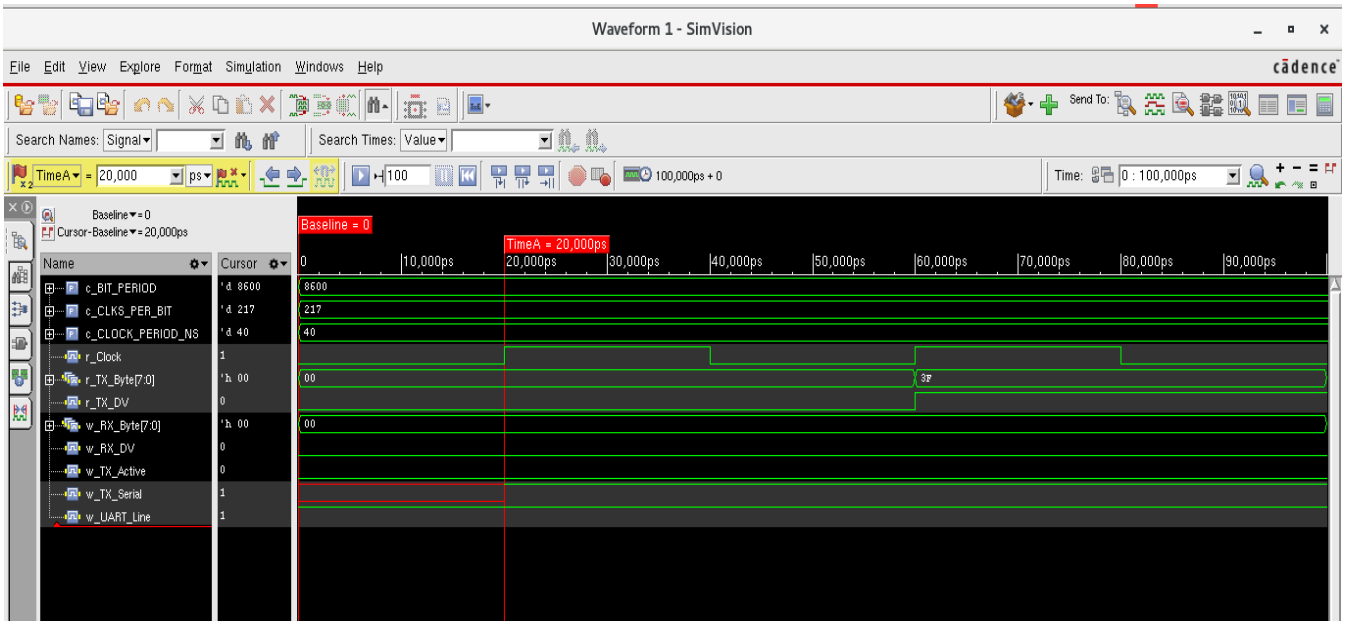
Test bench :

```
// 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
```

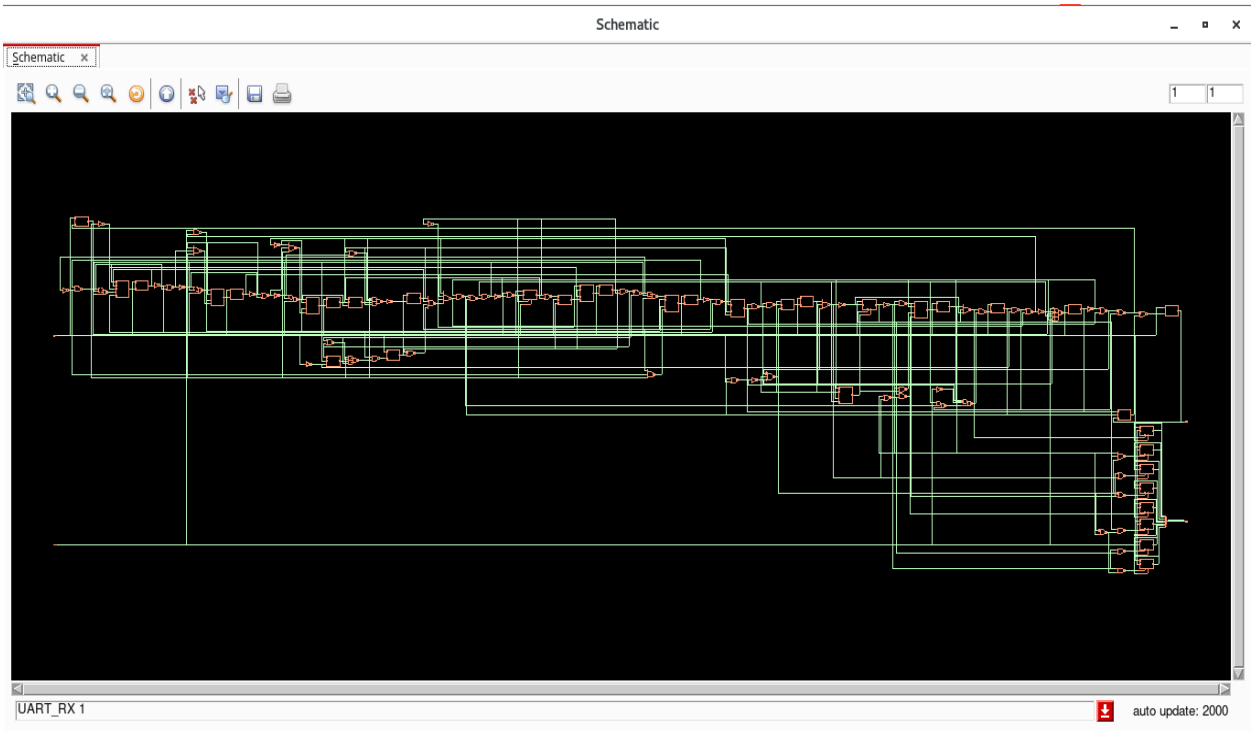
**Waveform**



### a) 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.sd //Reading Top Level SDC
5. synthesize -to_mapped -effort medium //Performing Synthesis Mapping and Optimisation
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 :




Note :-

1. You can 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.

# Lab 4 : 32-bit ALU

## Objective :-

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.

**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;
```

```
End
case 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
```

**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;
```

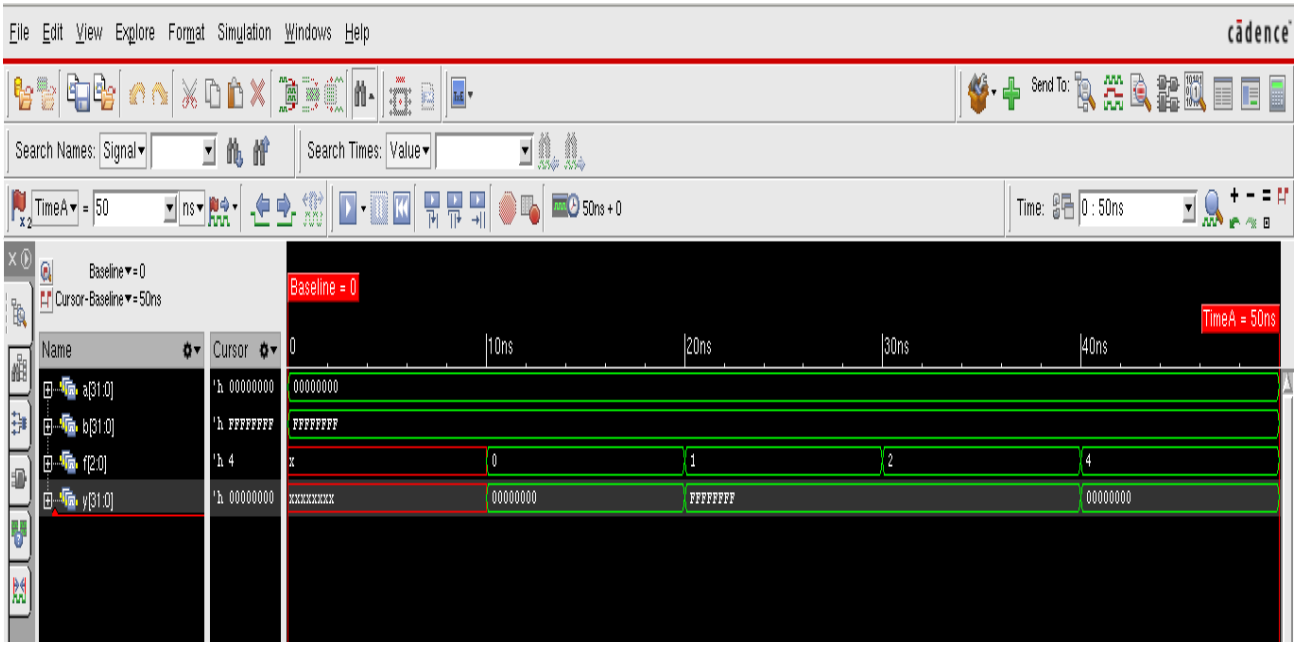
```
always@(*)
begin
if(f==3'b000)
  y=a&b;//AND Operation
  else if (f==3'b001)
  y=a|b;//OR Operation
  else if (f==3'b010)
  y=a+b;              //Addition else if (f==3'b011)
  y=a-b;              //Subtraction else if (f==3'b100)
  y=a*b;              //Multiply else
  y=32'bx;
   end
endmodule
```

**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
```

**Wave Forms :**




## b) Synthesize Design

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

**Synthesis Process :**

a. read_libs /home/install/FOUNDRY/digital/90nm/dig/lib/slow.lib
b. read_hdl {alu_32bit_if.v (OR) alu_32bit_case.v}
   //Choose any one
c. elaborate
d. read_sdc constraints_top.sdc //Optional-Reading Top Level SDC

e. synthesize -to_mapped -effort medium
   //Performing Synthesis Mapping and Optimisation
f. report_timing > alu_timing.rep

   //Generates Timing report for worst datapath and dumps into file
g. report_area > alu_area.rep
   //Generates Synthesis Area report and dumps into a file
h. report_power > uart_power.rep
   //Generates Power Report [Pre-Layout]
i. write_hdl > uart_netlist.v //Creates readable Netlist File
j. write_sdc > uart_sdc.sdc
//Creates Block Level SDC

## Note :-

1. You can 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.

## Lab 5 : Latches and Flip Flops

Objective :

a) Write verilog codes for Flip flops D,JK, SR.
b) Write verilog codes for Latches D,JK, SR.
c) Synthesize the designs

## Creating a Work space :

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

**a) Verilog Codes for D-Flip Flop, JK-Flip Flop and SR-Flip Flop.**

## Source code for D-Flip Flop :

```
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
```

**Source code for D-Latch :**

```
module DFF( Q,Qbar,D,en,Reset);
output reg Q;
output    Qbar;
input     D,en,Reset;
assign Reset = ~D;
always @(en)
begin
if (Reset == 1'b1) //If at reset
Q <= 1'b0;
 else
  Q <= D;
End
assign Qbar = ~Q;
endmodule
```

**Source code for SR Flip Flop :**

```
ModuleMain(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
```

Source Code for SR Latch :

```
module Main(S,R,en,Q,Qbar);
input S,R,en;
output Q,Qbar;
reg M,N;
always @(en)
begin
M <= !(S & clk);
N <= !(R & clk);
End
assign Q <= !(M & Qbar);
 assign Qbar <= !(N & Q);
endmodule
```

Source Code for JK Flip Flop :

```
Module jkff(J, K, clk, Q);
input J, K, clk;
output reg Q,Qm;
always @(posedge clk
begin
if(J == && K == 0)
Qm = 1;
else if(J == 0 && K == 1)
 Qm = 0;
else if(J == 1 && K == 1)
Qm = ~Qm;
end
endmodule
```

Source of JK Latch :

```
module jkff(J, K, en, Q);
input J, K, en;
 output reg Q,Qm;
always @(en)
 begin
if(J == 1 && K == 0) Qm <= 1;
else if(J == 0 && K == 1) Qm <= 0;
else if(J == 1 && K == 1) Qm <= ~Qm;
end
endmodule
```

## b) Synthesize the Design & Comparing Reports

Synthesis Commands :

1. read_libs /home/install_run/FOUNDRY/digital/90nm/dig/lib/slow.lib
2. read_hdl dff.v
3. elaborate
4. read_sdc constraints_top.sdc
5. synthesize -to_mapped -effort medium
6. report_timing > dff_timing.rep
7. report_area > dff_area.rep
8. report_power > dff_power.rep
9. write_hdl > dff_netlist.v
10. write_sdc > dff_sdc.sdc

# Example for SDC:

Open

constraints_top_ff.sdc
~/Desktop/VTU_Manuals/ff_latch

```
create_clock -name Clk -period 2 -waveform {0 1} [get_ports "Clk"]
set_clock_transition -rise 0.1 [get_clocks "Clk"]
set_clock_transition -fall 0.1 [get_clocks "Clk"]
set_clock_uncertainty 0.01 [get_ports "Clk"]
set_input_delay -max 1.0 [get_ports "Reset"] -clock [get_clocks "Clk"]
set_input_delay -max 1.0 [get_ports "D"] -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 "Qbar"] -clock [get_clocks "Clk"]
```

The Above is the example SDC for a D Flip Flop

Open

constraints_top_latch.sdc
~/Desktop/VTU_Manuals/ff_latch

```
set_input_delay -max 1.0 [get_ports "Reset"]
set_input_delay -max 1.0 [get_ports "en"]
set_input_delay -max 1.0 [get_ports "D"]
set_output_delay -max 1.0 [get_ports "Q"]
set_output_delay -max 1.0 [get_ports "Qbar"]
```

The above is an Example for SDC File for **D Latch.**

## Note :-

1. You can 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.**

# Lab 6 : Physical Design

Mandatory Inputs for PD:

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

Expected Outputs from PD:

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

- Make sure the Synthesis for the target design is done and open a terminal from the corresponding workspace.
- Initiate the Cadence tools and **cmd :**innovus (Press Enter)
- For Innovus tool, a GUI opens and also the terminal enters into innovus command prompt where in the tool commands can be entered.

Physical Design involves 5 stages as following : After Importing Design,
→ Floor Planning
→ Power Planning
→ Placement
→ CTS (Clock Tree Synthesis)
→ Routing

## Importing Design

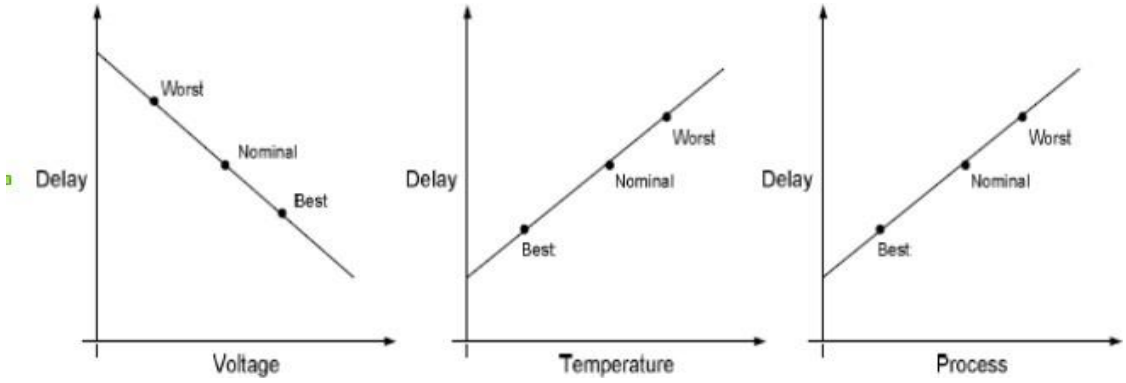
To Import Design, all the Mandatory Inputs are to be loaded and this can be done either using script files named with .globals and .view/.tcl or through GUI as shown below

**The target design considered here is 4-bit counter.**

**The procedure shall remain the same for any other design from the above discussed experiments.**

**Note :**

1. For Synthesis, slow.lib was read as input. Each liberty file contains a pre-defined Process, Voltage and Temperature (PVT) values which impact the ease of charge movement.
2. Process, Voltage and Temperature individually affect the ease of currents 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 Combination applicable 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 analyse 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 form of 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

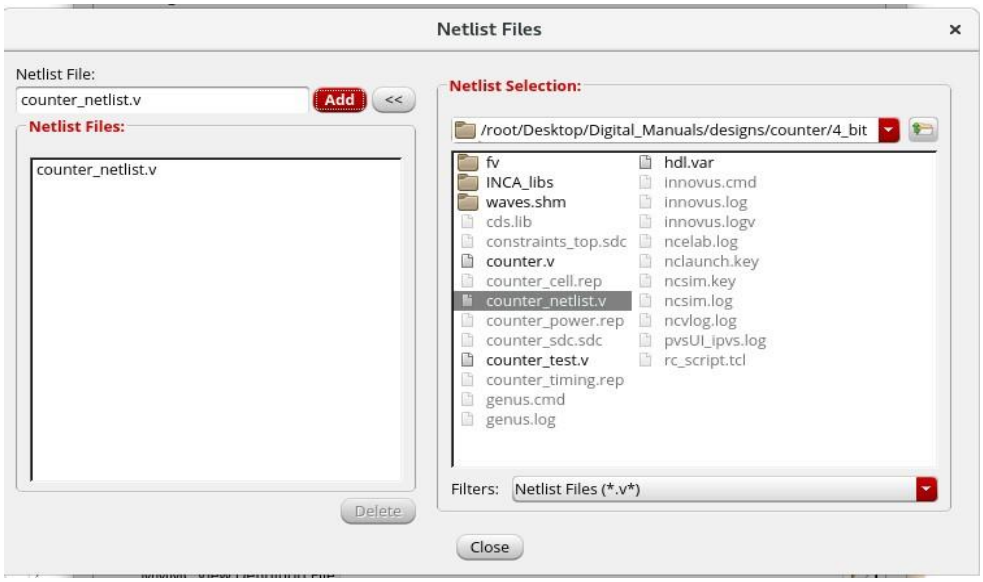
```
# Version:1.0 MMMC View Definition File
# Do Not Remove Above Line
create_library_set -name MAX_timing -timing {/root/Desktop/counter/lib/90/slow.lib}
create_library_set -name MIn_timing -timing {/root/Desktop/counter/lib/90/fast.lib}
create_constraint_mode -name Constraints -sdc_files {counter_sdc.sdc}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

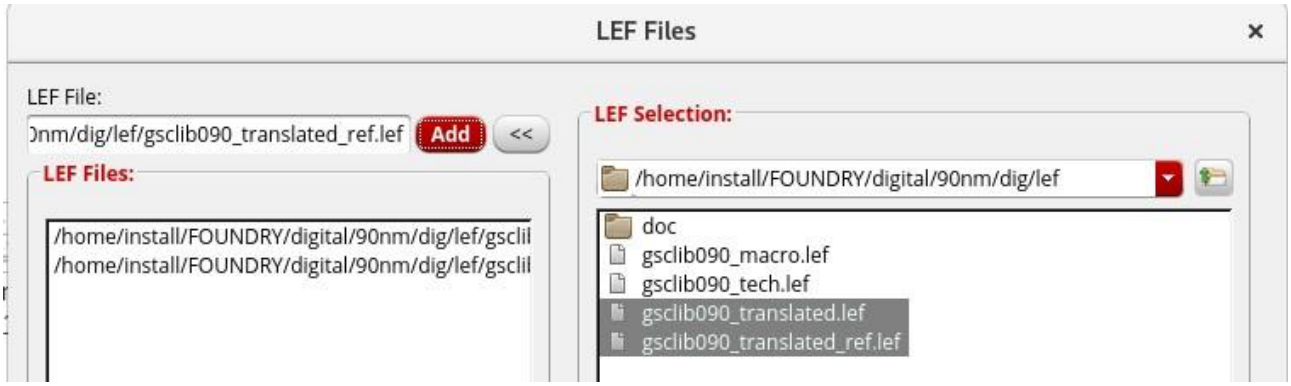
**Note :** Check the paths to properly read in the input files.

- Else, if you would like to import your design using GUI, open the Innovus 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

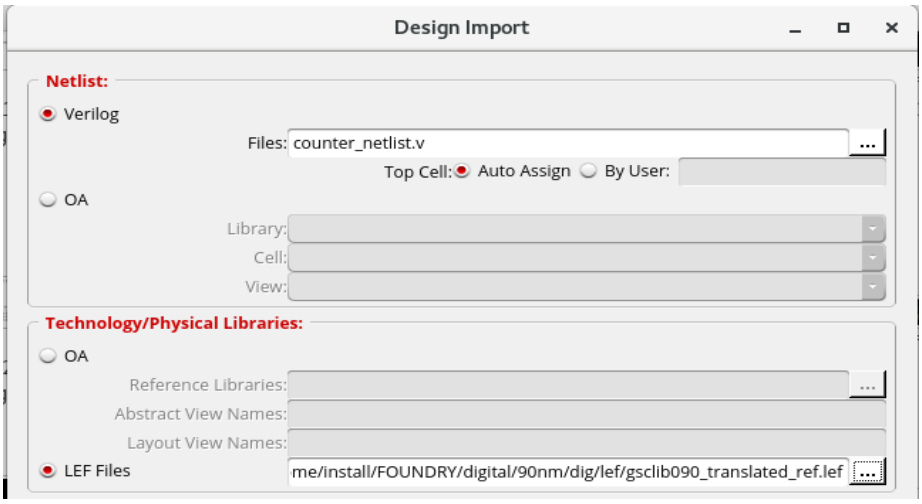
and select “Top cell : Auto Assign”.



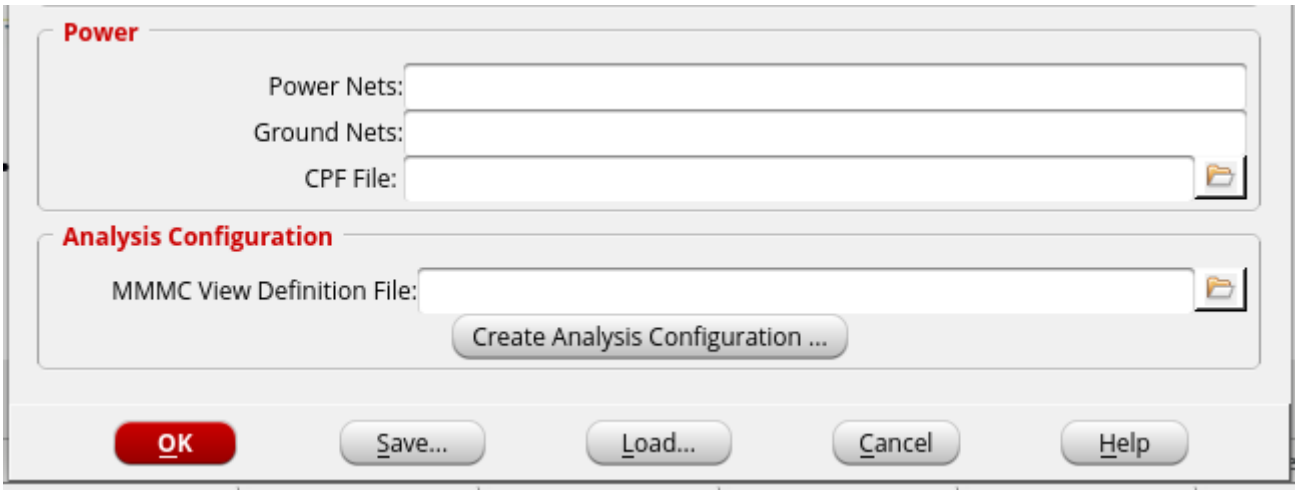

Similarly select your lef files from /home/install/FOUNDRY/digital/90nm/ dig/lef/ as shown below



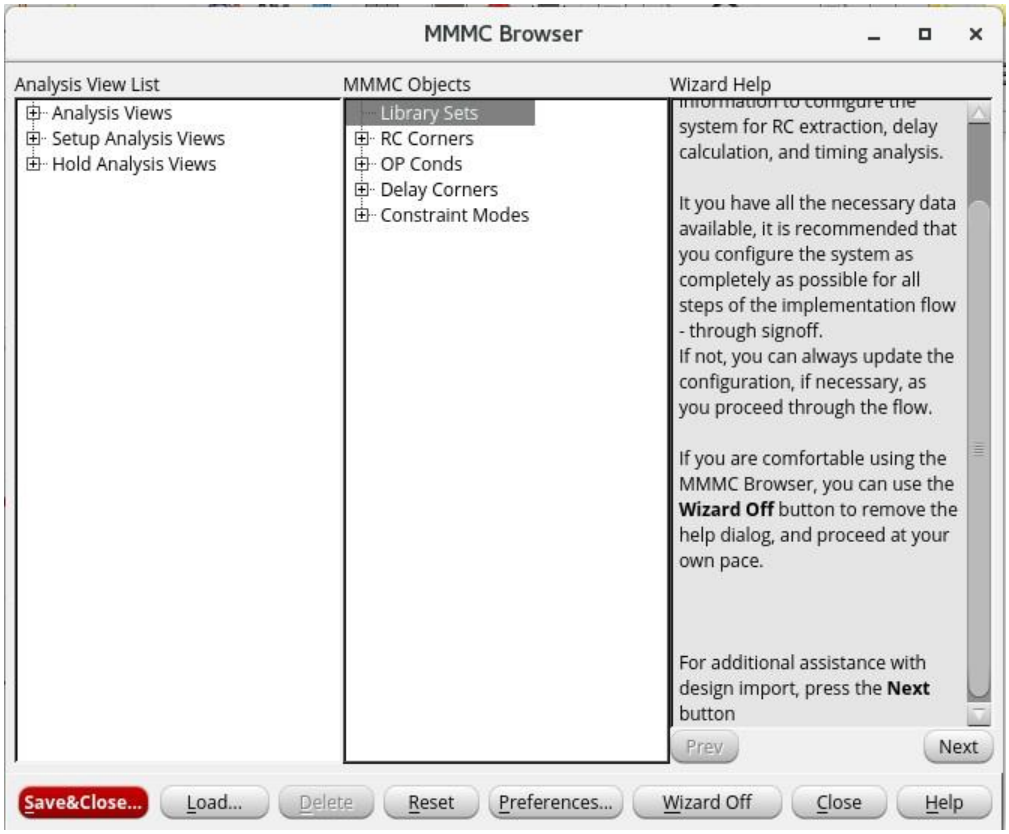

Once both the Netlist and LEF Files are loaded, your import design window is as follows.




- In order to load the Liberty File and SDC, create delay corners and analysis view, select the "Create Analysis Configuration" option at the bottom.




- An MMMC browser Pops Up.



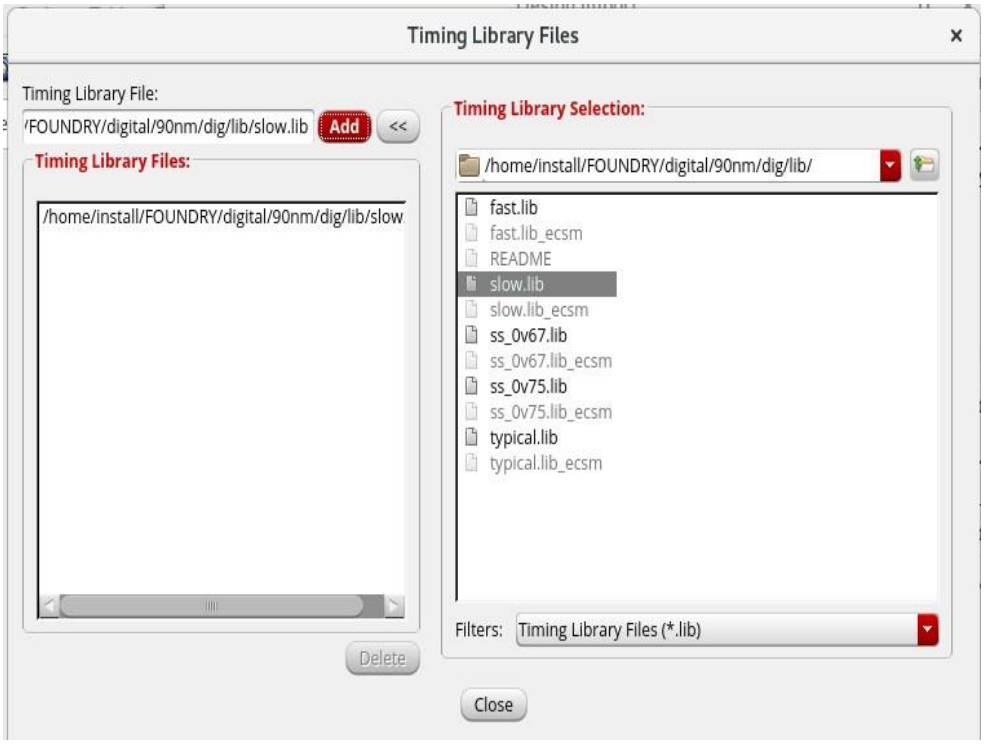

The order of adding the MMMC Objects is as follows.

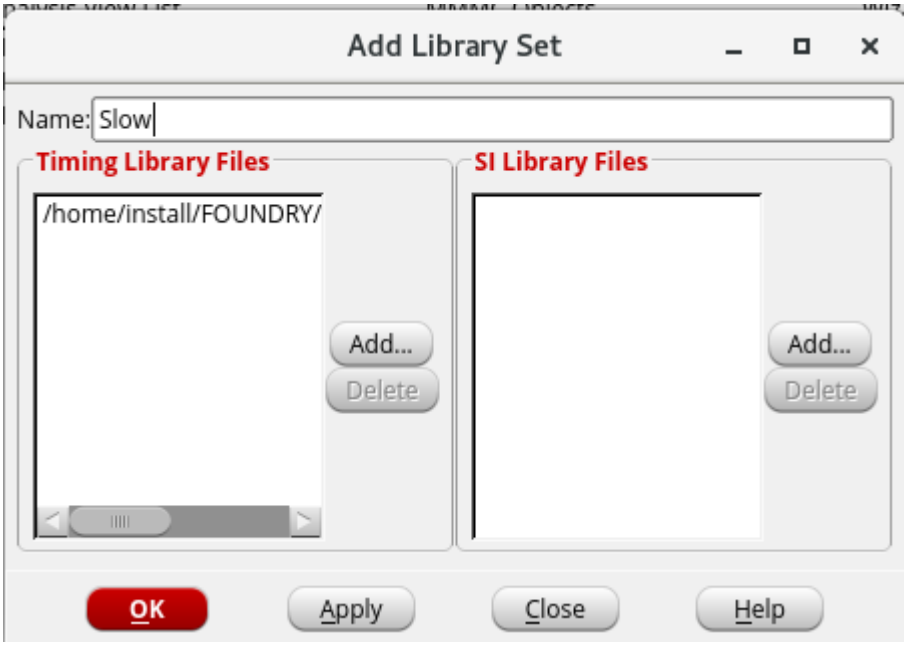
1. Library Sets
2. RC Corners
3. Delay Corners
4. Constraints (SDC)

Once all of them are added, Analysis Views are created and assigned to Setup and Hold.

In order to add any of the objects, make a right click on the corresponding label → Select New.

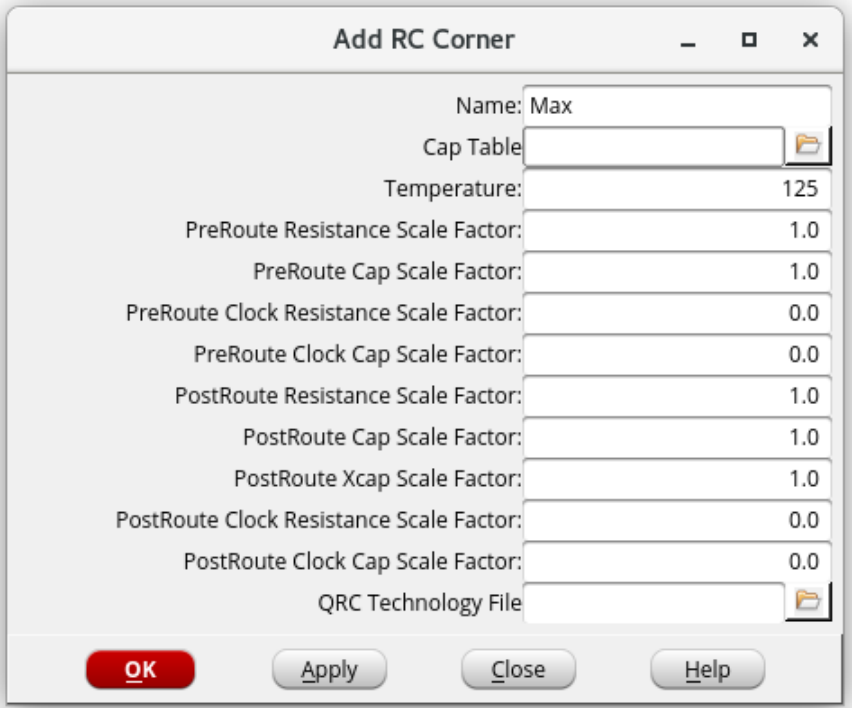
- Adding Liberty Files under “Library Sets”

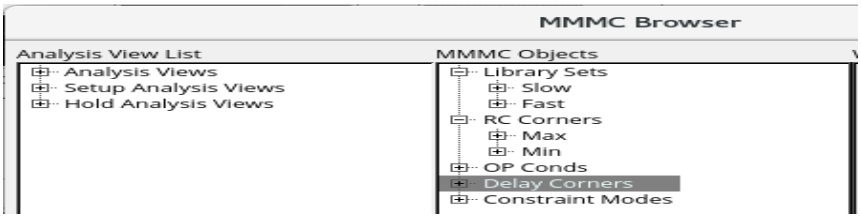

Timing Library Files
Timing Library File:
/FOUNDRY/digital/90nm/dig/lib/slow.lib
Add
<<
Timing Library Files:
/home/install/FOUNDRY/digital/90nm/dig/lib/slow
Delete
Timing Library Selection:
/home/install/FOUNDRY/digital/90nm/dig/lib/
fast.lib
fast.lib_ecsm
README
slow.lib
slow.lib_ecsm
ss_0v67.lib
ss_0v67.lib_ecsm
ss_0v75.lib
ss_0v75.lib_ecsm
typical.lib
typical.lib_ecsm
Filters:
Timing Library Files (*.lib)
Close


Add Library Set
Name:
Slow
Timing Library Files
/home/install/FOUNDRY/
Add...
Delete
SI Library Files
Add...
Delete
OK
Apply
Close
Help

- Similarly, add fast.lib with a label Fast or any identifier 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 where available.

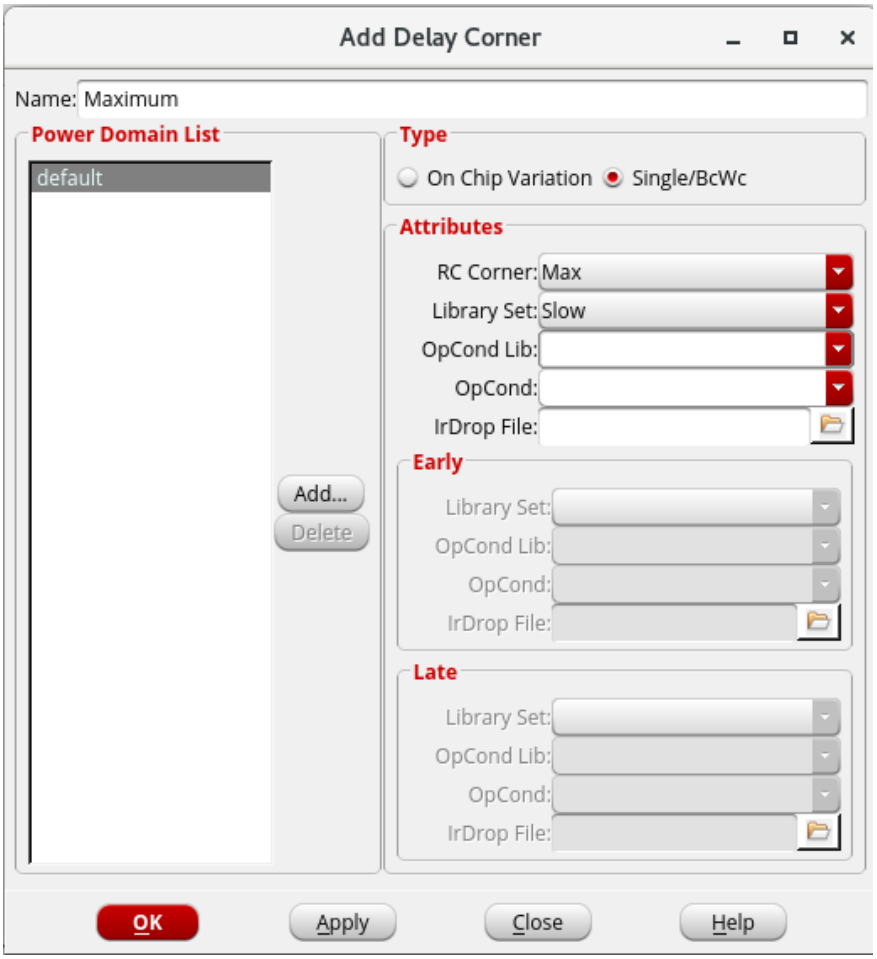


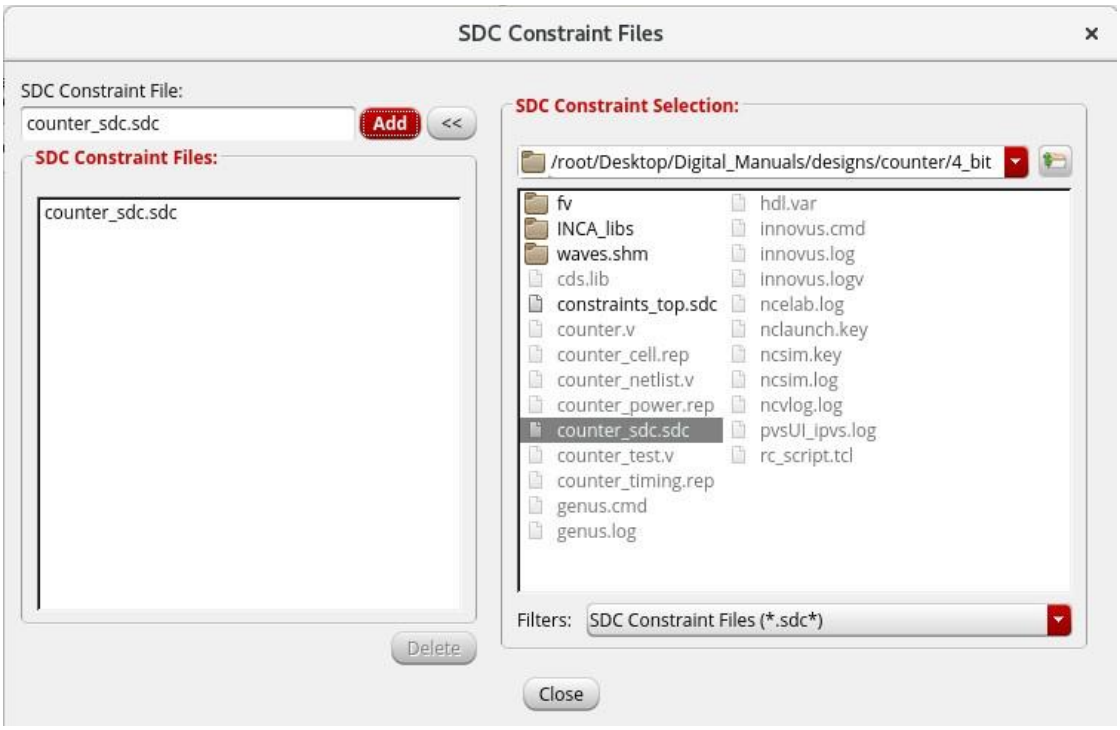

- Delay Corners are formed by

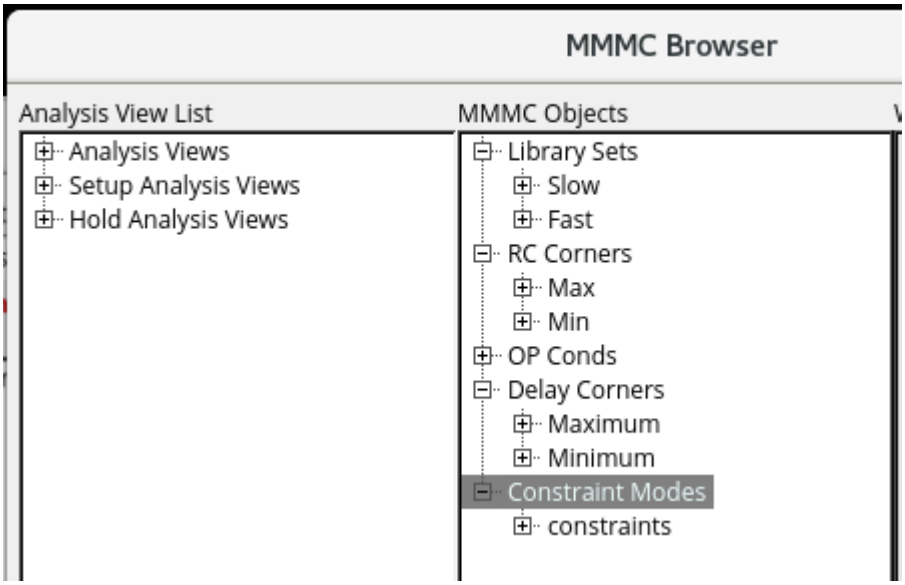
combining Library Sets with RC Corners.

- An example is shown below.

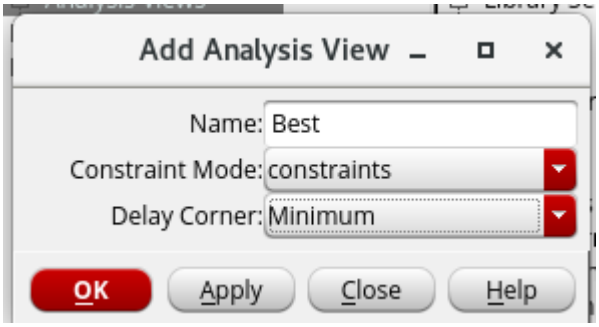



- Similarly, SDC can be read in under the MMMC Object of "Constraints".

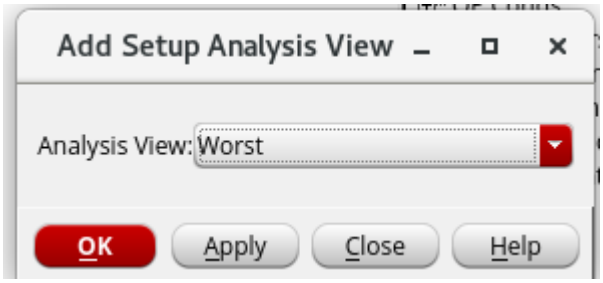

SDC Constraint Files
SDC Constraint File:
counter_sdc.sdc
Add
<<
SDC Constraint Files:
counter_sdc.sdc
Delete
SDC Constraint Selection:
/root/Desktop/Digital_Manuals/designs/counter/4_bit
fv
INCA_libs
waves.shm
cds.lib
constraints_top.sdc
counter.v
counter_cell.rep
counter_netlist.v
counter_power.rep
counter_sdc.sdc
counter_test.v
counter_timing.rep
genus.cmd
genus.log
hdl.var
innovus.cmd
innovus.log
innovus.logv
ncelab.log
nclaunch.key
ncsim.key
ncsim.log
ncvlog.log
pvsUI_ipvs.log
rc_script.tcl
Filters:
SDC Constraint Files (*.sdc*)
Close

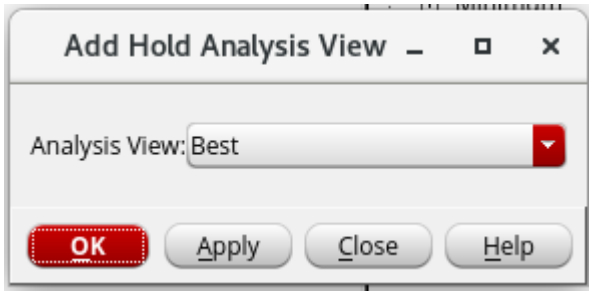

MMMC Browser
Analysis View List
Analysis Views
Setup Analysis Views
Hold Analysis Views
MMMC Objects
Library Sets
Slow
Fast
RC Corners
Max
Min
OP Conds
Delay Corners
Maximum
Minimum
Constraint Modes
constraints

- Analysis Views are formed from combinations of SDC and Delay Corner.



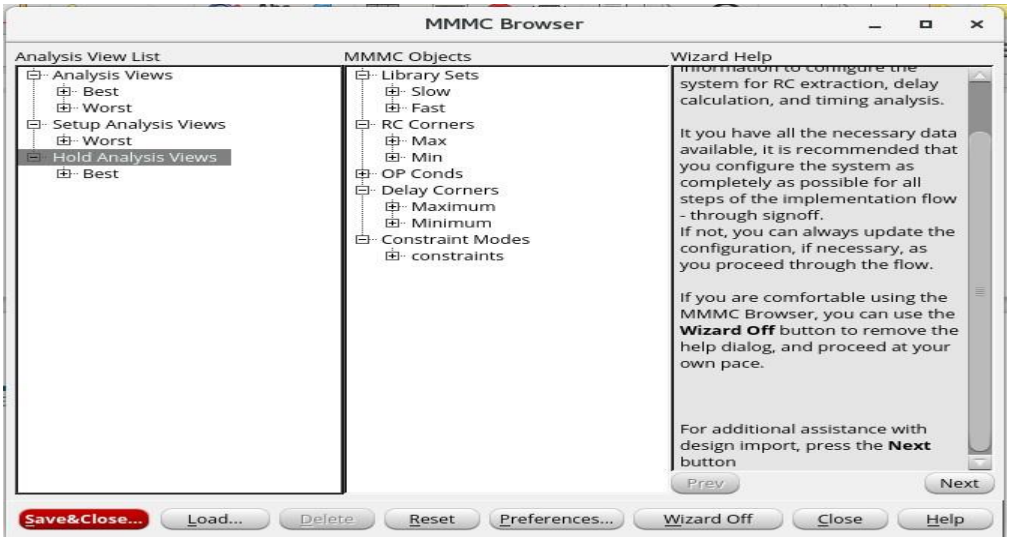

- Once “Best” and “Worst” Analysis views are created, assign them to Setup and Hold.

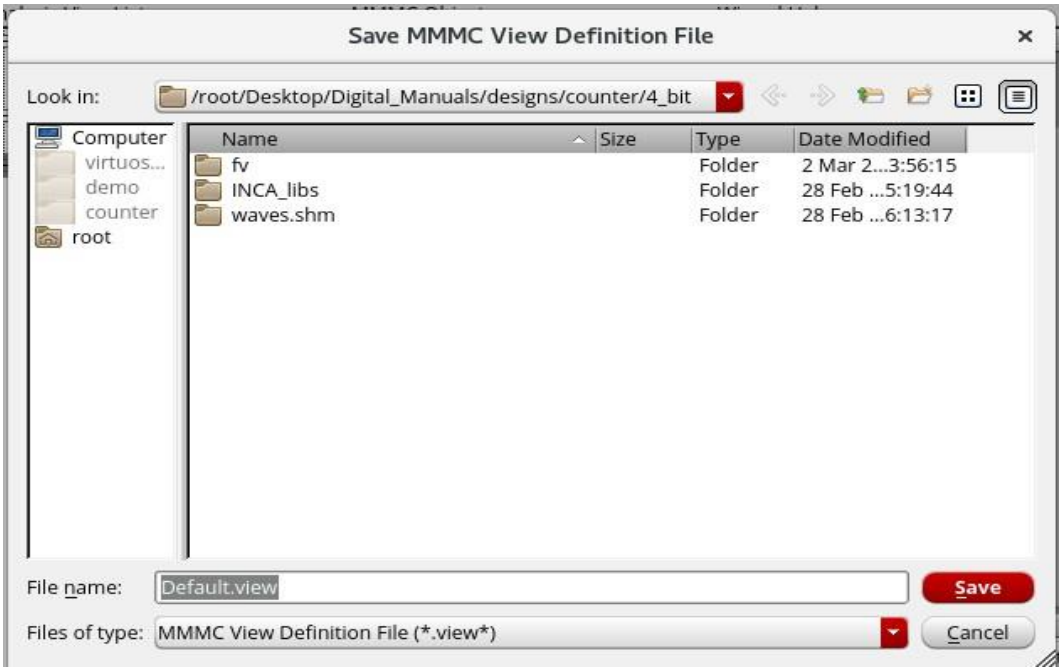


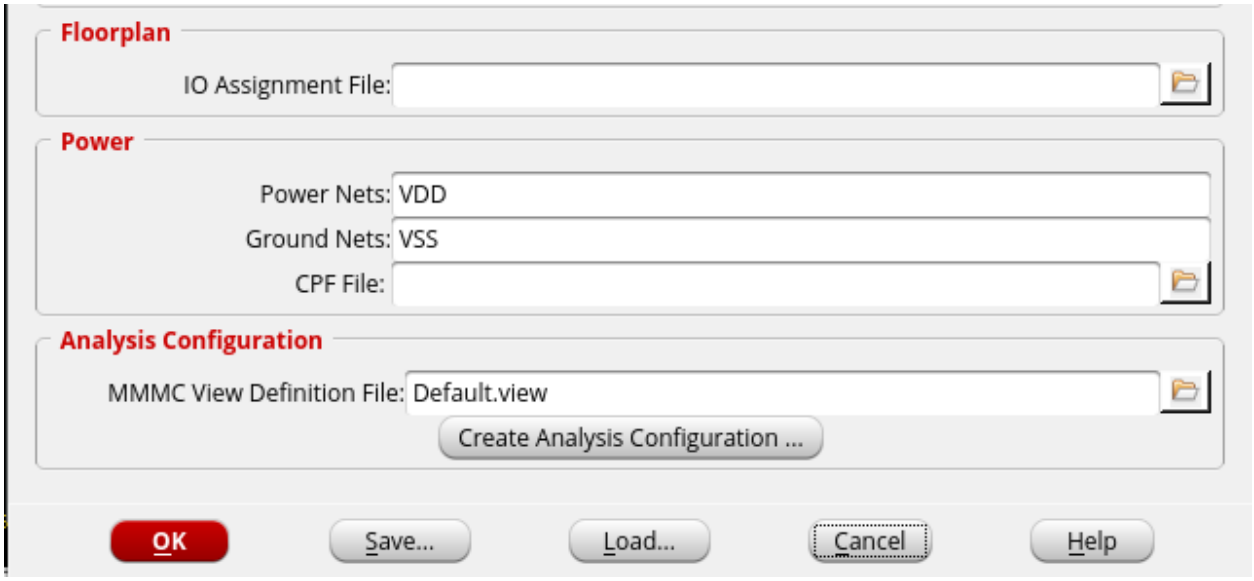
- Once all the process is done, Click on “Save&Close” and save the script generated with any name of your choice.

- 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.

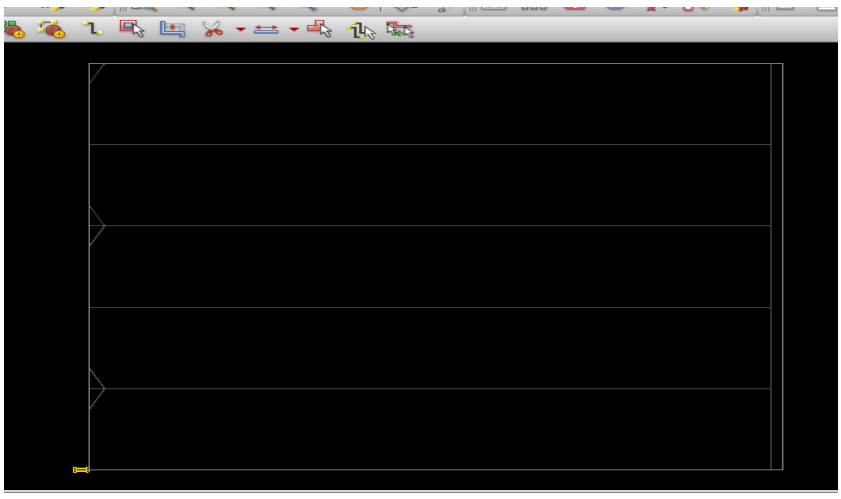


- Add Power and Ground Net names (Identifiers) under Import design window.




- A rectangular or square box appears in your GUI if and only if all the inputs are read properly If the box does not appear, check for errors in your log (Either on terminal or log file from pwd)



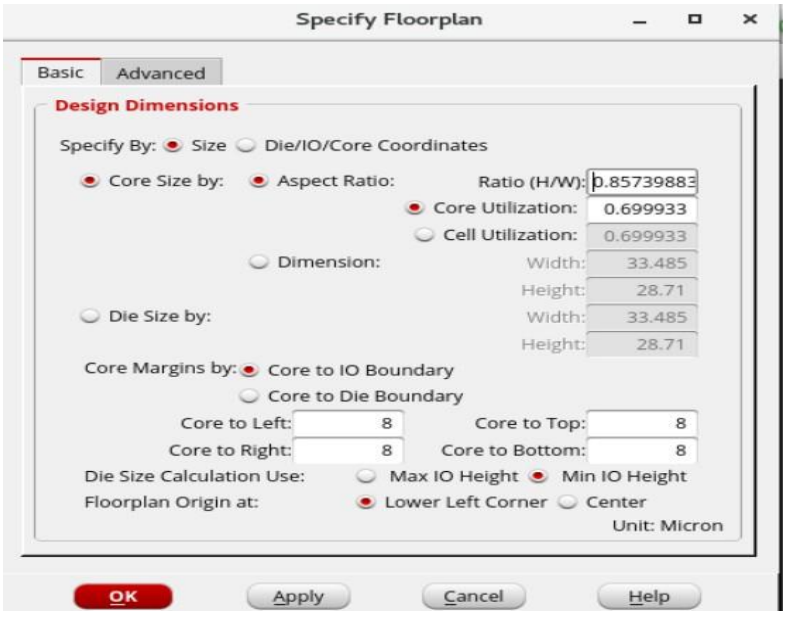
- The internal area of the box is called “Core Area”.
- The horizontal lines running along the width of Core are “Standard Cell Rows”. Every alternate of them are marked indicating alternate VDD and VSS rows.
- This setup is called “**Flipped Standard Cell Rows**”.

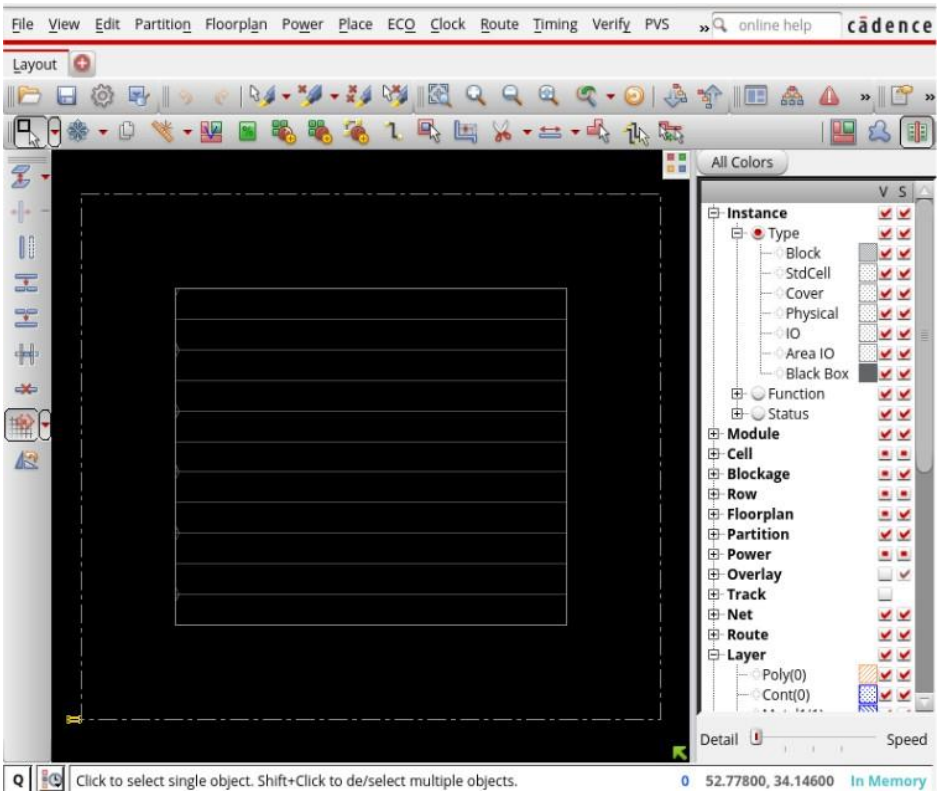
**→ Floorplan**

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 Floor Planning]
3. Channel Spacing between Core Boundary to IO Boundary

- Select Floorplan → Specify Floorplan to modify/add concerned values to the above Factors. On adding/modifying the concerned values, the core area is also modified.






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

**→ Power Planning**

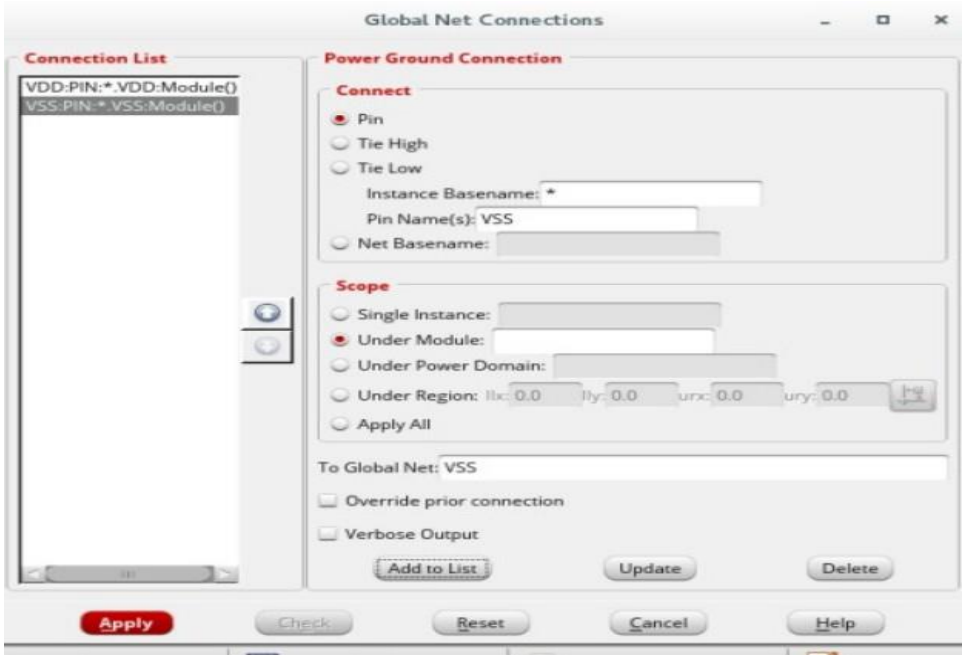
Steps under Power Planning :

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 VDD and one for VSS connecting them to corresponding Global Nets as mentioned in Globals file /

Power and Ground Nets.

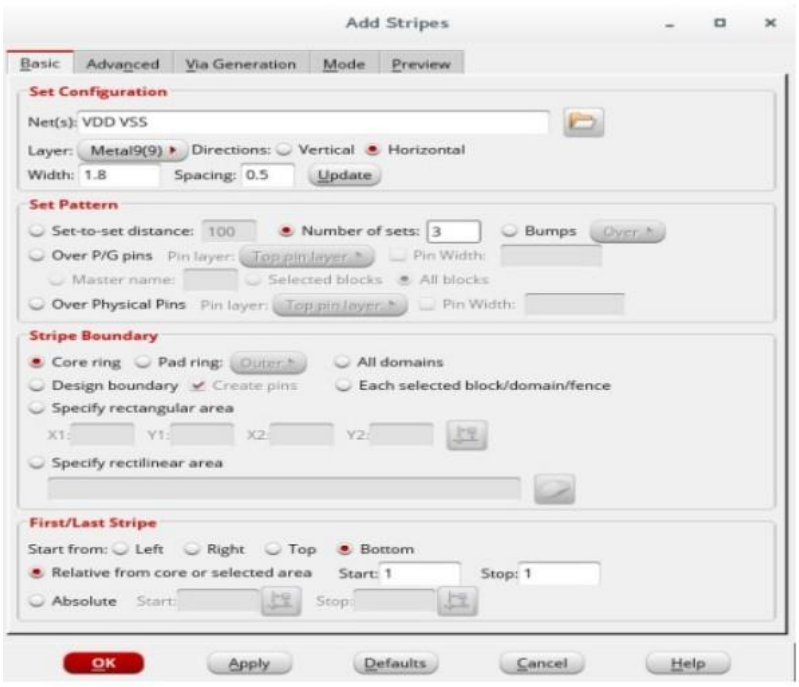
1. Select Power → Connect Global Nets.. to create “Pin” and “Connect to Global Net” as shown and use “Add to list”.
2. Click on “Apply” to direct the tool in enforcing the Pins and Net connects to Design and then Close the window.




- In order to Tap in Power from a distant Power supply, Wider Nets and Parallel connections improve efficiency. Moreover, the cells that would be placed inside the core area are expected to have shorter Nets for lower resistance.
- Hence Power Rings [Around Core Boundary] and Power Stripes [Across Core Boundary] are added which satisfies the above conditions.
- Select Power → Power Planning → Add Rings to add Power rings ‘around Core Boundary’.


- Select the Nets from Browse option OR Directly type in the Global Net 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 Widths and thus Lowest Resistance.
- Click on Update after the selection and “Set Offset : Centre in Channel” in order to get the Minimum Width and Minimum Spacing of the corresponding Metals and then Click “OK”.
- Similarly, Power Stripes are added using similar

content to that of Power Rings.



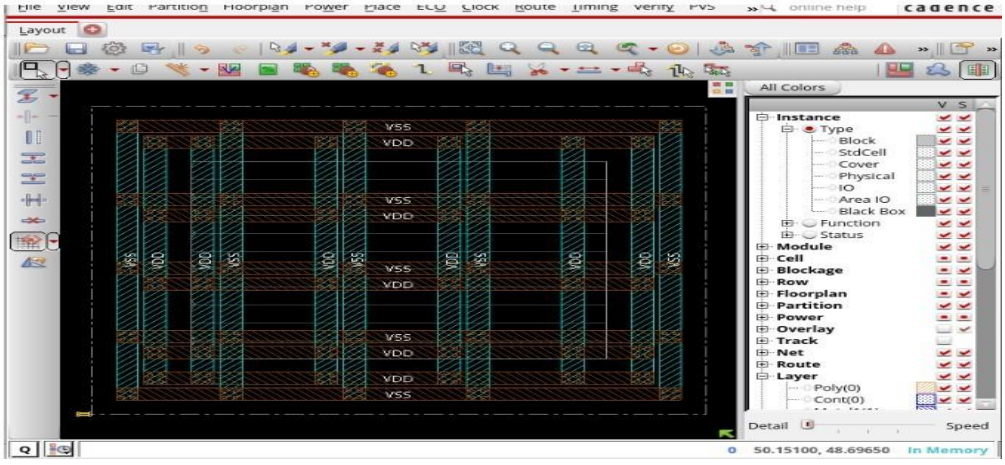

Factors to be considered under Power Stripes :

→ Nets

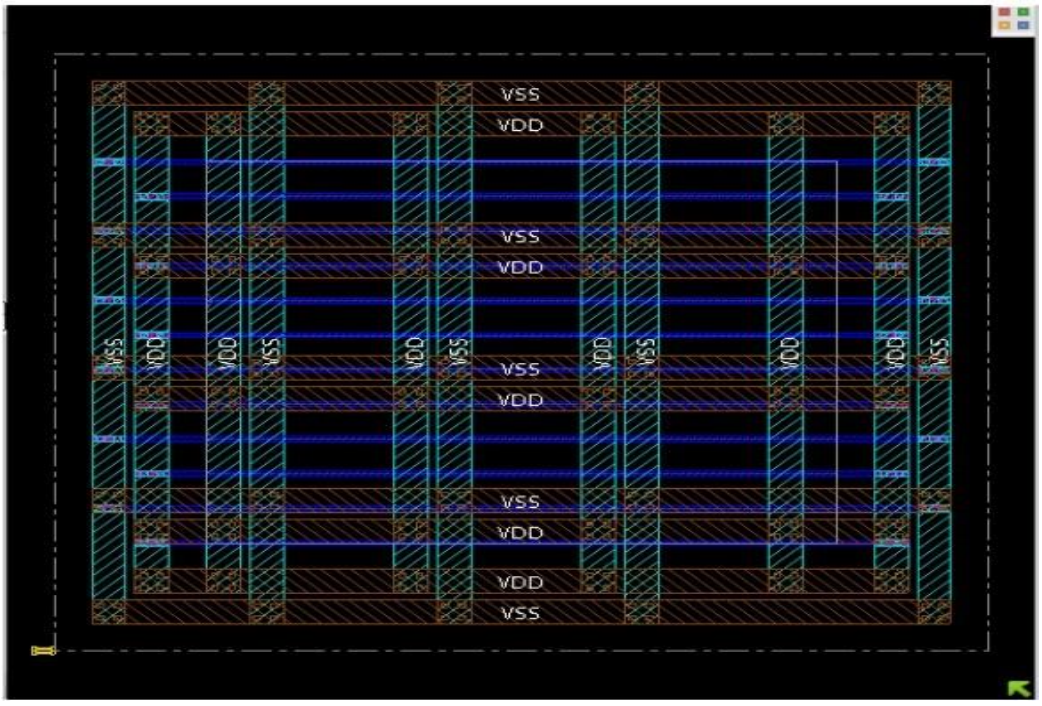
→ Metal and It's Direction

→ Width and Spacing [Updated]

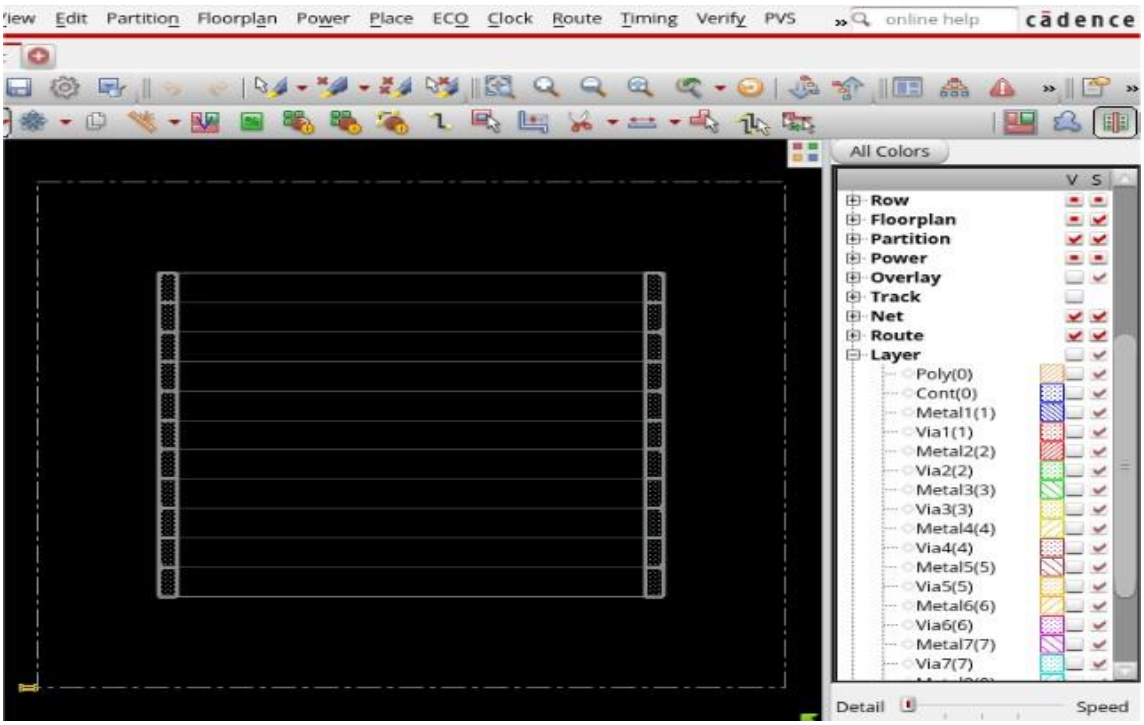
→ Set to Set Distance = ( Minimum Width of Metal + Min. Spacing ) x 2



- On adding Power Stripes, The Power mesh setup is complete as shown. However, There are no Vias that could connect Metal 9 or Metal 8 directly with Metal 1 [VDD or VSS of Standard Cells are generally made up of Metal 1].
- The connection between the Highest and Lowest Metals is done through Stacking of Vias done using “Special Route”.
- To perform Special Route, Select Route → Special Route → Add Nets → OK.
- After the Special Route is complete, all the Standard Cell Rows turn to the Color coded for Metal 1 as shown below.




The complete Power Planning process makes sure Every

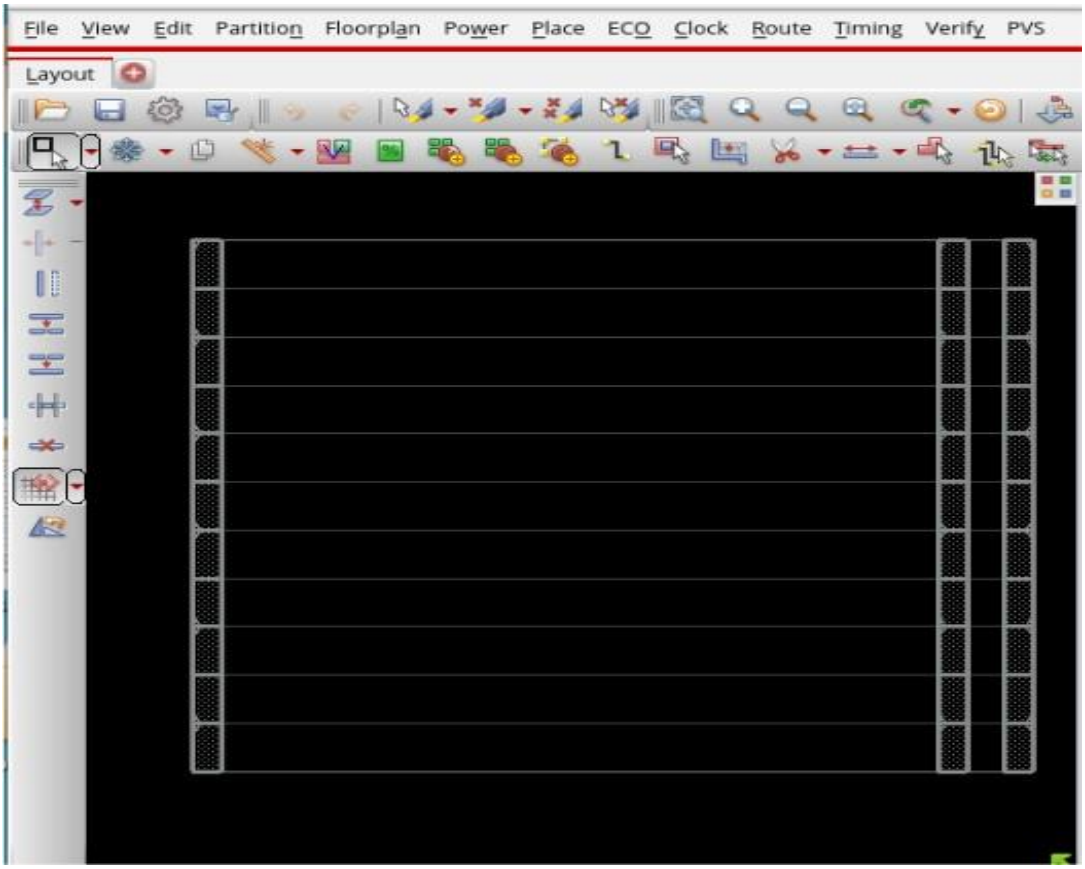



Standard Cell receives enough power to operate smoothly.

→ **Pre – Placement** :

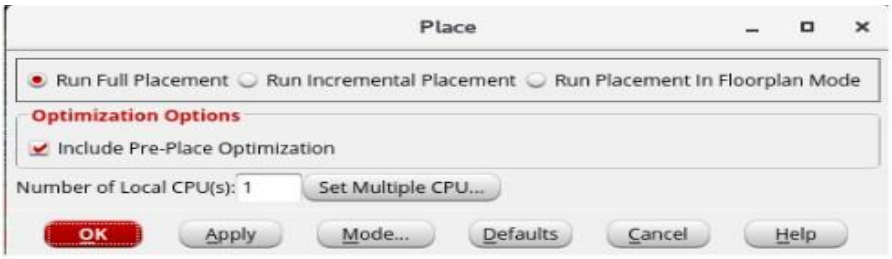
- After Power Planning, a few Physical Cells are added namely, End Caps and Well Taps.
- End Caps : They are Physical Cells which are added to the Left and Right Core Boundaries acting as blockages to avoid Standard Cells from moving out of boundary.
- Well Taps : They act like Shunt Resistance to avoid Latch Up effects.
- To add End Caps, Select Place → Physical Cell → Add End Caps and "Select" the FILL's from the available list. Higher Fills have Higher Widths. As shown Below, The End Caps are added below your Power Mesh.
- 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

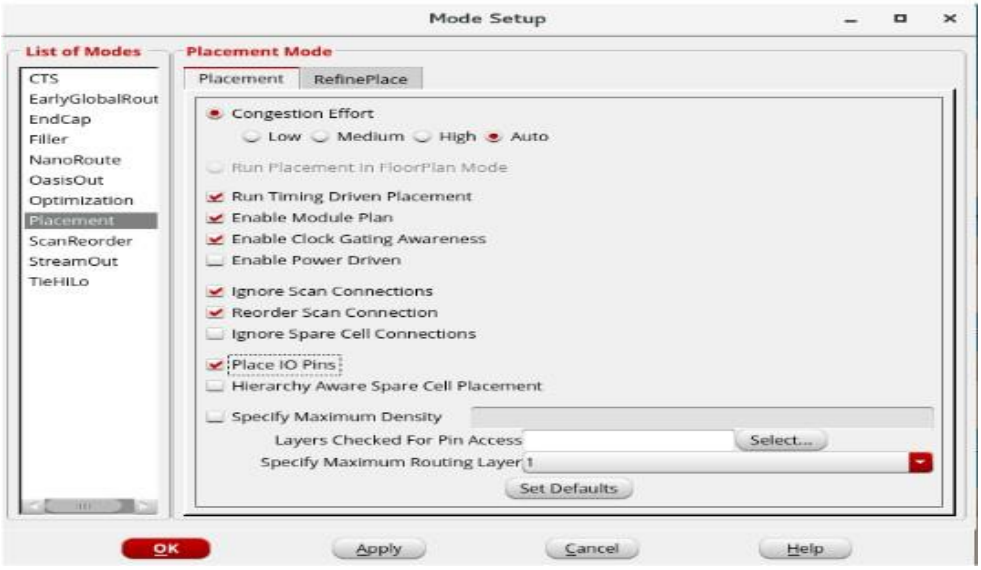
range of 30-45u] → OK



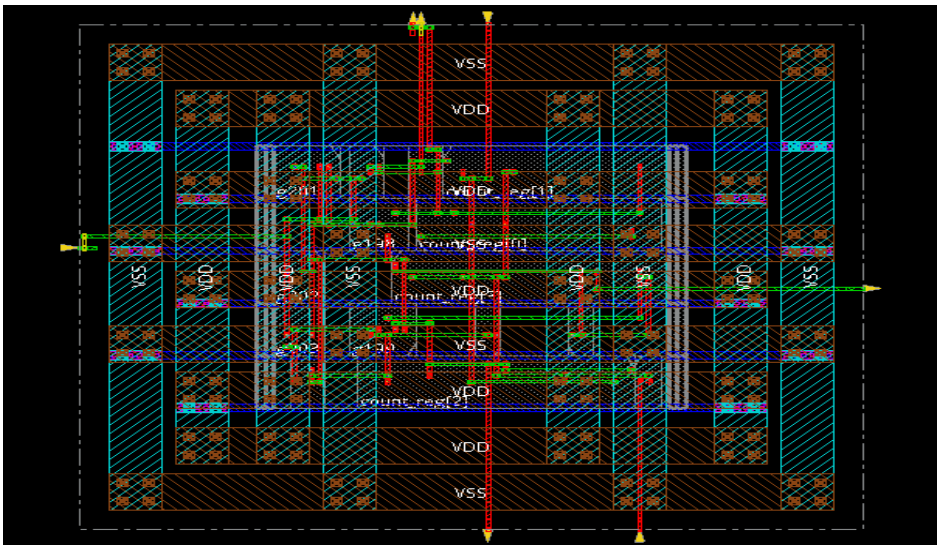

**Placement**

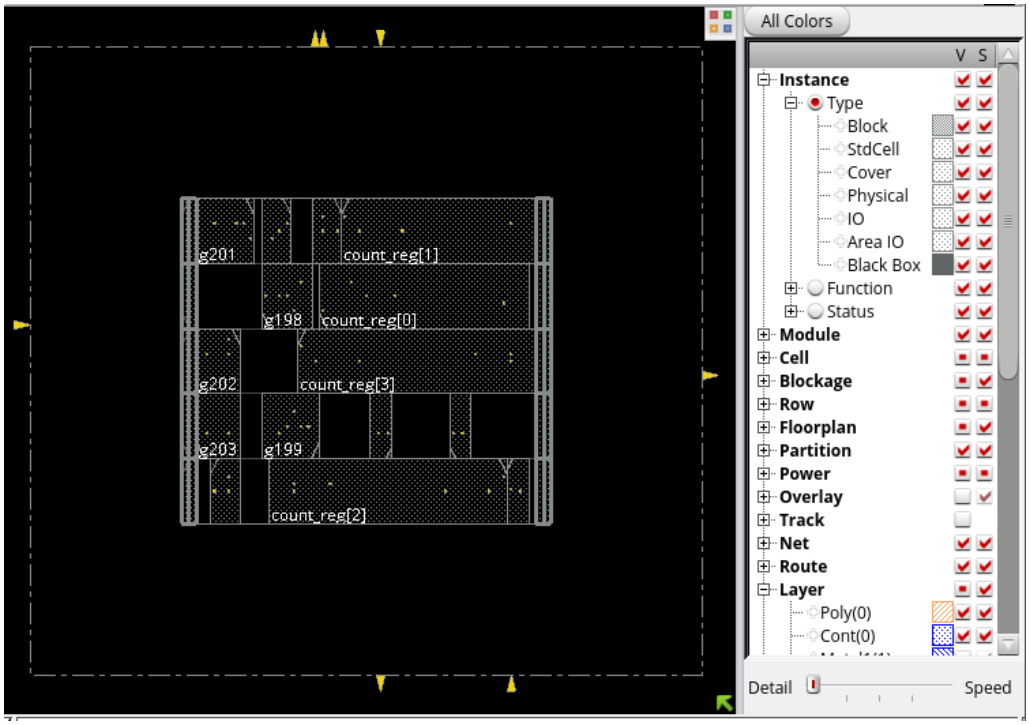
1. The Placement stage deals with Placing of Standard Cells as well as Pins.
2. Select Place → Place Standard Cell → Run Full Placement → Mode
   → Enable 'Place I/O Pins' → OK → OK .




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




- You can toggle the Layer Visibility from the list on the Right. The List of Layers available are shown on the right under “Layer” tab with colour coding.

→ Report Generation and Optimization :

1. Timing Report :
    1. To generate Timing Report, Timing → Report Timing → Design Stage – 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)       |
+----------------+------------------+------------+------------------+
```

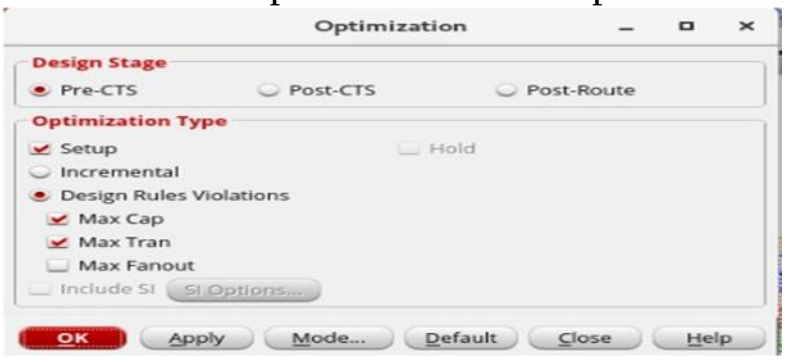
```
*       Power View : Worst
*
*       User-Defined Activity : N.A.
*
*       Activity File: N.A.
*
*       Hierarchical Global Activity: N.A.
*
*       Global Activity: N.A.
*
*       Sequential Element Activity: N.A.
*
*       Primary Input Activity: 0.200000
*
*       Default icg ratio: N.A.
*
*       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 the following way.
- To optimize the Design, **Select ECO → Optimize Design → Design Stage [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.



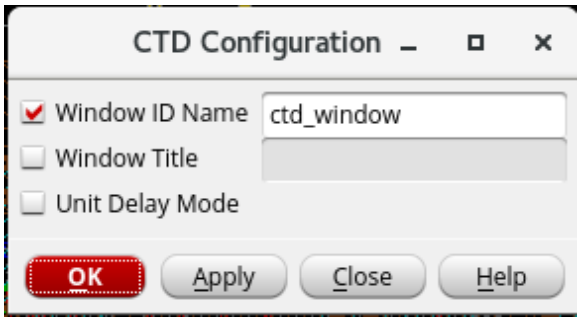

**Clock Tree Synthesis**

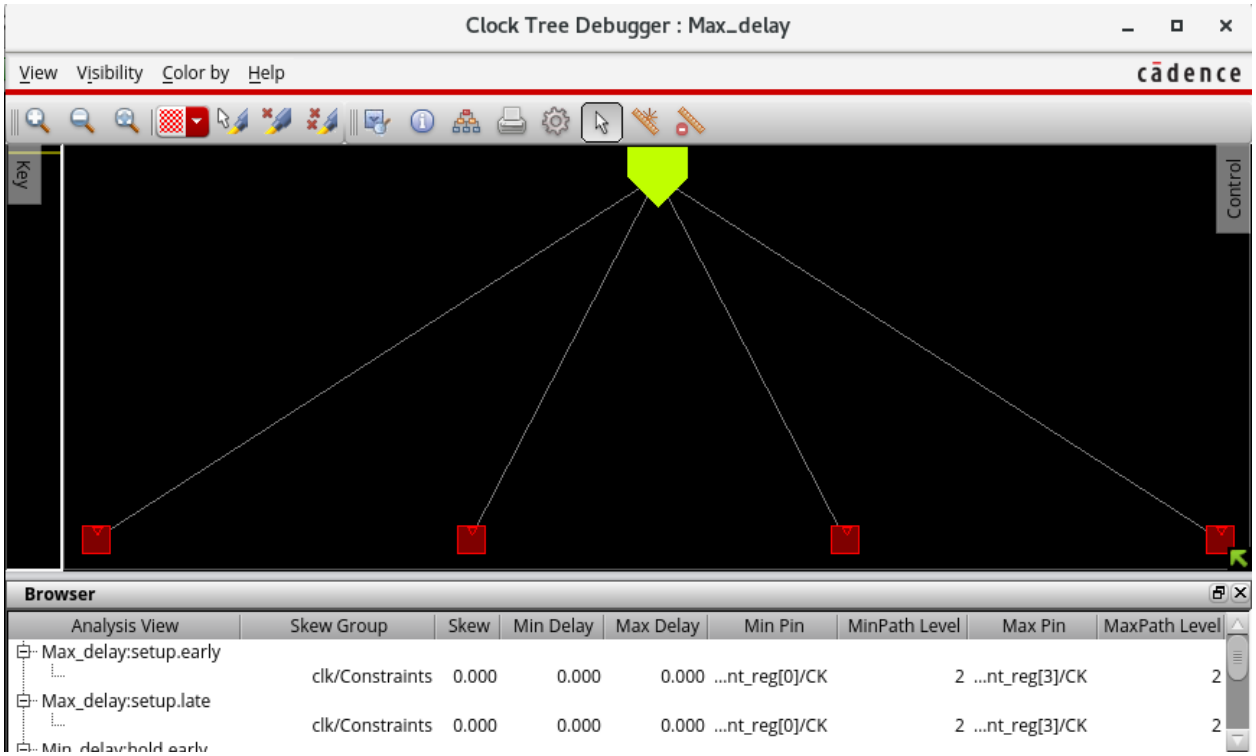
- The CTS Stage is meant to build a Clock Distribution Network such that every Register (Flip Flop) acquires Clock at the same time (Atleast Approximately) to keep them in proper communication.
- A Script can be used to Build the Clock Tree as follows :

```
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

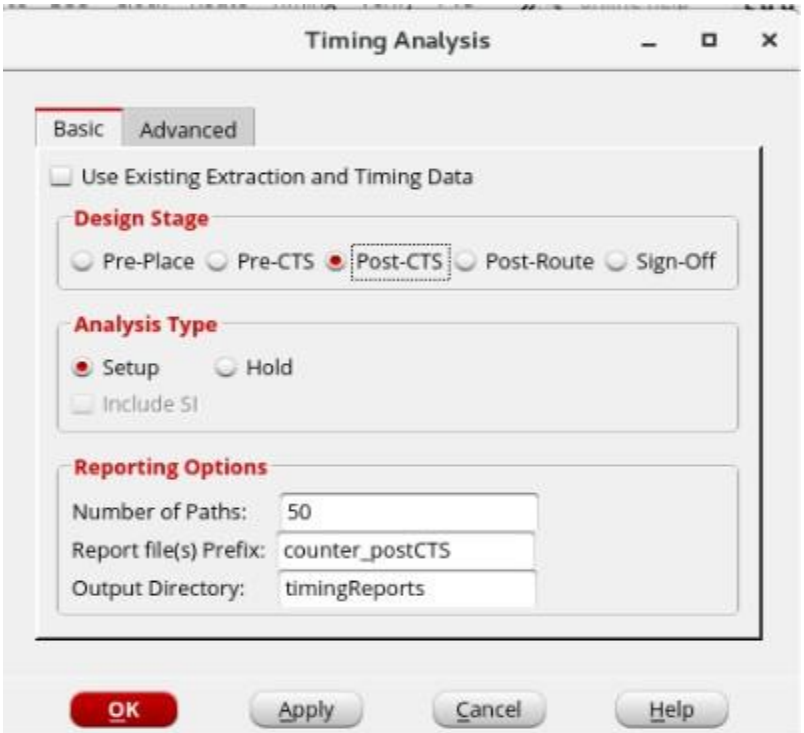
top represents Clock Source.

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

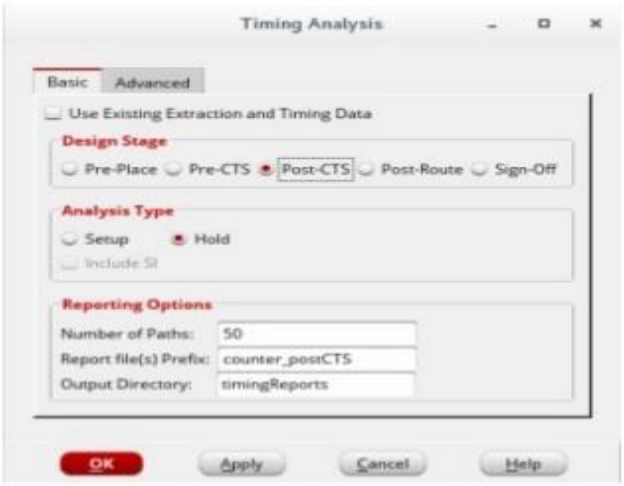
Report Generation and Design Optimization :

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

Setup Timing Analysis :



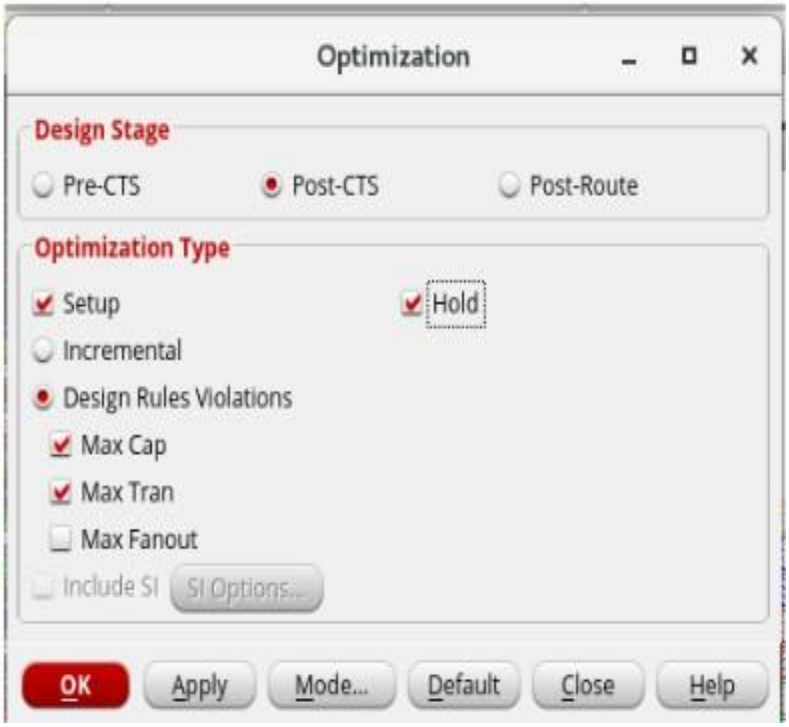
## Hold Timing Analysis :




For Area and Power Report Generation,

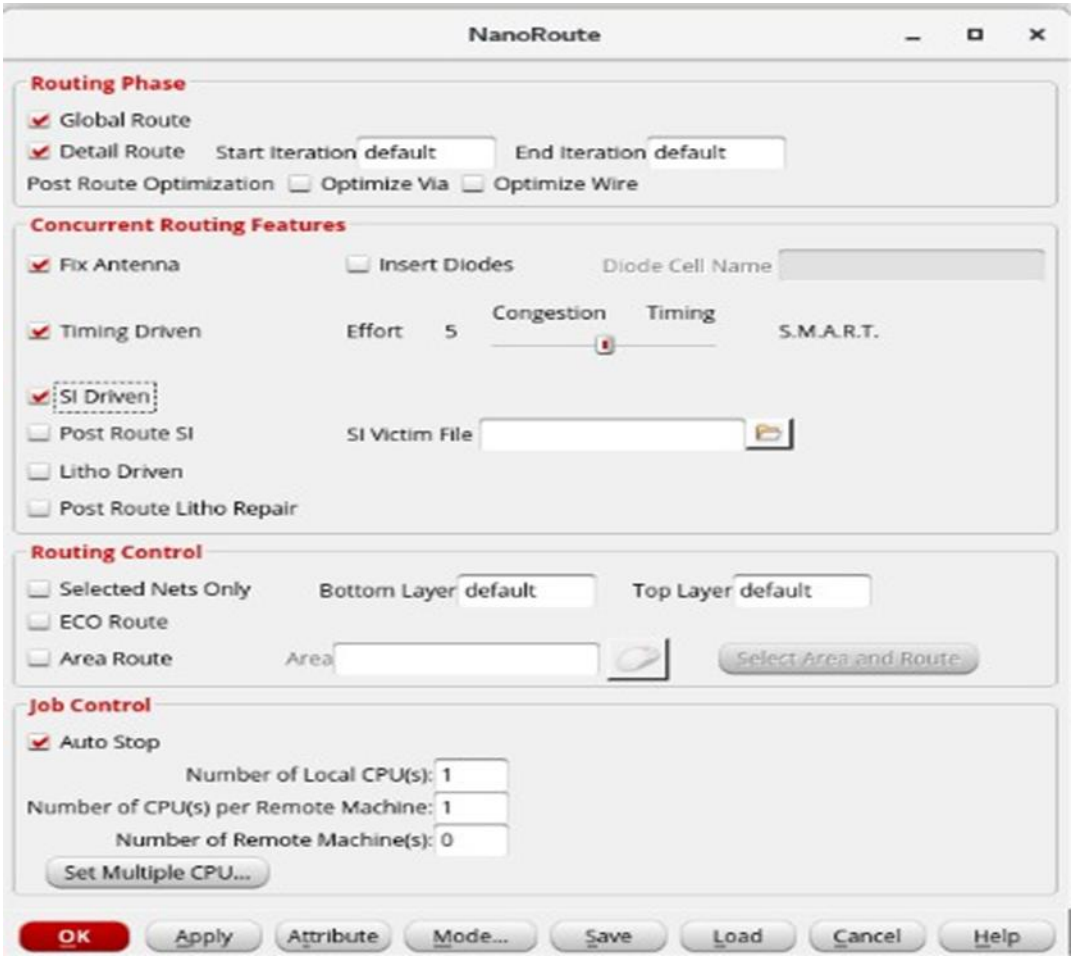
**report_area & report_power** commands can be used.

**Design Optimizations :**



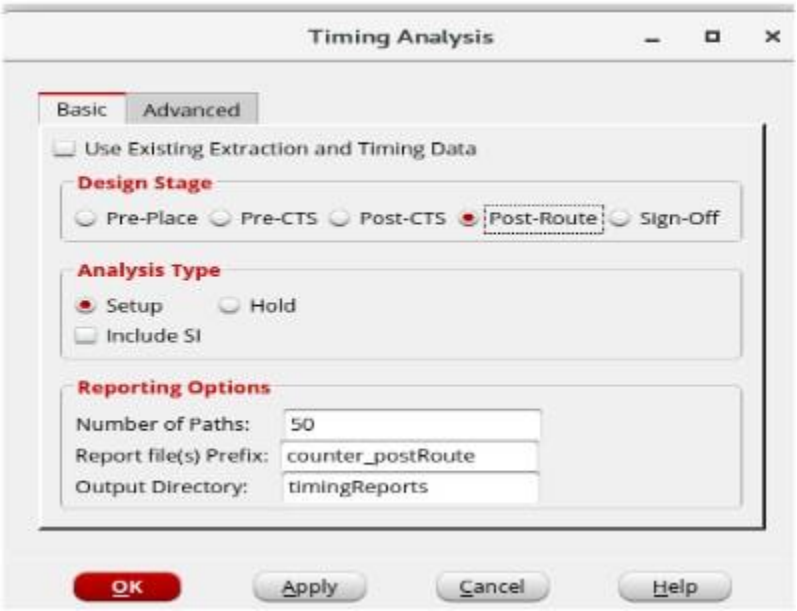
**Routing :**

1. All the net connections shown in the GUI till CTS are only based on the 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 enable Timing Driven and SI Driven for Design Physical Efficiency and Reliability.

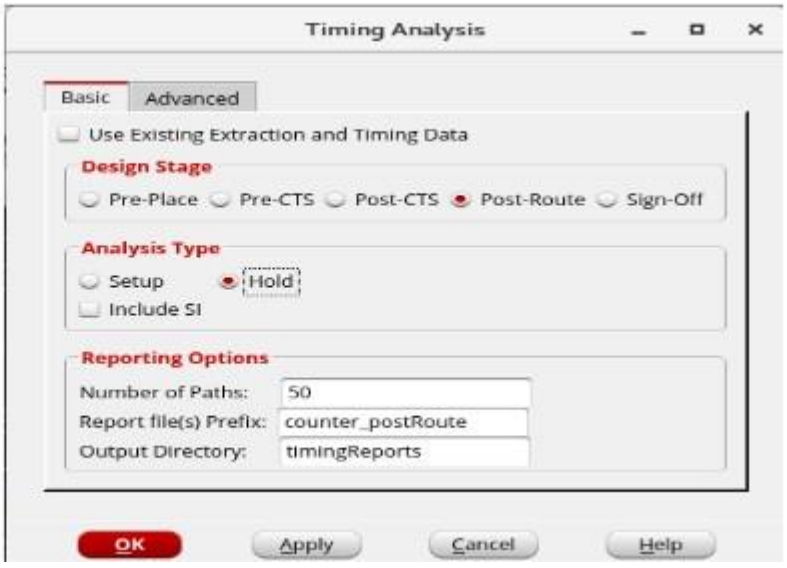


## Report Generation and Design Optimization :

### Setup Report :




### Hold Report :




**Area and Power Reports :**

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

**Design Optimization :**

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

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

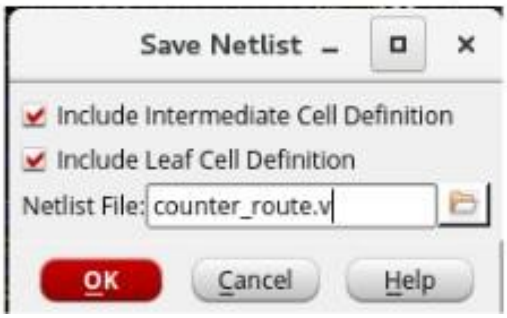

- As an alternate to the setAnalysisMode command, you can use the GUI at Tools → Set Mode → Set Analysis Mode → Select On-Chip- Variation and CPPR.

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

**Saving Database :**

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


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

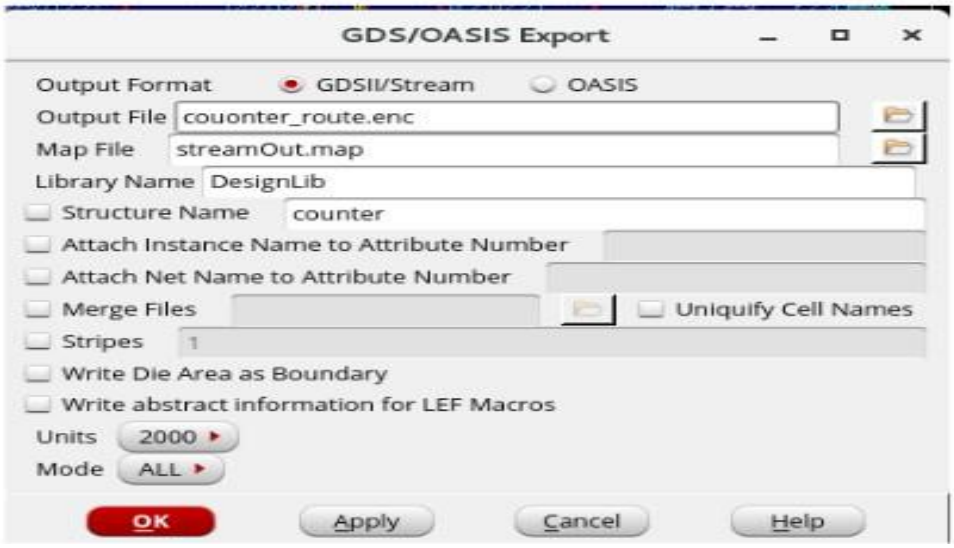



**It is recommended to save Netlist and Design at every stage.**

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

## GDS File Generation

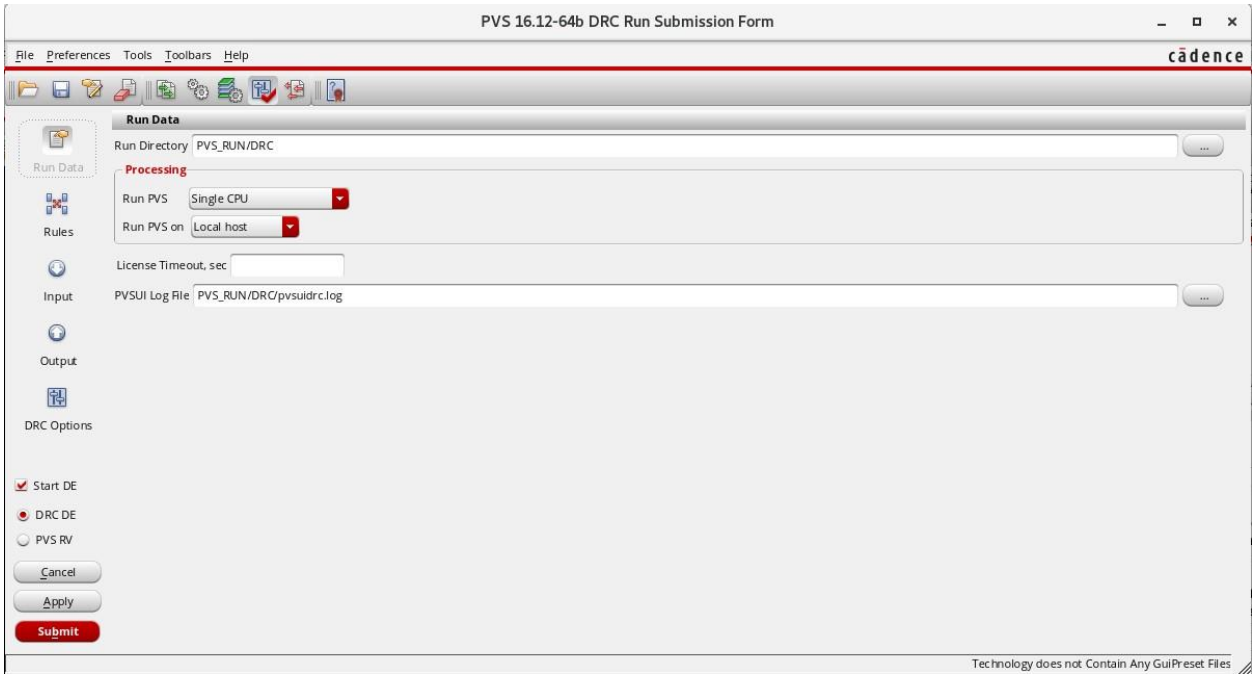
3. Saving GDS => File → Save → GDS/OASIS → <FileName>.gds → OK



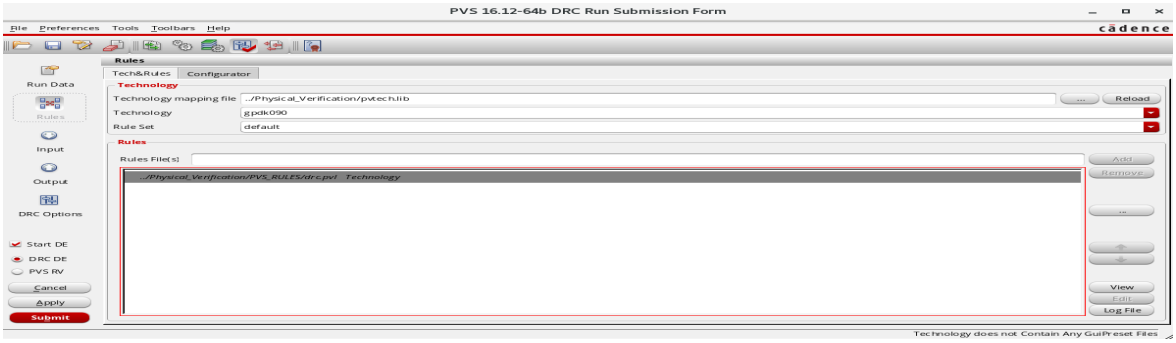

## Physical Verification – Capturing DRC and LVS :

- After saving the routed Database, you can proceed for Physical Verification 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 "DRC Submission Form".

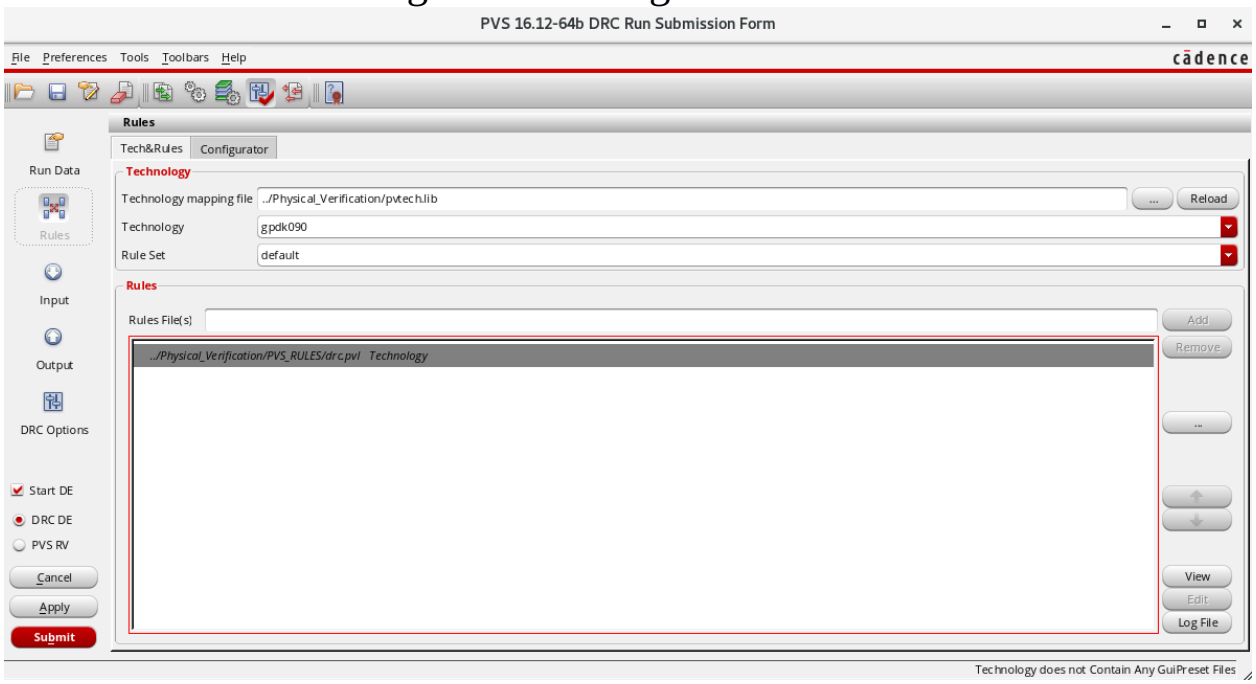



The DRC Run Submission Form begins with mentioning the Run Directory. The Run Directory is the location where all the logs, reports and other files concerned with PVS are saved.

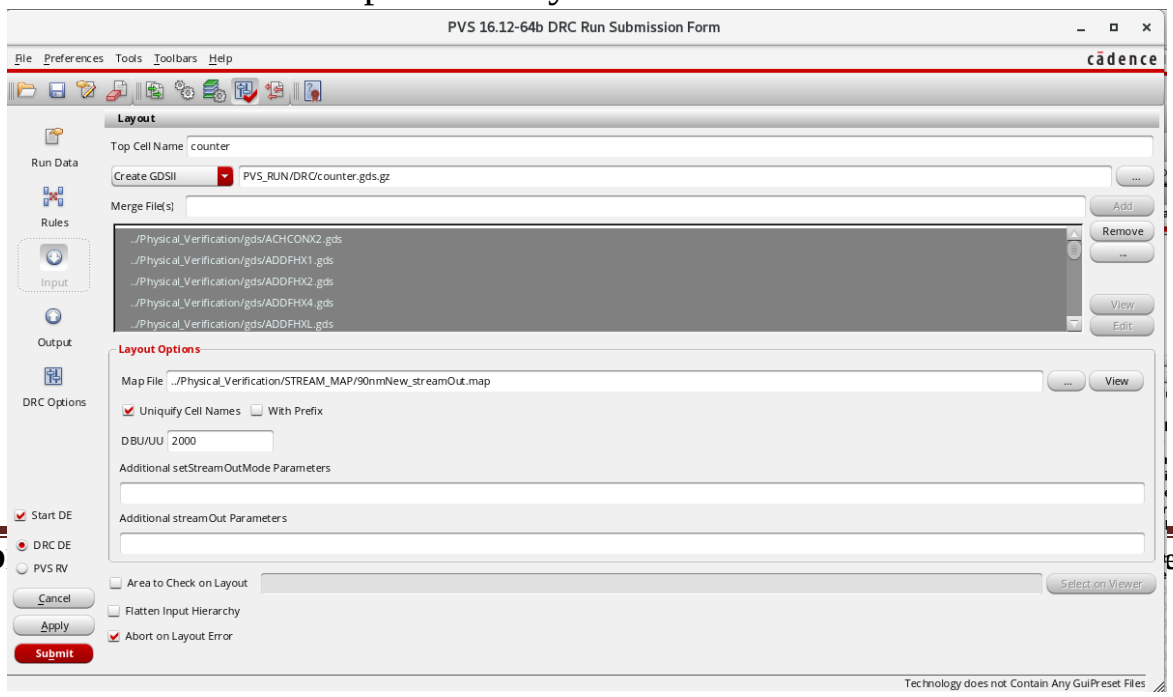



The technology Library is to be loaded under “Rules tab”.

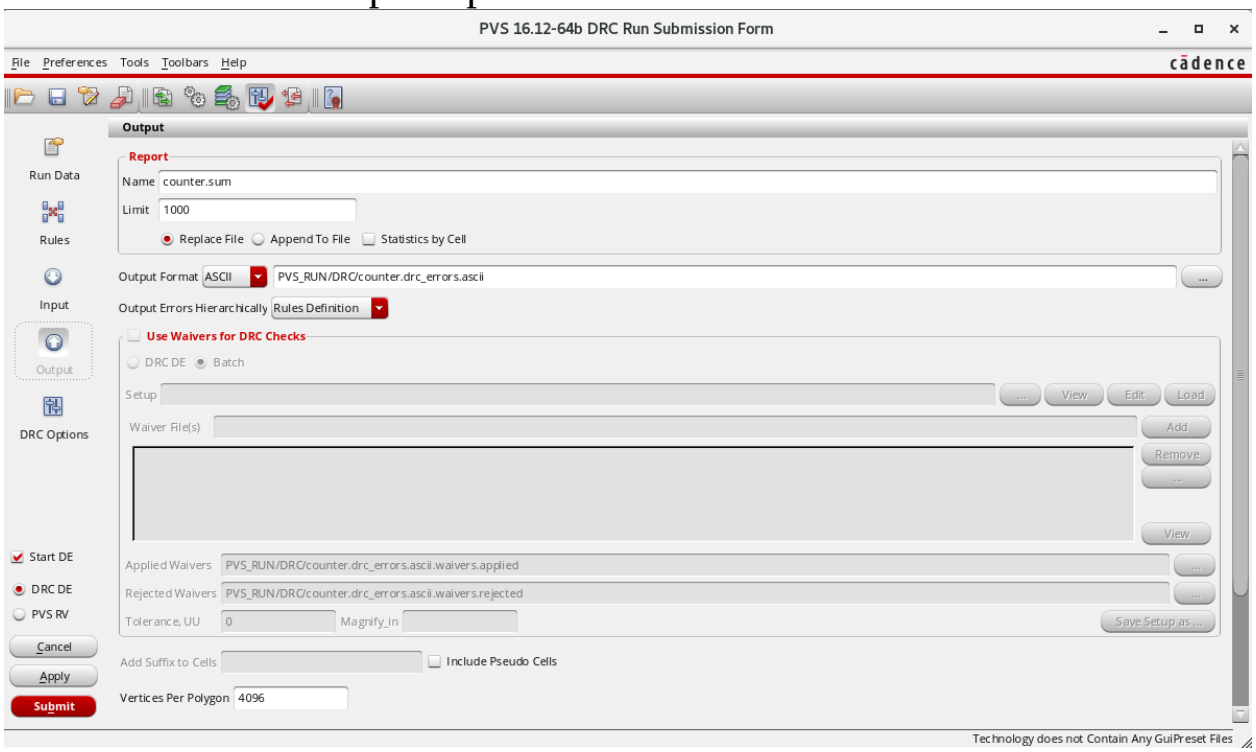
- The Technology Library is specific for PVS Tool and technology node on which the design is created.
- On reading the tech lib, the rule set is loaded and the corresponding fabrication rules are read in to be checked against the design.



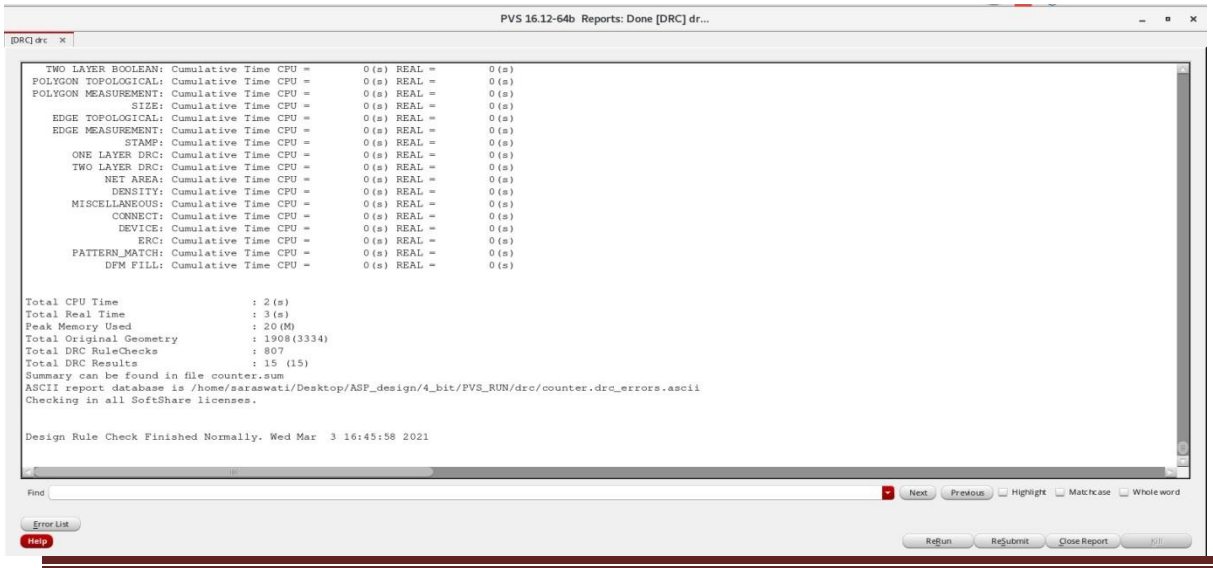

- The GDS format files of all standard cells available with the corresponding technology node are also provided by the vendor. Select all of them

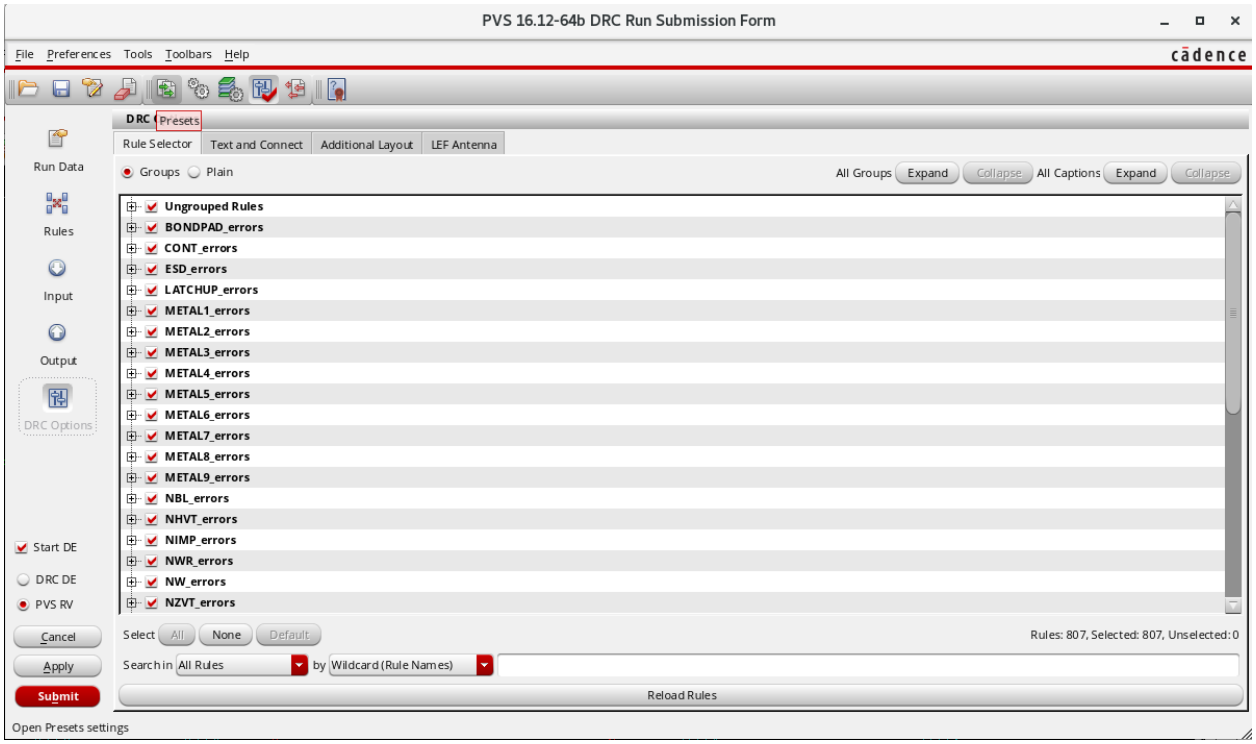


- The output report can be named and saved as shown.

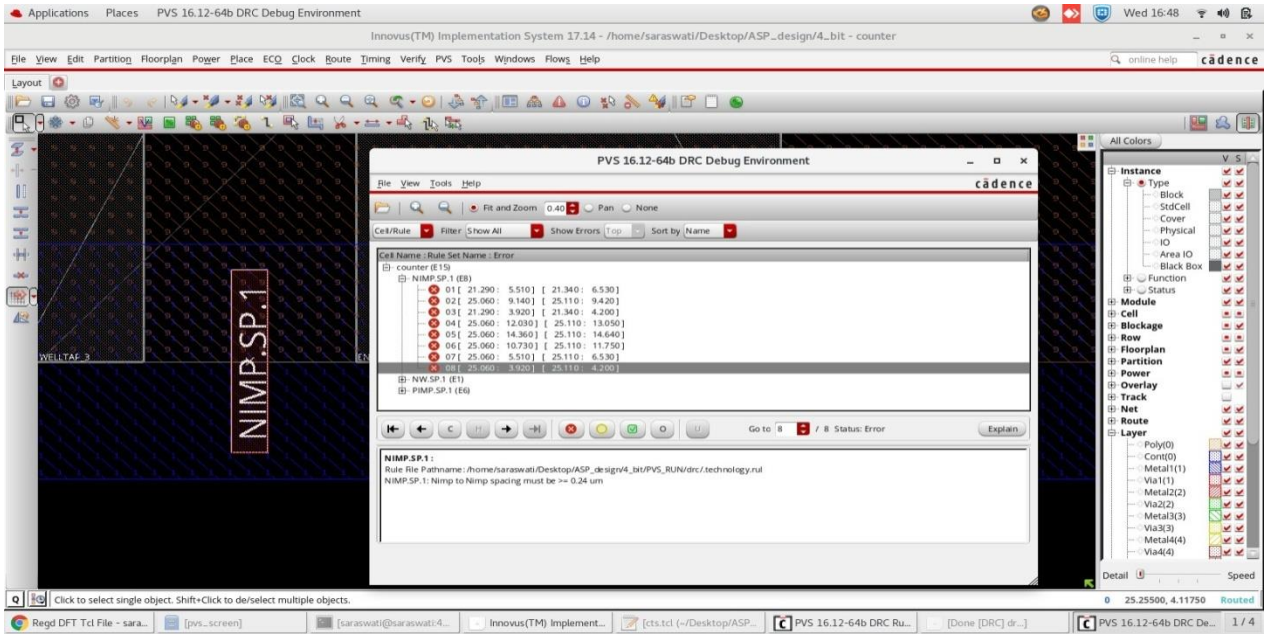



- Hit “Submit” to run the DRC and the following windows appear.

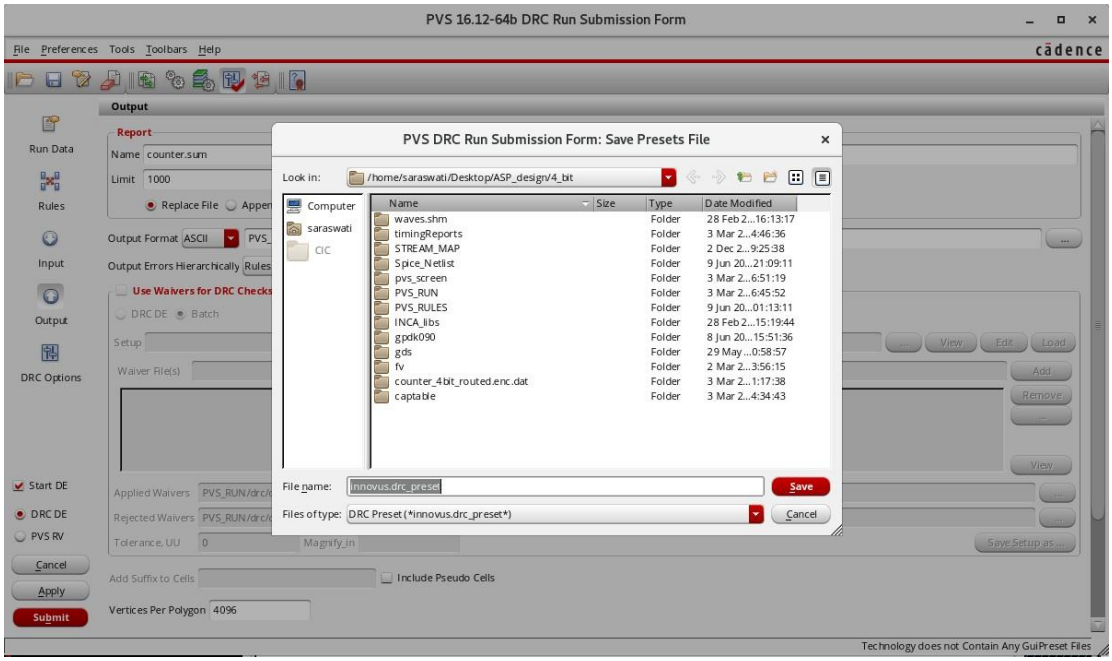




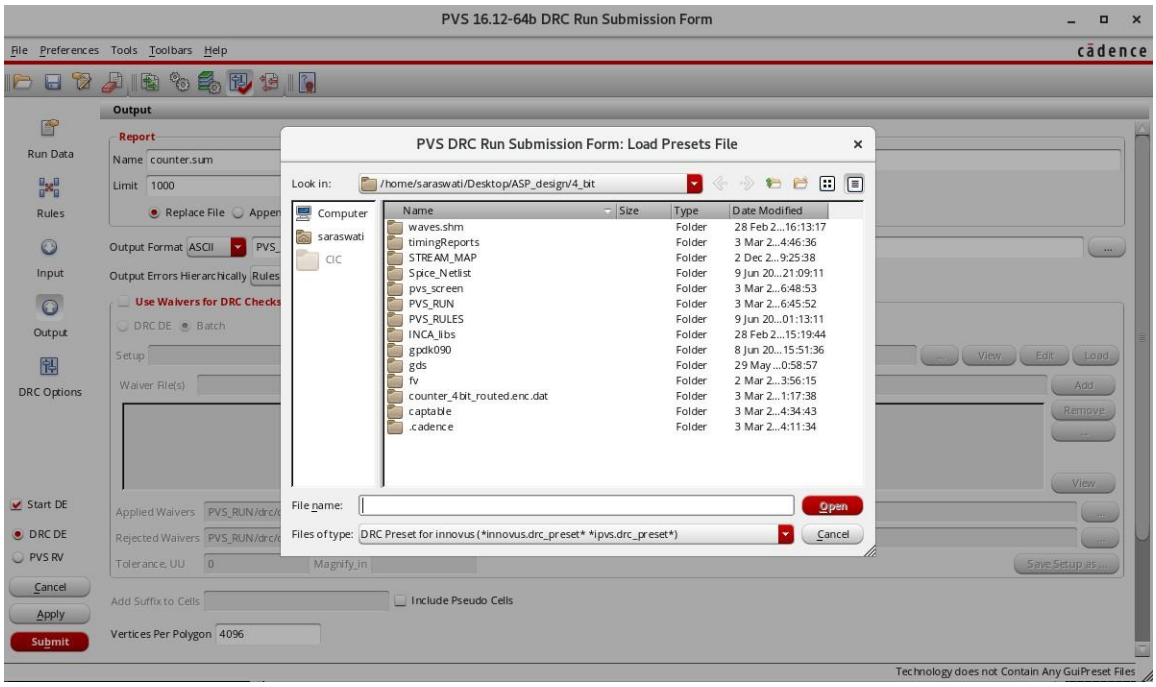

- All the list of DRC Errors can be seen in the above window of which the location of the DRC Violation occuring can be highlighted dealing one to one.



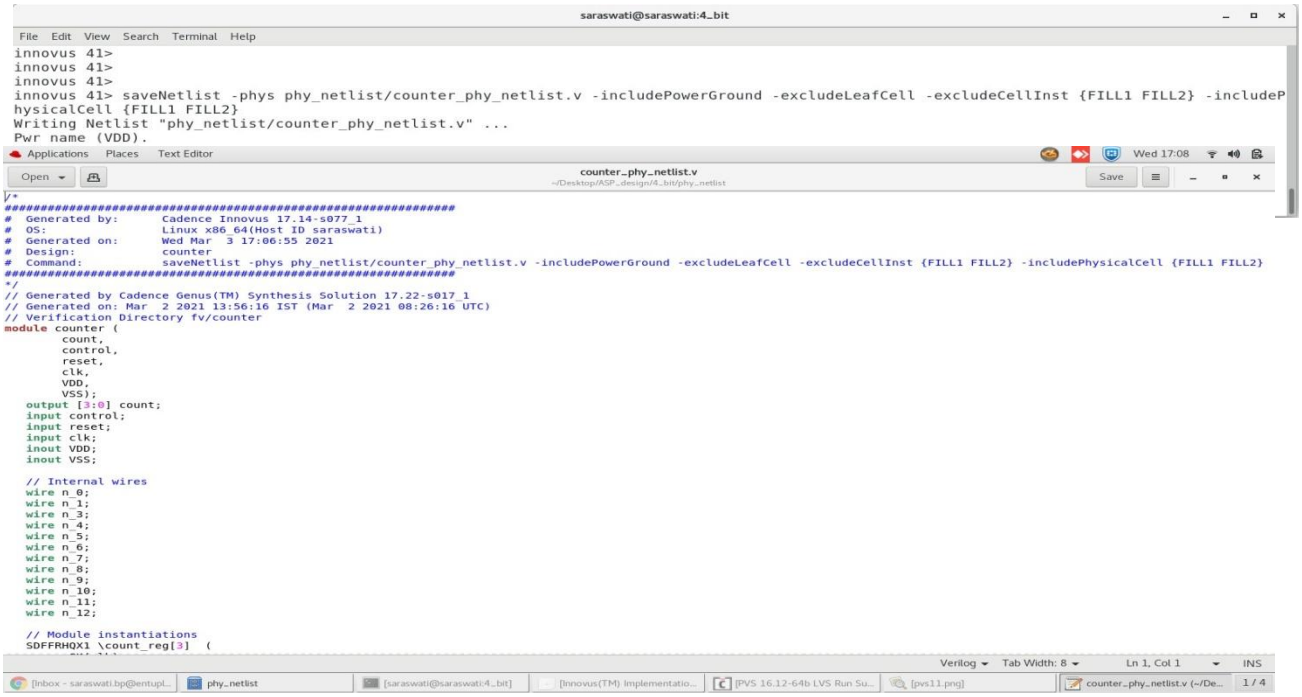
- For example, in the above shown snapshot, the errors associated with N-Implant can be seen. (Select a error occurrence and click on the right arrow below to highlight/zoom in the location.)
- You can save the DRC Run as a “Preset” file to rerun the DRC if required at a later point of time.
- Saving/loading the Preset File is shown below.



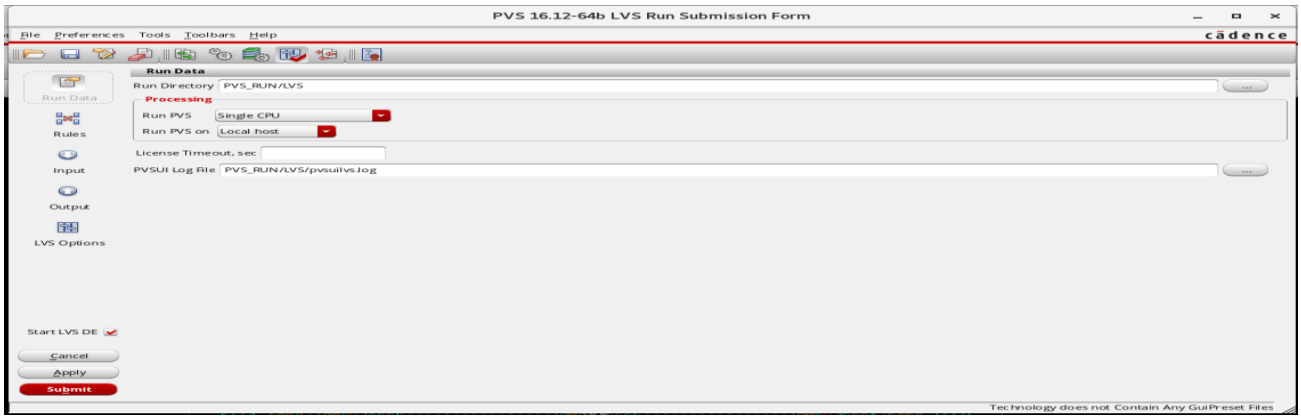

- Loading a Preset file is shown below.



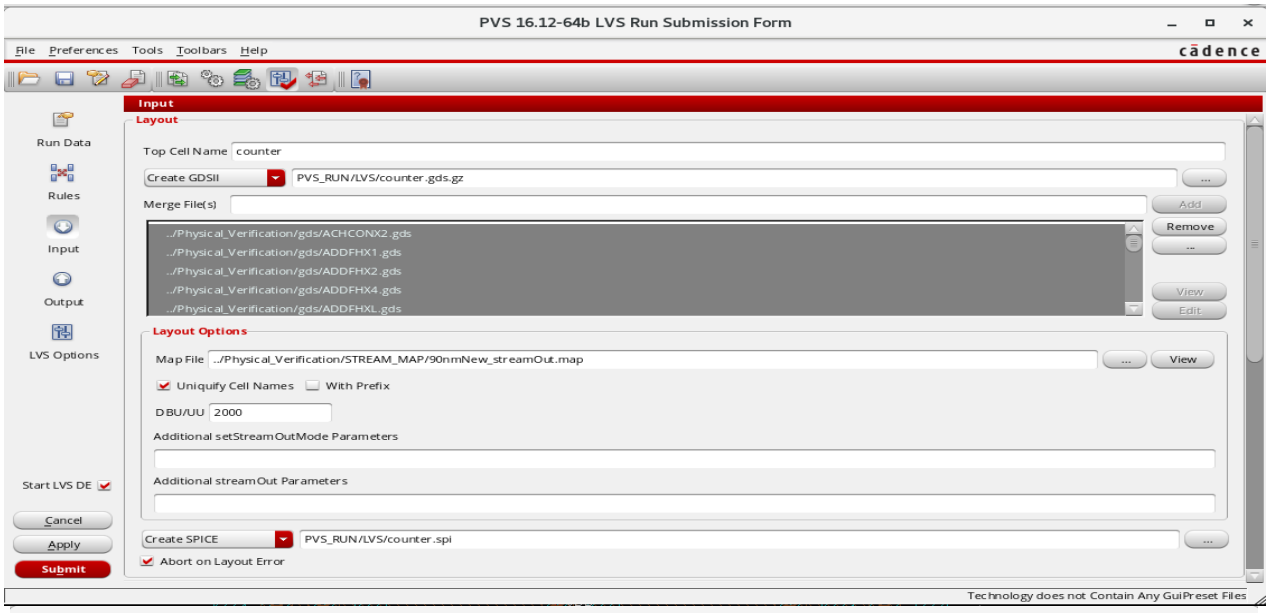

**Note :** A Physical Netlist can be saved after the DRC Run as shown below.

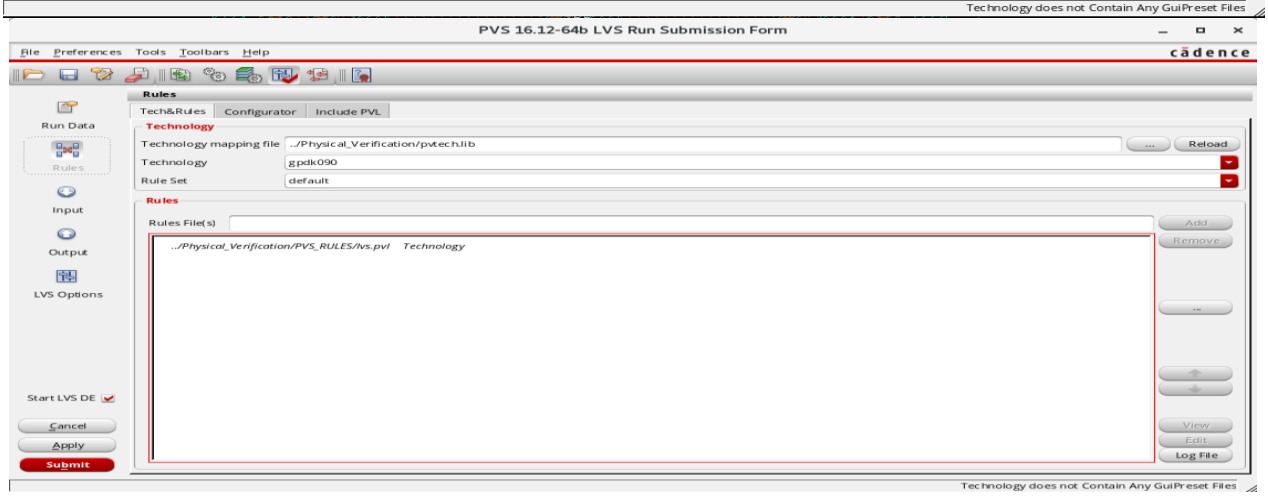


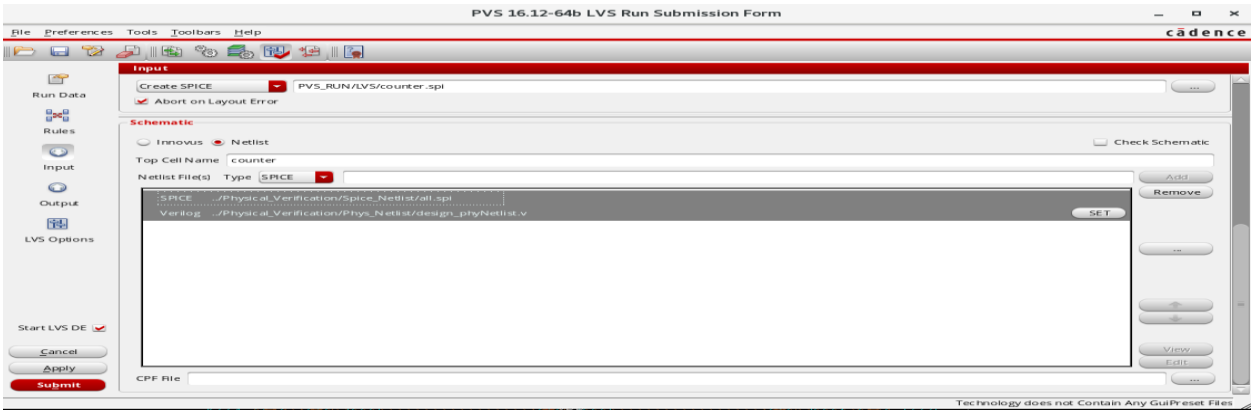
- Inputs Required – LVS :
  - Technology Library
  - Standard Cell GDS Files
  - Spice Netlist of all Standard Cells (Provided by Library Vendor)
- Outputs – LVS :
  - LVS Match/Mismatch Report
- From the Innovus GUI, Select PVS → Run LVS to open the LVS run submission form.

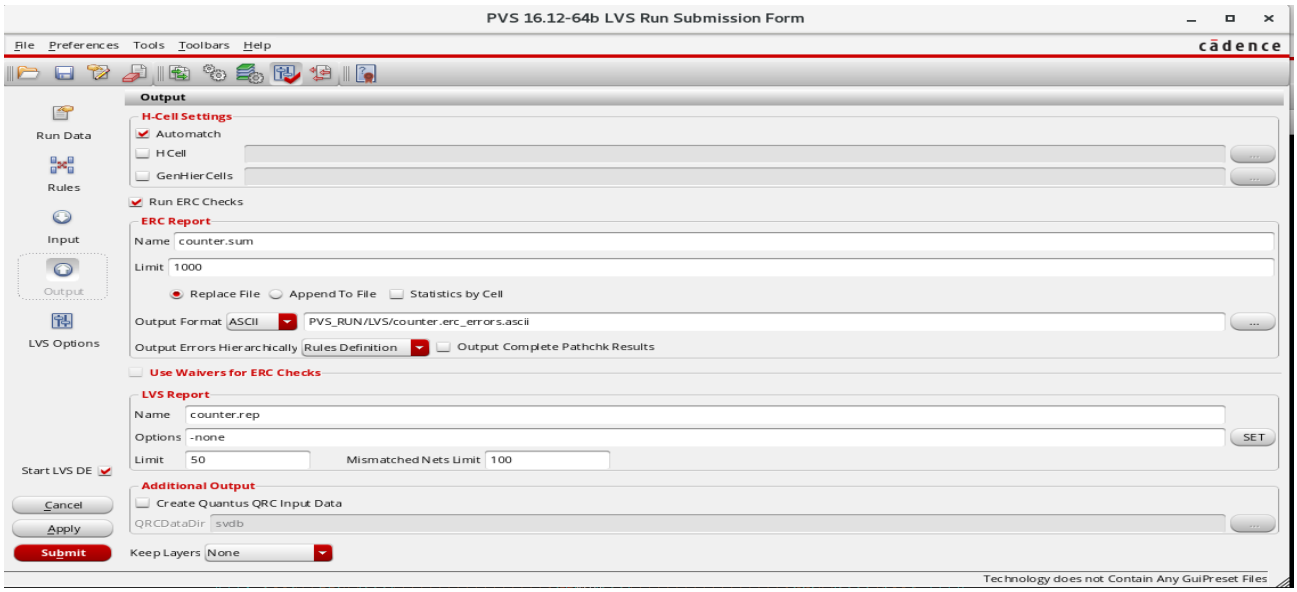



Provide the Run directory and log file name (Along with path – Optional)
Load the Tech Lib, GDS Files and Spice Netlist of all Standard Cells under the corresponding technology node.

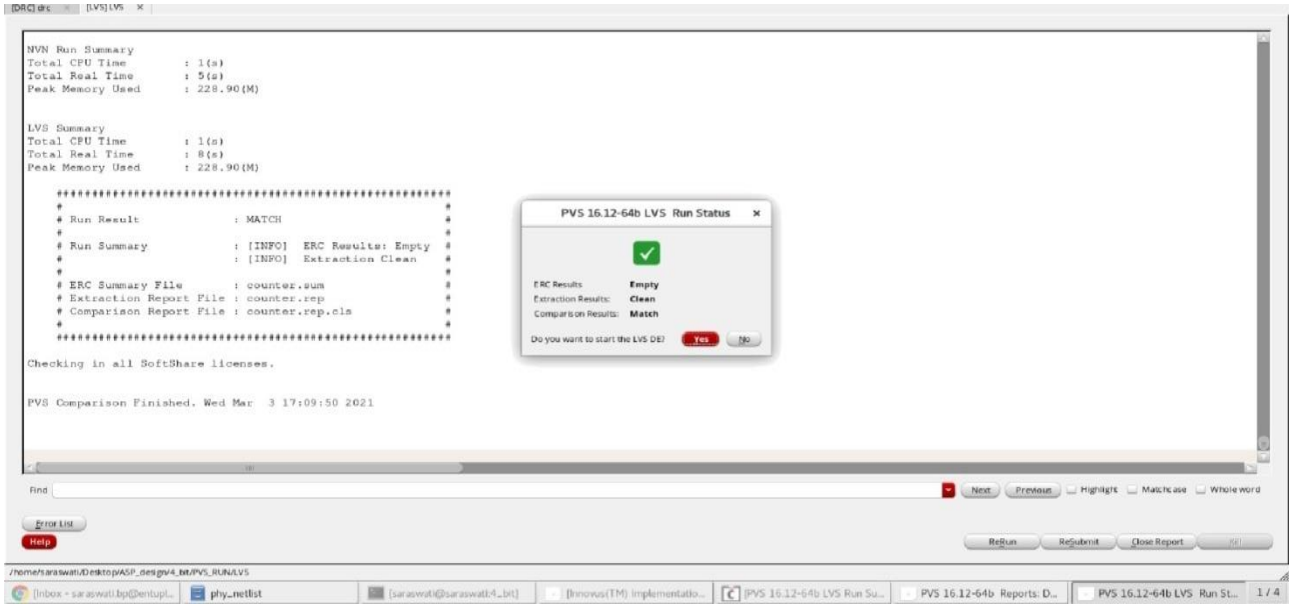

PVS 16.12-64b LVS Run Submission Form
File Preferences Tools Toolbars Help
cadence
Input
Layout
Run Data
Rules
Input
Output
LVS Options
Top Cell Name counter
Create GDSII
PVS_RUN/LVS/counter.gds.gz
Merge File(s)
Add
Remove
../Physical_Verification/gds/ACHCONX2.gds
../Physical_Verification/gds/ADDFHX1.gds
../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

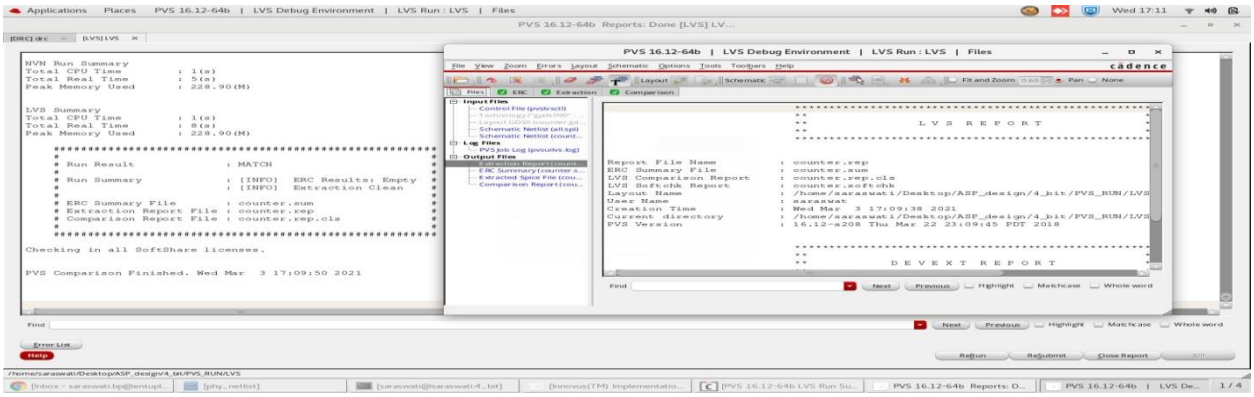

PVS 16.12-64b LVS Run Submission Form
File Preferences Tools Toolbars Help
cadence
Rules
Tech&Rules
Configurator
Include PVL
Run Data
Rules
Input
Output
LVS Options
Technology
Technology mapping file ../Physical_Verification/pvtech.lib
Reload
Technology
gpdk090
Rule Set
default
Rules
Rules File(s)
Add
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

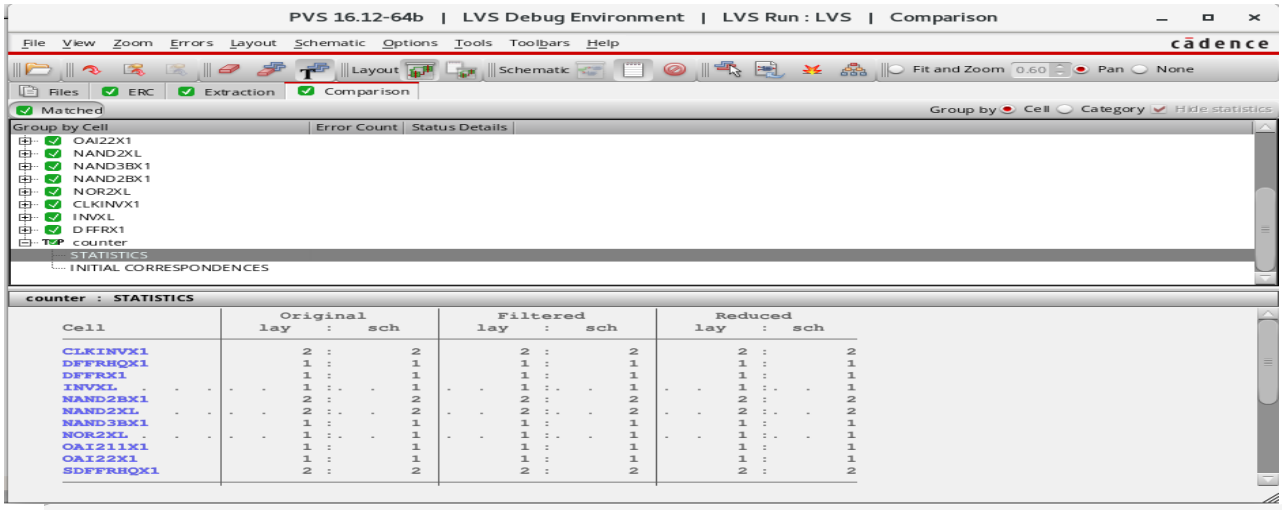

PVS 16.12-64b LVS Run Submission Form
File Preferences Tools Toolbars Help
cadence
Input
Run Data
Rules
Input
Output
LVS Options
Create SPICE
PVS_RUN/LVS/counter.spi
Abort on Layout Error
Schematic
Innovus
Netlist
Check Schematic
Top Cell Name counter
Netlist File(s)
Type SPICE
Add
Remove
SPICE ../Physical_Verification/Spice_Netlist/all.spi
Verilog ../Physical_Verification/Phys_Netlist/design_phyNetlist.v
SET
View
Edit
Start LVS DE
Cancel
Apply
Submit
CPF File
Technology does not Contain Any GuiPreset Files

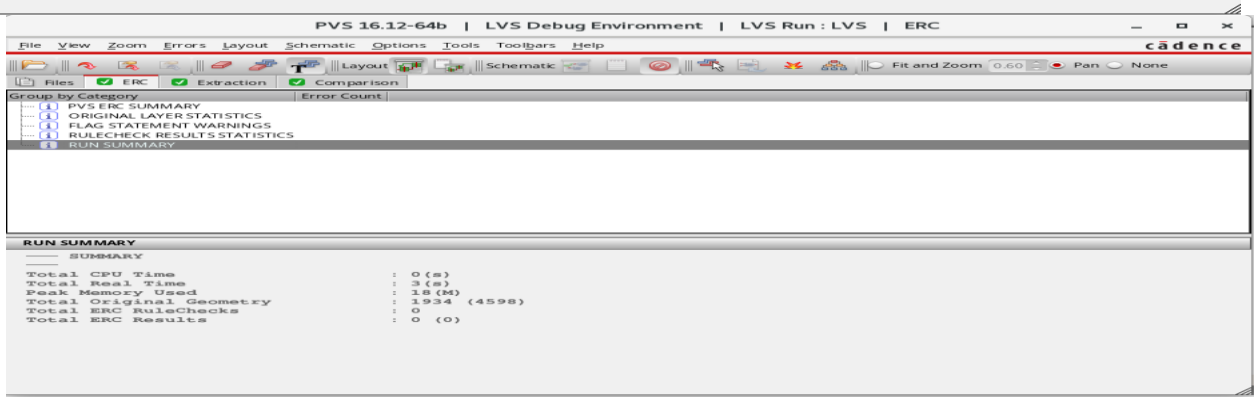



On successful completion of LVS Run, the following windows appear.








- You can create a GDS file along with Stream out file either using the GUI as File → Save → GDS/Oasis or use the following command.
- **Cmd :** streamOut <GDSFileName>.gds -streamOut <streamOut>.map

## Viva Questions:

1. List the three popular Hardware languages.
2. Which are the different levels of abstraction that can be specified using VERILOG?
3. Which are the different modes of port declaration?
4. Which are the valid characters for identifier declaration?
5. Which are the different classes of operators?
6. Where do you write the sequential statement?
7. In which model always statement appears?
8. Device configuration for CPLD and FPGA Used in your Lab.
9. Expand CPLD and FPGA.
10. Differentiate sequential and concurrent statement.
11. What is synthesis
12. What is simulation
13. What is the difference between synchronous and asynchronous reset.
14. What is the basic element of memory?
15. What do you mean by latch?
16. What is the difference between synchronous and asynchronous counter.
17. Expand ASIC.
18. What are the differences between VHDL and verilog.
19. What information is present in .Bit or .Jed File
20. Which file used to configure the FPGA
21. What are four generations of Integration Circuits?
22. Give the advantages of IC?
23. Give the variety of Integrated Circuits?
24. Give the basic process for IC fabrication?
25. What are the various Silicon wafer Preparation?
26. What is the transistors CMOS technology provides?
27. What are the different layers in MOS transistors?
28. What is Enhancement mode transistor?
29. What is Depletion mode Device?
30. When the channel is said to be pinched –off?
31. Give the different types of CMOS process?
32. What are the steps involved in twin-tub process?
33. Define Short Channel devices?
34. Define Threshold voltage in CMOS?
35. What is Body effect?
36. What are the advantages and disadvantages of λ based design rules.
37. What is Switch-level modeling?
38. What are the differences between 'C' language and 'Verilog'?
39. What is the difference between wire and reg data types?
40. What are the applications of Adders and Subtractions?
41. What are the differences between gate level and data flow models, switch level?