P.E.S.COLLEGE OF ENGINEERING

Mandya– 571401, Karnataka

(An Autonomous Institution under VTU, Belgaum)

2018-2019

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

LABORATORY MANUAL

ANALOG AND DIGITAL VLSI DESIGN LABORATORY P18ECL67

VI Semester, BE (ECE)

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

Vision

The department of E & C would endeavour to create a pool of Engineers who would be extremely competent technically, ethically strong also fulfil their obligation in terms of social responsibility.

Mission

M1: Adopt the best pedagogical methods and provide the best facility, infrastructure and an ambience conducive to imbibe technical knowledge and practicing ethics.

M2: Group and individual exercises to inculcate habit of analytical and strategic thinking to help the students to develop creative thinking and instil team skills

M3: MoUs and Sponsored projects with industry and R & D organizations for collaborative learning

M4: Enabling and encouraging students for continuing education and moulding them for life-long learning process

Program Educational Objectives:

PEO1: Graduates to exhibit knowledge in mathematics, engineering fundamentals applied to Electronics and Communication Engineering for professional achievement in industry, research and academia

PEO2: Graduates to identify, analyse and apply engineering concepts for design of Electronics and Communication Engineering systems and demonstrate multidisciplinary expertise to handle societal needs and meet contemporary requirements

PEO3: Graduates to perform with leadership qualities, team spirit, management skills, attitude and ethics need for successful career, sustained learning and entrepreneurship.

Program Specific Outcomes (PSOs):

Program Specific Outcomes of bachelor degree (B.E, E&C) program are defined as follows which are in line with the Program specific criteria (PSC) as defined by IEEE.

After the graduation, the student will have:

- An ability to understand the basic concepts in Electronics & Communication Engineering and to apply them in the design and implementation of Electronics and communication systems.
- An ability to solve complex problems in Electronics and Communication Engineering, using latest hardware and software tools, along with analytical skills to arrive at appropriate solutions.

Laboratory			
Course Title : **Analog and Digital VLSI Design Laboratory**			
Course Code: **P18ECL67**	Semester : **VI**	L-T-**P**-**H** : 0-0-**3**-**3**	Credits: **1.5**
Contact Period: Lab: **36 Hrs.,** Exam: **3 Hrs.**		Weightage: CIE: **50%** SEE: **50%**	

A. Course Learning Objectives (CLOs)

This course aims to:

1. Explore the CAD tool and understand the flow of the Full Custom IC design cycle.
2. Learn DRC, LVS and Parasitic Extraction of the various designs.
3. Design and simulate the various basic CMOS analog circuits and use them in higher circuits like operational amplifiers using design abstraction concepts.
4. Design and simulate the various basic CMOS digital circuits and use them in higher circuits like adders and shift registers using design abstraction concepts.
5. Understand simulation and synthesis of digital design.
6. Analyze the ASIC Design flow.
7. RTL Design, simulate and verify digital circuits.

B. Course Content

Part A: Analog VLSI Design

Analog Design Flow:

The design flow must consist of the following:

1. Draw the schematic and verify the following:
   - DC Analysis
   - Transient Analysis
2. Draw the Layout and verify the DRC, ERC, and LVS.
3. Check for LVS.
4. Extract RC and Back annotate the same and verify the Design
5. Design an Inverter gate with given specification.
6. Design an NAND and NOR gate with given specification.
7. Design the following circuits, in different styles, for the given specification
   - Common source amplifier
   - Common Drain amplifier.
8. Design a Single Stage Differential Amplifier for given specifications.
9. Design an OPAMP for given specifications using Differential Amplifier.
10. Analysis, Design and Characterization of SRAM memory cell/block.

Part B. Digital VLSI Design

ASIC-Digital Design / FPGA Digital Design

1. Develop Verilog Code for the n inverter, Buffer and their Test Bench for verification.
2. Develop Verilog Code for the Transmission gate and their Test Bench for verification.
3. Design and Develop Verilog code for 4/8-bit Carry Ripple Adder.
4. Design and Develop Verilog code for 4/8-bit Carry Look Ahead adder.
5. Develop Verilog Code for Radix-4 Booth Multiplication.
6. Develop Verilog code for 4/8-bit Universal Shift Register.

Open Ended Experiments:

1. Design and simulate Gilbert cell for Analog multiplication

OR

2. Demonstration of place and Route steps with DMA MAC example using INNOVUS.

C. Course Outcomes

CO #	Course Outcome	Program Outcome Addressed (PO #) with BTL
CO1	**Apply** the knowledge of the digital system to design of the schematic and layout in cadence tools.	**PO1 (L1)**
CO2	**Interpret** the outcome of DC Analysis, AC Analysis and Transient Analysis in analog circuits.	**PO4, PO9 (L4)**
CO3	**Design** and **simulate** basic CMOS circuits like inverter, common source amplifier and differential amplifiers.	**PO3, PO5, PO8, (L5)**
CO4	**Analysis** of the **design** for power, timing and area.	**PO2, PO5 (L4)**
CO5	**Develop** 4/8-bit Carry Ripple Adder, Carry Look Ahead adder and Booth Multiplication using Verilog code.	**PO3, PO5, PO7, (L5)**

D. Course Articulation Matrix (CAM)

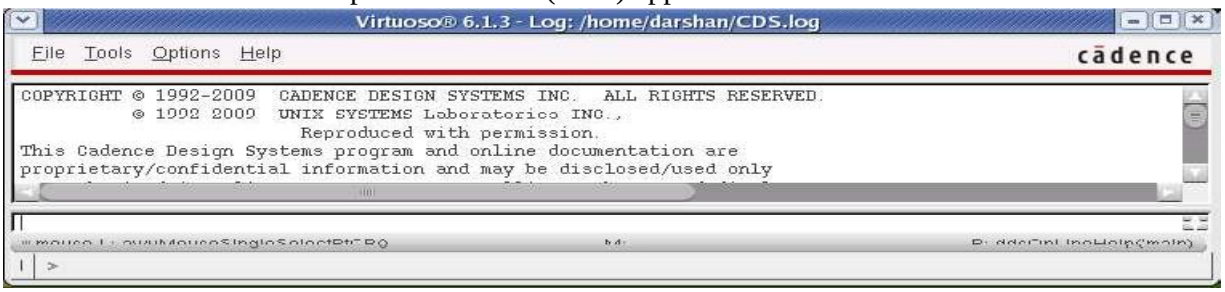
CO	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	PS O1	PS O2
#1	**3**												**3**	
#2				**2**					**3**					
#3			**2**		**3**			**2**						
#4		**3**			**3**									**3**
#5			**2**		**3**		**2**							

A. Analog VLSI Design

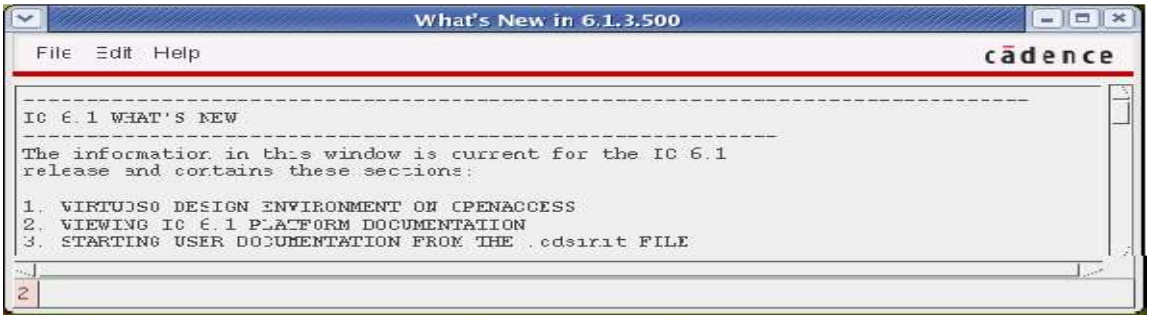
Getting Started with Cadence tools suite:

1. Log in to your workstation using the username and password.
   Username: su
   Password: redhat
2. In a terminal window, type csh at the command prompt to invoke the C shell.
   >csh
3. Create the path from server.
   Source /cadence/cshrc
   Then it will display welcome to cadence tool.
4. Change to the course directory by entering the command:
   >cd
5. In the same terminal window, enter:
   >virtuoso &

The virtuoso or Command Interpreter Window (CIW) appears at the bottom of the screen.



6. If the "What's New ..." window appears, close it with the File-Close command.



7. Keep the CIW window open.

Experiment 1: INVERTER.

Aim: Design an Inverter with given specifications, completing the design flow mentioned below:

a) Draw the schematic and verify the following
   - DC Analysis
   - Transient Analysis

b) Draw the Layout and verify the DRC, ERC

c) Check for LVS

d) Extract RC and back annotate the same and verify the Design.

Optimize the design for Time, Power and Area for the given constraint

Objective: To create a library and build a schematic of an Inverter

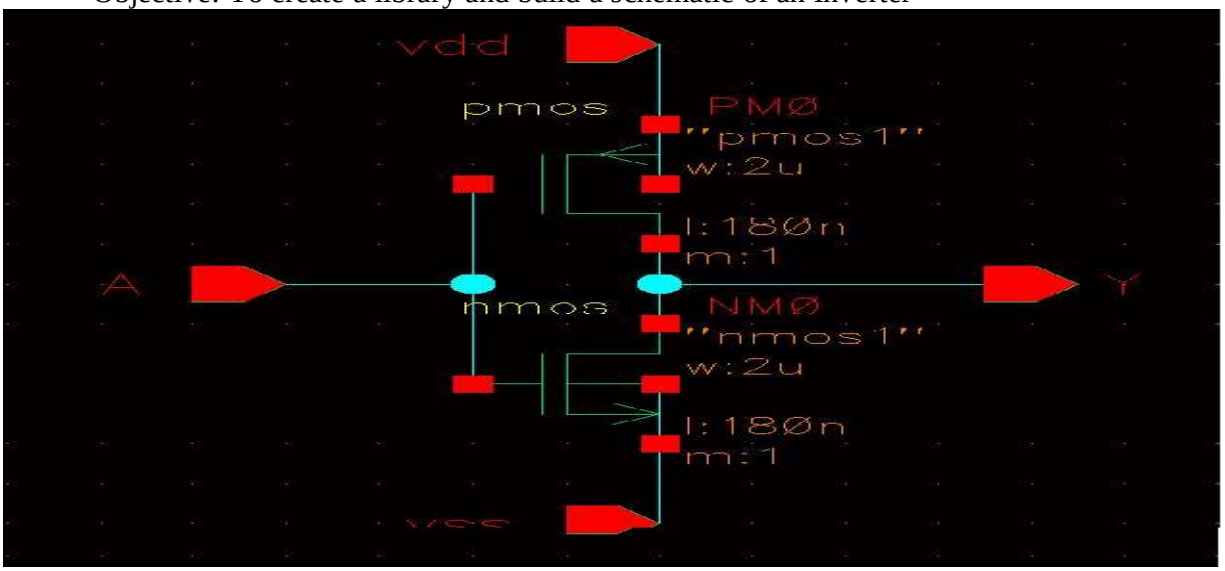


Figure 1.1: Schematic Entry

Create Design Library:

1. In the Library Manager in CIW window, Click on the file tab in CIW window, In the dropdown list click on New then click on Library. The new library window appears on the screen.
2. In the "New Library" window, type "myDesignLib" in the Name field.(any name can be given instead of mydesignlib) and click OK

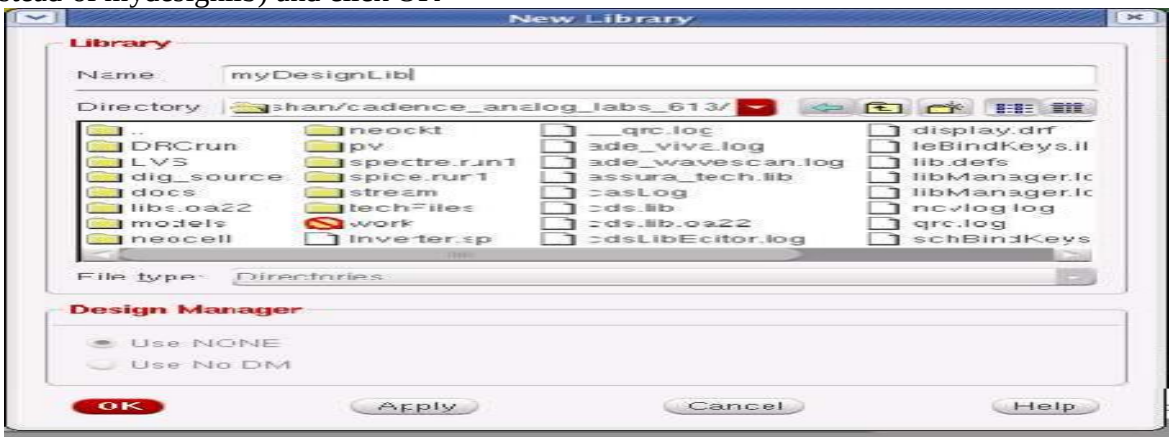


Figure1.2: New library dialogue box

3. In the Directory field, verify that the path to the library is set to cadence_analog_labs_613 and click OK.

Attaching the Technology File:

4. In "Technology File for New library" window, select option Attach to an existing technology file and click OK.

5. In the "Attach Design Library to Technology File" window select gpdk180 and click OK.

Figure 1.3: Technology library

6. After creating a new library you can verify it from the library manager.
7. If you right click on the "myDesignLib" and select properties, you will find that gpdk180 library is attached as techlib to "myDesignLib".

Figure 1.4: Library manager

Creating a Schematic Cellview:

In this section we will learn how to open new schematic window in the new "myDesignLib" library and build the inverter schematic as shown in the figure 1. 1.

1. In the CIW or Library manager, Go to File – New – Cellview.
2. Set up the new file form as follows:

Figure 1.5: New file

Do not edit the Library path file and the one above might be different from the path shown in your form.
3. Click OK. A blank schematic window for the schematic design appears.

Adding Components to the schematic window:

Figure 1.6: shortcut to add components.

1. In the schematic window, click the Instance fixed menu icon to display the Add Instance form. 

Tip: You can also execute Create — Instance or press I in the keyboard.

2. Click on the Browse button. This opens up a Library browser from which you can select components and the symbol view. You will update the Library Name, Cell Name, and the property values given in the table on the next page as you place each component.

3. After you complete the Add Instance form, move your cursor to the schematic window and click left to place a component. The table1.1 shows the components for building the Inverter schematic as an example.

Library name	Cell Name	Properties/Comments
gpdk180	pmos	For M0: Model name = pmos1, W= wp, L=180n
gpdk180	nmos	For M1: Model name = nmos1, W= 2u, L=180n

Table1.1: Components for building the Inverter schematic

If you place a component with the wrong parameter values, use the Edit— Properties— Objects command to change the parameters. Use the Edit— Move command if you place components in the Wrong location.

You can rotate components at the time you place them, or use the Edit— Rotate command after they are placed.

4. After entering components, click Cancel in the Add Instance form or press Esc with your cursor in the schematic window.

Adding pins to Schematic:

1. Click the Pin icon in the schematic window.

You can also execute Create — Pin or press p in the keyboard.  The Add pin form appears.

2. Type the following in the Add pin window in the exact order leaving space between the pin names.

Pin Names	Direction
vin	Input
vout	Output

Make sure that the direction field is set to input/output/inputOutput when placing the input/output/inout pins respectively and the Usage field is set to schematic.

3. Select Cancel from the Add – pin window after placing the pins. In the schematic window, execute Window— Fit or press the f in the keyboard. 

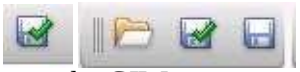
Adding Wires to a Schematic:

Add wires to connect components and pins in the schematic design.

1. Click the Wire (narrow) icon in the schematic window. You can also press the w key, or execute Create — Wire (narrow). 
2. In the schematic window, click on a pin of one of your components as the first point for your wiring. A diamond shape appears over the starting point of this wire.
3. Follow the prompts at the bottom of the design window and click left on the destination point for your wire. A wire is routed between the source and destination points.
4. Complete the wiring as shown in figure1.1 and when done wiring press ESC key in the schematic window to cancel wiring.

Saving the Design:

1. Click the Check and Save icon in the schematic editor window.



2. Observe the CIW output area for any errors.

Symbol Creation:

1. In the Inverter schematic window Go to Create — Cell view— From Cell view.
   The Cell view from Cell view window appears on the screen as shown in figure 1.7.
2. Verify that **From View Name** field is set to schematic, and To **View Name** field is set to symbol, with the **Tool/Data Type** set as Schematic Symbol.

Figure1.7: Cellview From Cellview Window

3. Click OK in the Cellview from Cellview window.
   The Symbol Generation Option appears as shown in figure1.8.
4. Modify the Pin Specifications as follows in figure 1.8.

Figure 1.8: Symbol Generation Options

5. Click OK in the Symbol Generation Options window.
6. A new window displays an automatically created Inverter symbol as shown in figure 1.9.

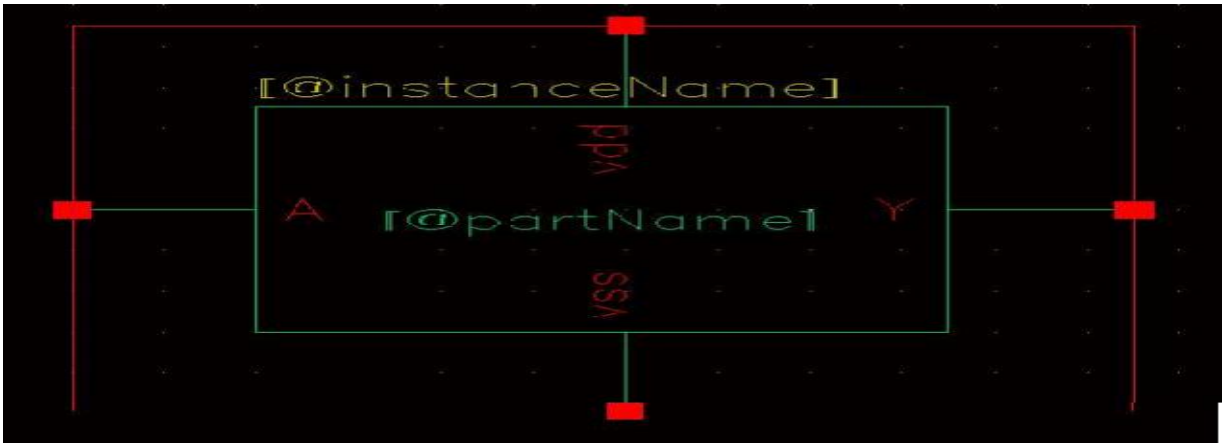


Figure1.9: Inverter symbol

Editing a Symbol:

In this section we will modify the inverter symbol to look like a Inverter gate symbol.

1. Move the cursor over the automatically generated symbol, until the green rectangle is highlighted, click left to select it.
2. Click Delete icon in the symbol window, similarly select the red rectangle and delete that.
3. Execute Create – Shape – polygon, and draw a shape similar to triangle.
4. After creating the triangle press ESC key.
5. Execute Create – Shape – Circle to make a circle at the end of triangle.
6. You can move the pin names according to the location.
7. Execute Create — Selection Box. In the Add Selection Box form, click Automatic.

A new red selection box is automatically added.

8. After creating symbol, click on the save icon in the symbol editor window to save the symbol.

In the symbol editor, execute File — Close to close the symbol view window.

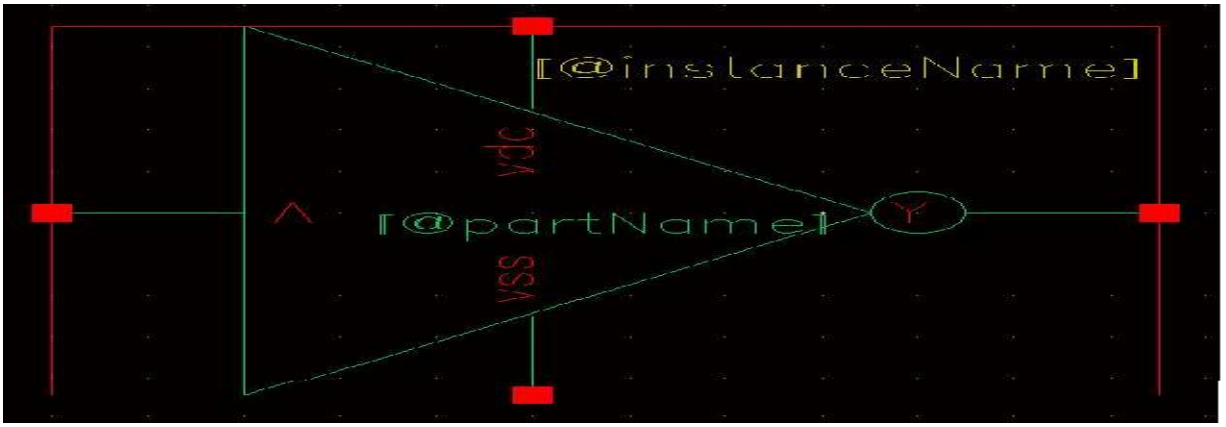


Figure 1.10: Edited Inverter Symbol

Creating the Inverter_Test Cellview:

You will create the Inverter_Test (Cell name) cell view that will contain an instance of the Inverter cellview. In the next section, you will run simulation on this design

1. In the CIW or Library Manager, Go To File— New— Cellview.
2. Set up the New File window as shown in figure 1.11.

Figure 1.11: New file window for Test bench

3. Click OK. A blank schematic window for the Inverter_Test design window appears on the screen .

Building the Inverter_Test Circuit:

1. Using the component list and Properties/Comments in the table1.2, build the Inverter_Test schematic.

Library name	Cellview name	Properties/Comments
myDesignLib	Inverter	Symbol
analogLib	vpulse	v1=0, v2=1.8,td=0 tr=tf=1ns, ton=10n, T=20n
analogLib	vdc, gnd	vdc=1.8

Table 1.2: Inverter Test bench Components

Note: Remember to set the values for VDD and VSS. Otherwise, your circuit will have no power.

2. Add the above components using Create — Instance or by pressing I.

3. Click the Wire (narrow) icon and wire your schematic.

Tip: You can also press the w key, or execute Create— Wire (narrow).

4. Click Create — Wire Name or press L to name the input (Vin) and output (Vout) wires as in the below schematic.

5. Click on the Check and Save icon to save the design.

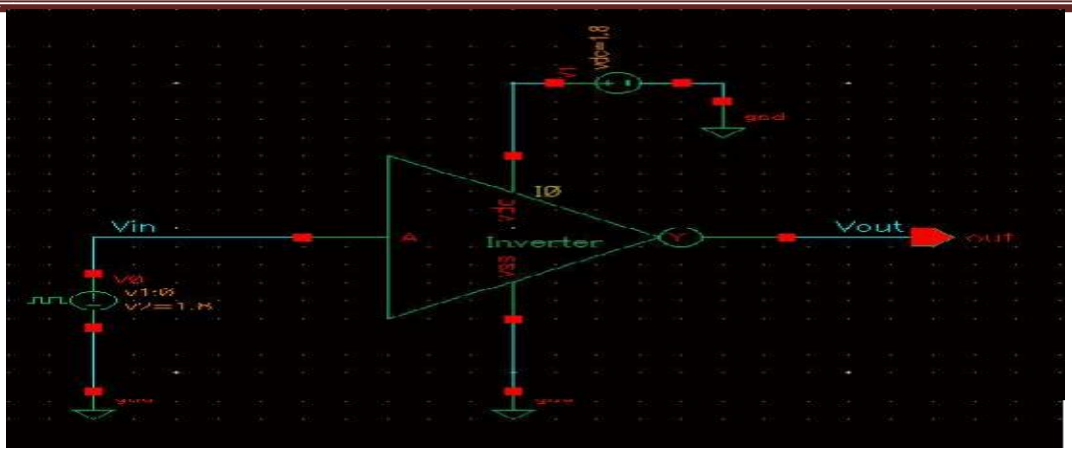


Figure 1.12: Inverter_Test schematic

Analog Simulation with Spectre:

1. In the Inverter_Testschematic window, Go To Launch – ADE L
The Virtuoso Analog Design Environment (ADE) simulation window appears.

Setting the Model Libraries:

The Model Library file contains the model files that describe the nmos and pmos devices during simulation.

1. In the simulation window (ADE), Go To Setup - Model Libraries.

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

Remember to select the section type as stat in front of the gpdk.scs file.

Your Model Library Setup window shown in figure 1.13.

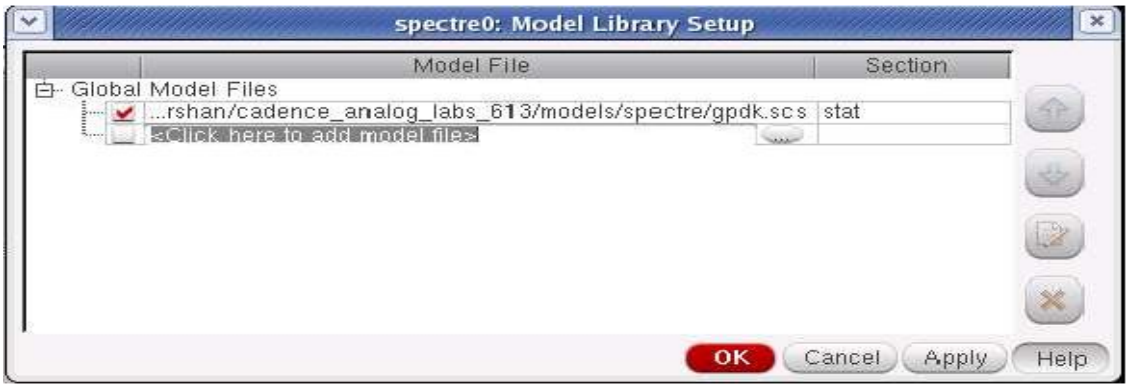


Figure 1.13: Model Libraries setup window

Choosing Analyses:

This section demonstrates how to view and select the different types of analyses to complete the circuit when running the simulation.

1. In the Simulation window (ADE), Go to Analyses--- Choose
The Choosing Analysis window as shown in Figure 1.14.

2. **Setup for transient analysis**

a. In the Analysis section select tran

b. Set the stop time as 200n

c. Click at the moderate or enabled button at the bottom, and then click Apply.

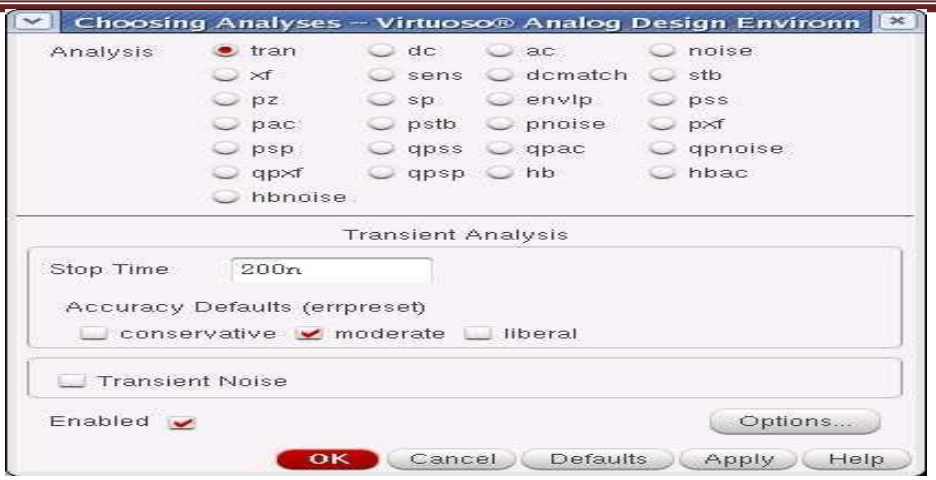


Figure 1.14: Choosing Tran Analyses Window

4. **Set up for DC Analyses:**

a. In the Analyses section, select dc.

b. In the DC Analyses section, turn on Save DC Operating Point.

c. Turn on the Component Parameter.

d. Double click the Select Component, Which takes you to the schematic window.

e. Select input signal vpulse source in the test schematic window.

f. Select —DC Voltage in the Select Component Parameter window and click OK.

g. In the analysis window types start and stop voltages as 0 to 1.8 respectively.

h. Select the enable button and then click Apply.

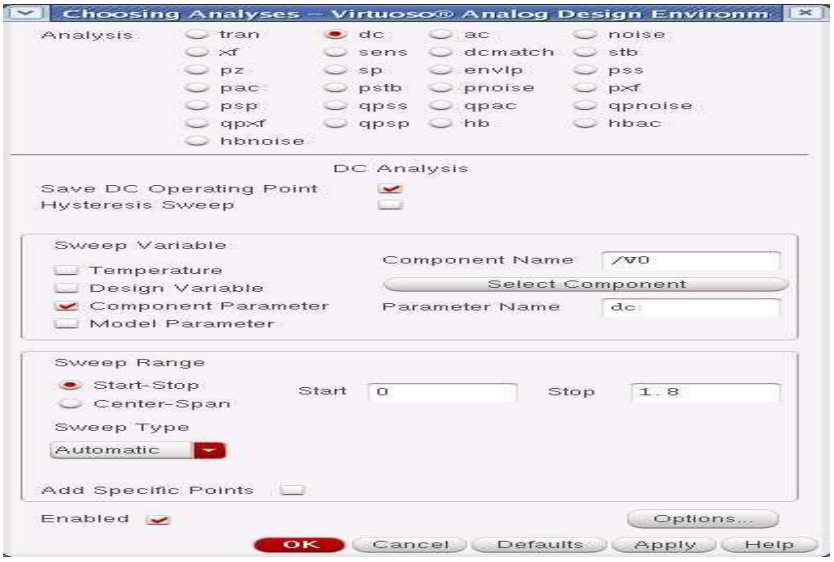


Figure 1.15: Choosing DC analysis window

5. Click OK in the Choosing Analyses Form.

Setting Design Variables:

Set the values of any design variables in the circuit before simulating. Otherwise, the simulation will not run. 

1. In the Simulation window, click the Edit Variables icon. The Editing Design Variables form appears.

2. Click Copy From at the bottom of the form. The design is scanned and all variables found in the design are listed. In a few moments, the wp variable appears in the Table of Design variables section.

3. Set the value of the wp variable: With the wp variable highlighted in the Table of Design Variables, click on the variable name wp and enter the following:

Value(Expr)	2u

Click Change and notice the update in the Table of Design Variables.

3. Click OK or Cancel in the Editing Design Variables window.

Selecting Outputs for Plotting:

1. Go to Outputs – To be plotted – Select on Schematic in the simulation window.

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

Finally the virtuoso Analog Environment window is shown in figure 1.16

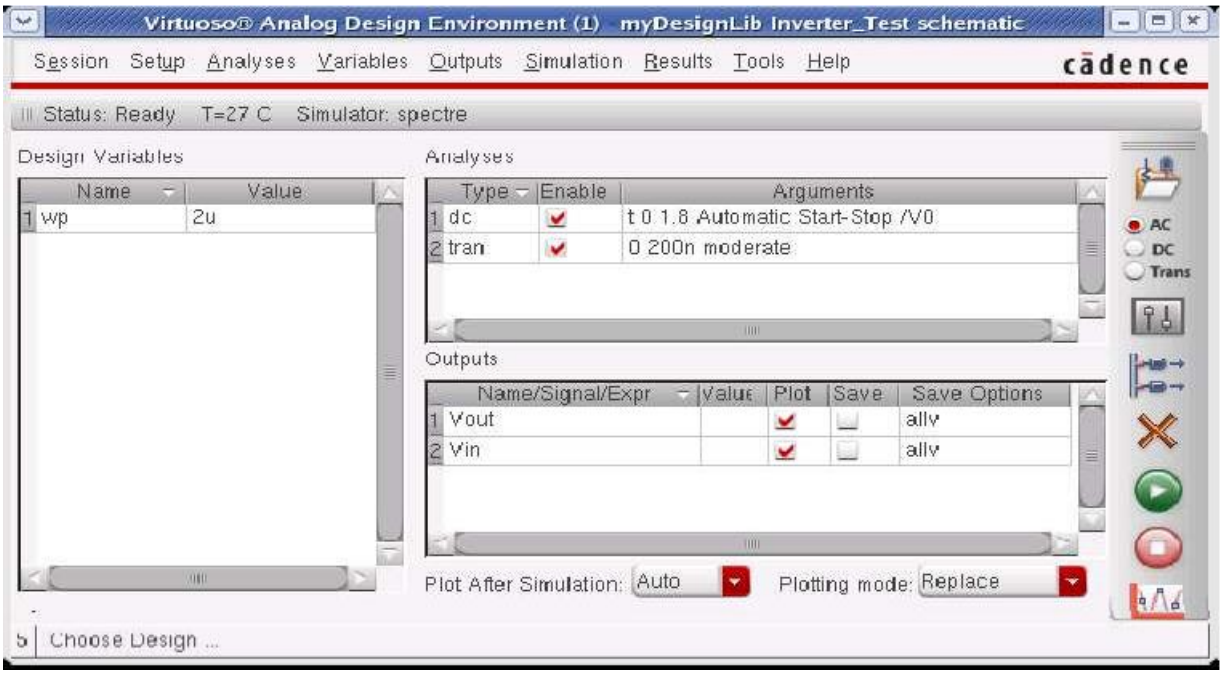


Figure 1.16: Virtuoso Analog Design Environment

Running the Simulation:

1.In the Virtuoso Analog Design Environment window select Simulation – Netlist and Run in the simulation window or the icon,  . This will create the netlist as well as run the simulation.

2. When simulation finishes, the Transient, DC plots automatically will be popped up along with log fille as shown in figure1.17

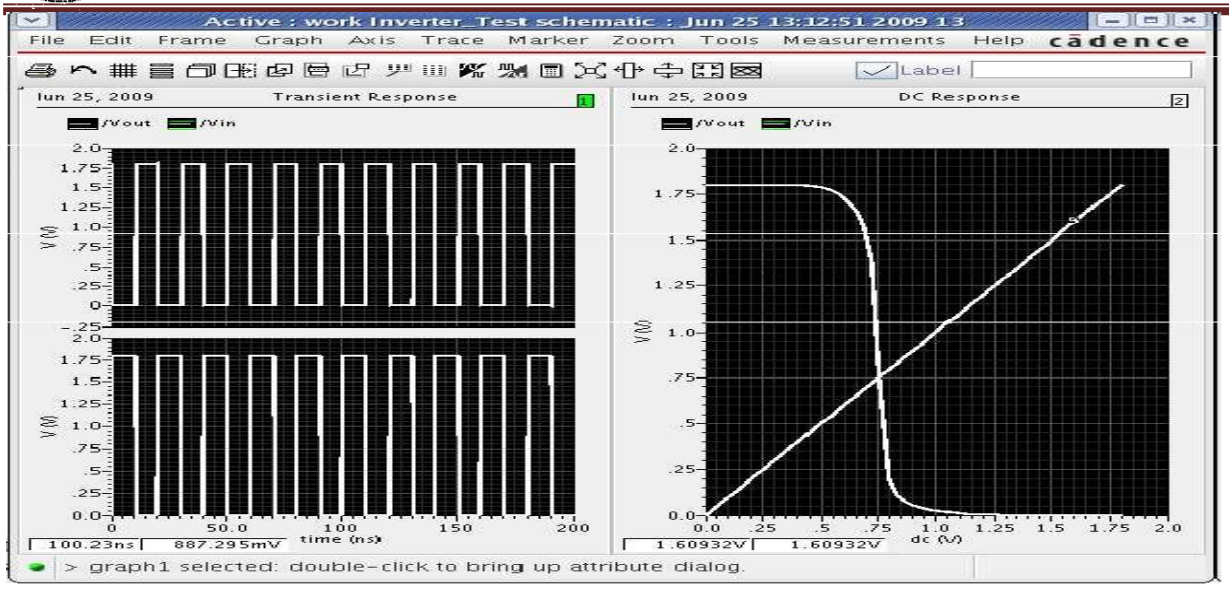


Figure 1.17: Transient and DC Response

Saving the Simulator State:

We can save the simulator state, which stores information such as model library file, outputs, analysis, variable etc. This information restores the simulation environment without having to type in all of setting again.

1. In the Simulation window, execute Session – Save State. The Saving State form appears.
2. Set the Save as field to state1_inv and make sure all options are selected under what to save field.
3. Click OK in the saving state form. The Simulator state is saved.

Loading the Simulator State:

1. From the ADE window execute Session – Load State.
2. In the Loading State window, set the State name to state1_inv as shown in figure 1.18

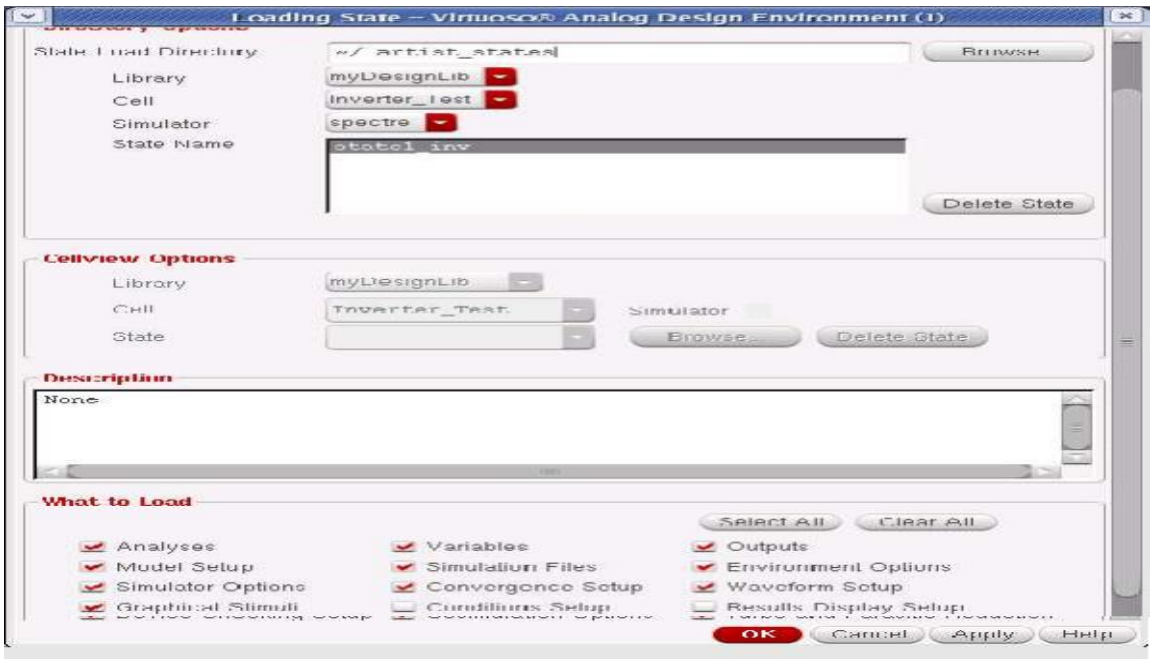


Figure1.18: Loading State window

3. Click OK in the Loading State window.

Starting the Parametric Analysis Tool:

1. In the Simulation window, execute Tools—Parametric Analysis. The Parametric Analysis form appears as shown in figure 1.19.
2. In the Parametric Analysis form, Go to Setup—Pick Name For Variable—Sweep 1.

A selection window appears with a list of all variables in the design that you can sweep. This list includes the variables that appear in the Design Variables section of the Simulation window.

3. In the selection window, double click left on wp.

The Variable Name field for Sweep 1 in the Parametric Analysis form is set to wp.

4. Change the Range Type and Step Control fields in the Parametric Analysis form as shown below:

Range Type : From/To **From** : 1u To10u
Step Control : Auto **Total Steps** : 10

These numbers vary the value of the wp of the pmos between 1um and 10um at ten evenly spaced intervals.

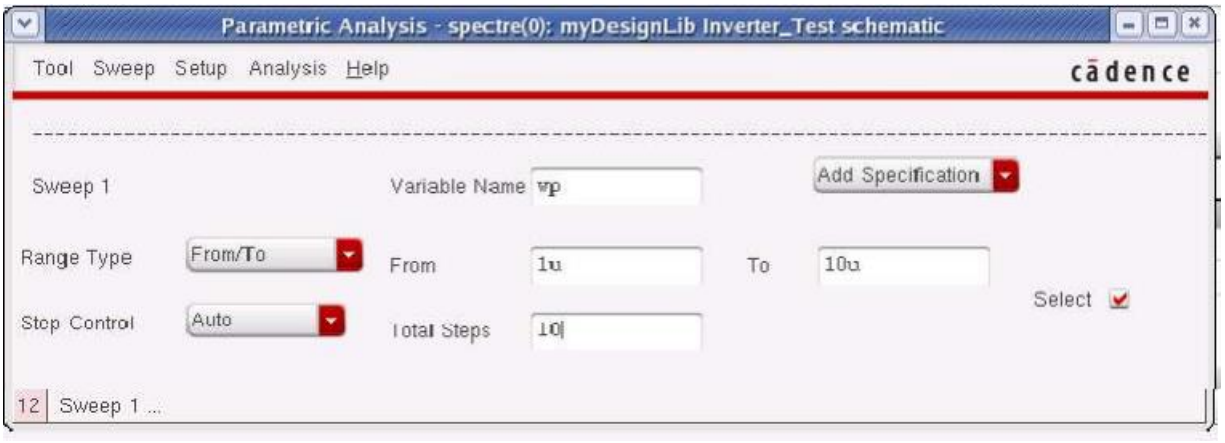


Figure1.20: Parametric Analysis window

5. Execute Analysis

The Parametric Analysis window displays the number of runs remaining in the analysis and the current value of the swept variable(s) . Look in the upper right corner of the window. Once the runs are completed the wave scan window comes up with the plots for different runs as shown in figure 1.21.

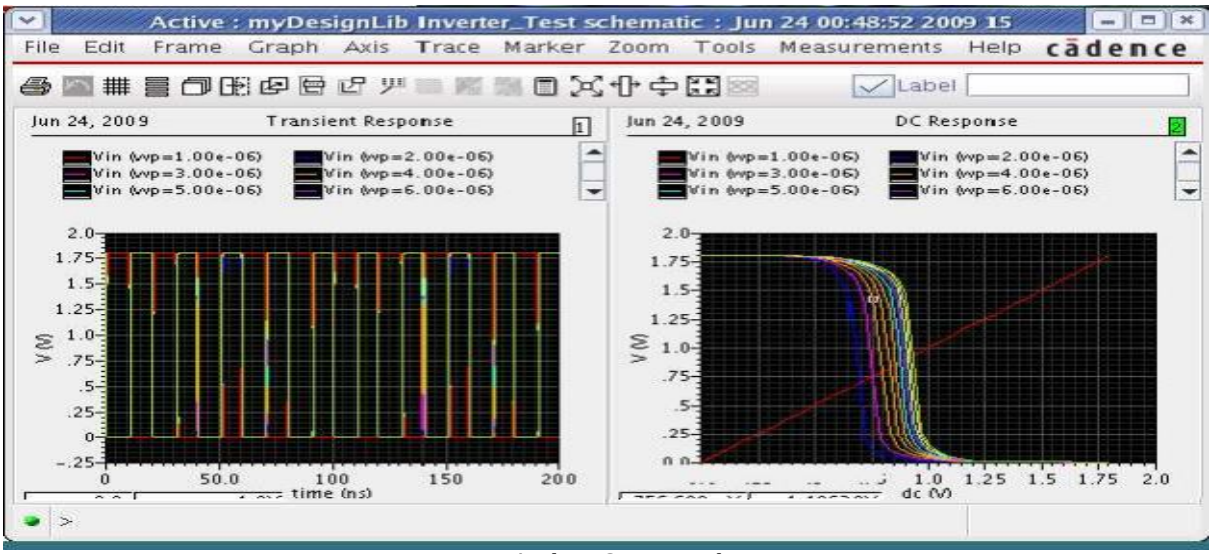


Figure 1.21: Parametric Analysis Output window

Note: Change the wp value of pmos device back to 2u and save the schematic before proceeding to the next section of the lab. To do this use edits property option. 

Creating Layout View of Inverter:

1. From the Inverter schematic window menu Go to Launch – Layout XL. A Startup Option window appears.
2. Select Create New option. This gives a New Cell View window
3. Check the Cellname(Inverter), Viewname(layout).
4. Click OK from the New Cellview window.

LSW and a blank layout window appear along with schematic window.

Adding Components to Layout:

1. Go to Connectivity – Generate – All from Source or click the icon in the layout editor window, Generate Layout window appears. Click OK which imports the schematic components in to the Layout window automatically.
2. Rearrange the components with in PR-Boundary as shown in the figure1.22
3. To rotate a component, Select the component or press R in the keyboard and select Edit –Properties. Now select the degree of rotation from the property edit window.

4. To Move a component, Select the component and select Edit -Move command.

Making interconnection

1. Go to Connectivity –Nets – Show/Hide selected Incomplete Nets or click the icon in the Layout Menu. 
2. Move the mouse pointer over the device and click LMB to get the connectivity information, which shows the guide lines (or flight lines) for the inter connections of the components.
3. From the layout window Go to Create – Shape – Path/ Create wire or Create – Shape – Rectangle (for vdd and gnd bar) and select the appropriate Layers from the LSW window and Vias for making the inter connections.

Creating Contacts/Vias:

You will use the contacts or vias to make connections between two different layers.

1. Go to Create — Via or select  command to place different Contacts, as given in the table1.3.

Connection	Contact Type
For Metal1- Poly Connection	Metal1-Poly
For Metal1- Psubstrate Connection	Metal1-Psub
For Metal1- Nwell Connection	Metal1-Nwell

Table 1.3: Contact types

Saving the design:

1. Save your design by selecting File — Save or click to save the layout, and layout appears as shown in the figure1.22

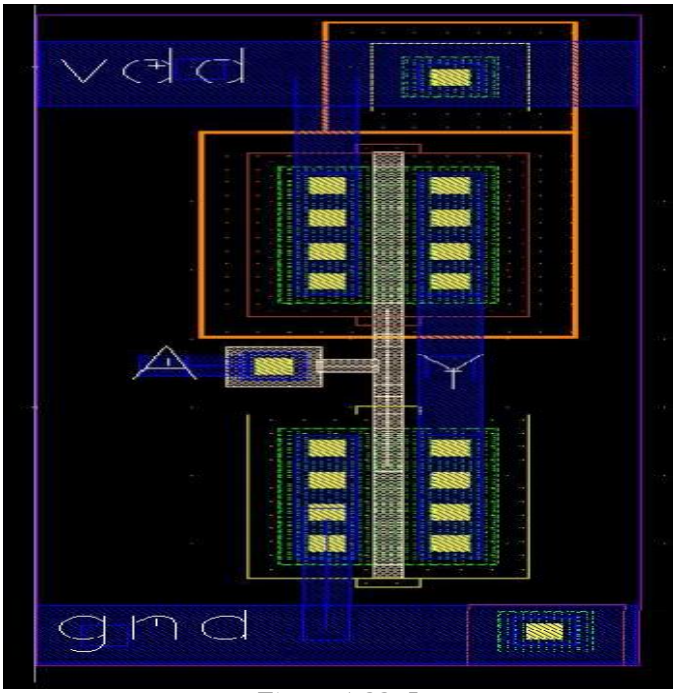


Figure 1.22: Layout

Physical Verification

ASSURA DRC:

Steps to perform design rule check

1. In the Inverter layout window Select Assura - Run DRC.

The DRC window appears as shown in figure1.23. The Library and Cellname are taken from the current design window, but rule file may be missing. Select the Technology as gpdk180. This automatically loads the rule file.

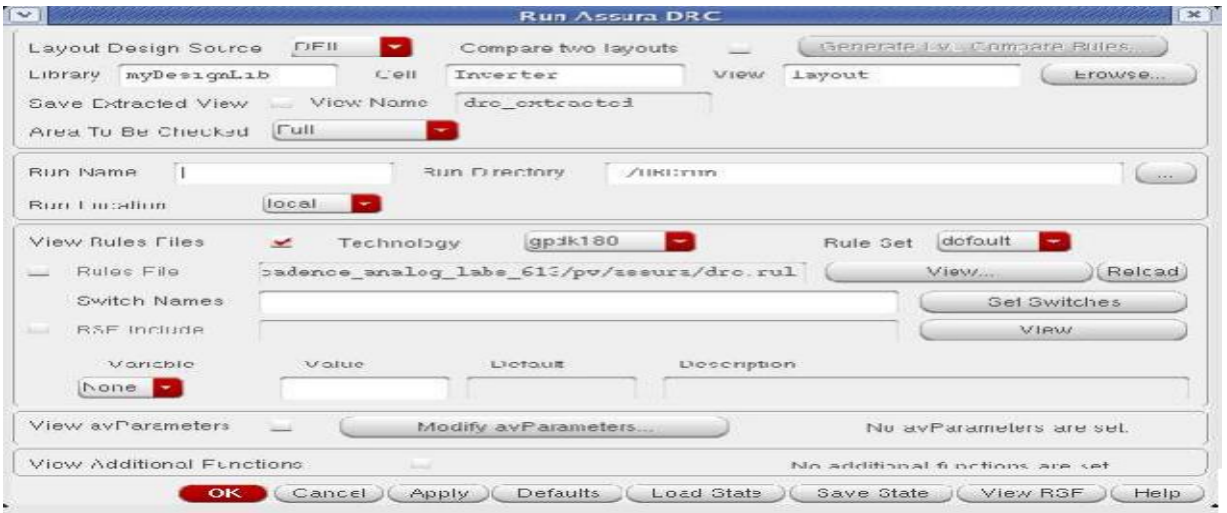


Figure 1.23: DRC Window

2. Click OK to start DRC.

3. A Progress window will appears. You can click on the watch log file to see the log file.
4. When DRC finishes, a dialog box appears asking you if you want to view your DRC results, and then click Yes to view the results of this run.
5. If there any DRC error exists in the design View Layer Window (VLW) and Error Layer Window (ELW) appears. Also the errors highlight in the design itself.
7. Click View – Summary in the ELW to find the details of errors.
8. You can refer to rule file also for more information, correct all the DRC errors and Re – run the DRC.
9. If there are no errors in the layout then a dialog box appears with No DRC errors found written in it, click on close to terminate the DRC run.

ASSURA LVS(Layout versus Schematic):

In this section we will perform the LVS check that will compare the schematic netlist and the layout netlist

Steps to perform LVS

1. Select Assura – Run LVS from the layout window.
The Assura Run LVS window appears. It will automatically load both the schematic and layout view of the cell.
2. Set the LVS window as shown in figure1.24 and click OK.

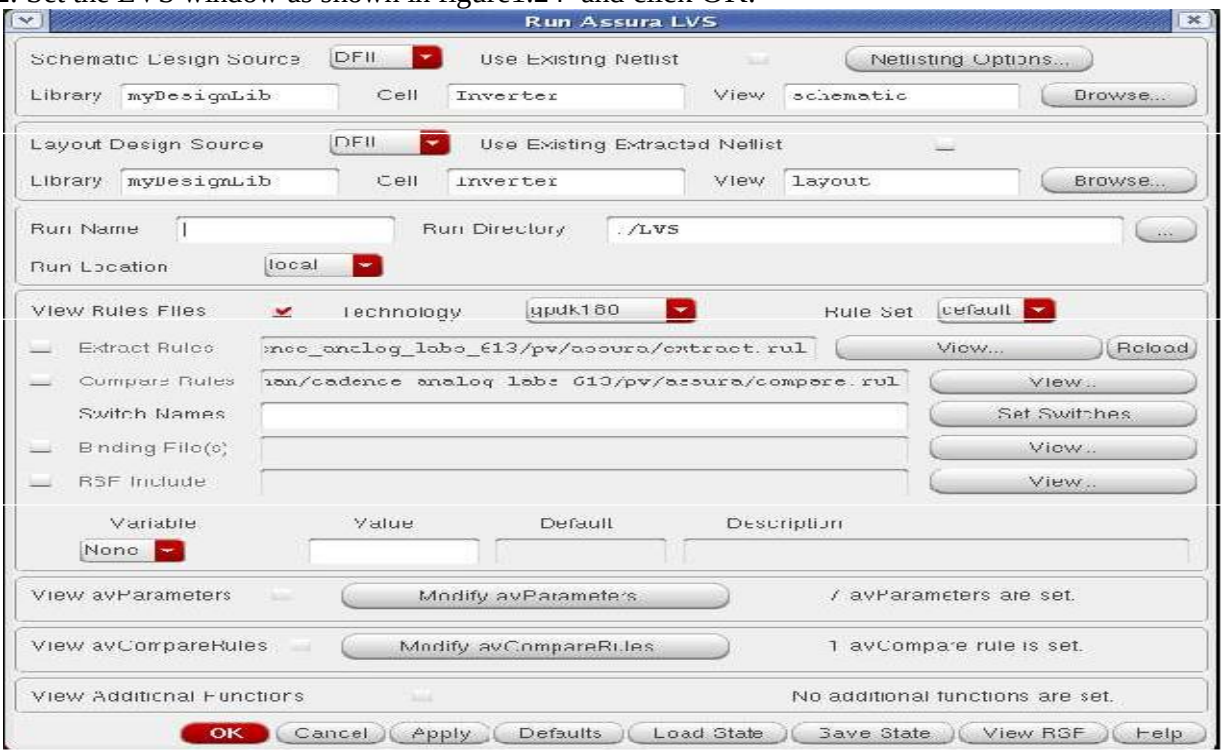


Figure1.24:LVS Window

3. The LVS start and a Progress window appears.
4. If the schematic and layout matches completely, you will get the window displaying Schematic and Layout Match.
5. If the schematic and layout do not matches, a form informs that the LVS completed successfully and asks if you want to see the results of this run.
6. Click Yes in the window. LVS debug window appears, and you are directed into LVS debug environment.
7. In the LVS debug window you can find the details of mismatches and you need to correct all those mismatches and Re – run the LVS till you will be able to match the schematic with layout.

Assura RCX

In this section we will extract the RC values from the layout and perform analog circuit simulation on the designs extracted with RCX. Before using RCX to extract parasitic devices for

simulation, the layout should match with schematic completely to ensure that all parasites will be backannoted to the correct schematic nets.

Running RCX :

1. From the layout window Go to Assura – Run RCX.
2. Set the Assura parasitic extraction form as shown in the figure1.25. Select output type under Setup tab of the form.

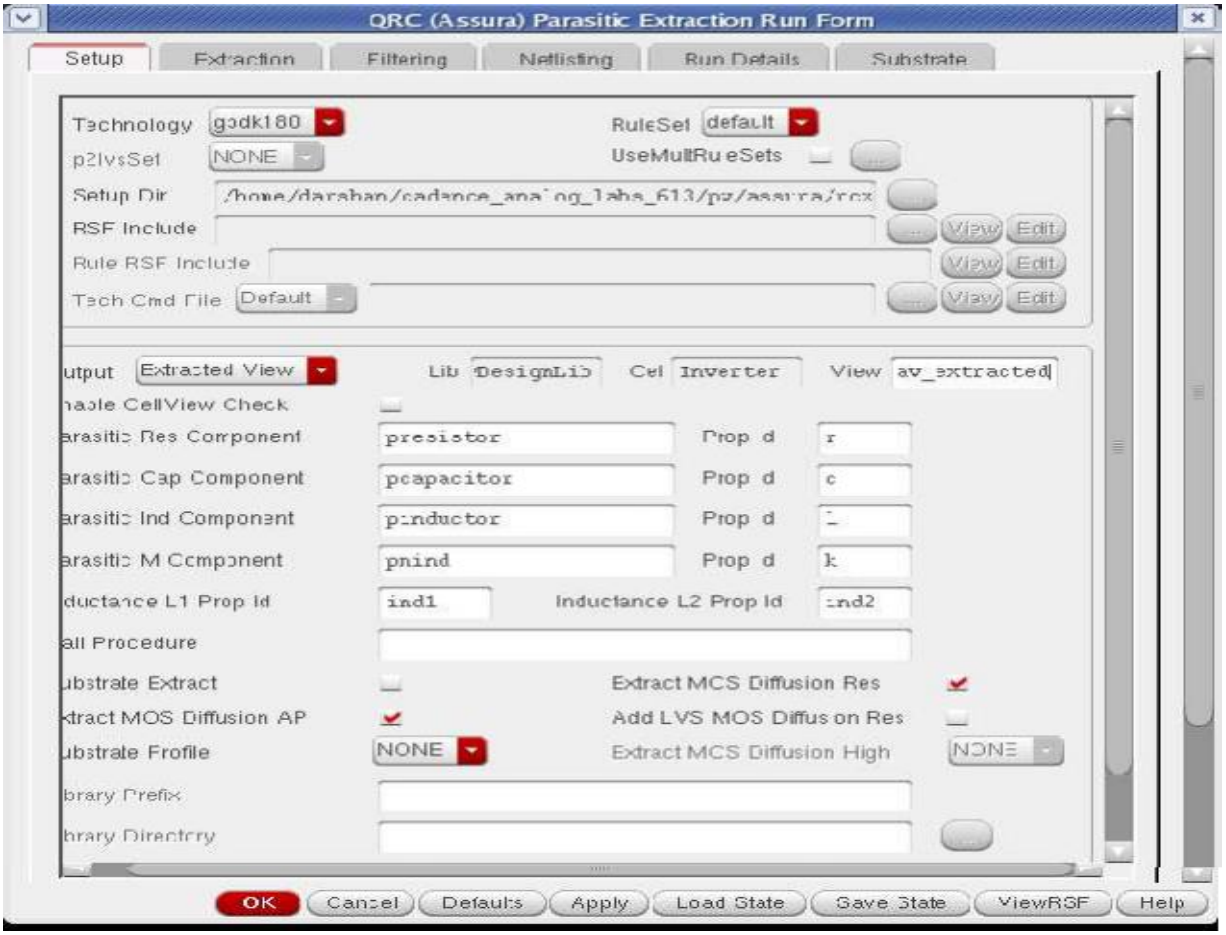


Figure1.25: Parasitic extraction Run Form

3. In the Extraction tab of the form, choose Extraction type, Cap Coupling Mode and specify the Reference node for extraction as shown in figure1.26

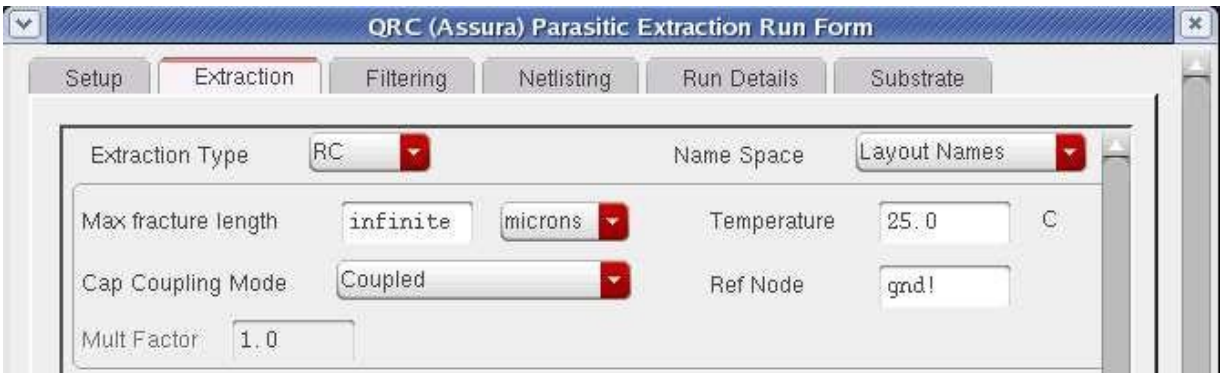


Figure1.26: Parasitic extraction Run Form

4. In the Filtering tab of the form, Enter Power Nets as vdd!,vss! And Enter Ground Nets as gnd! As shown in figure1.27

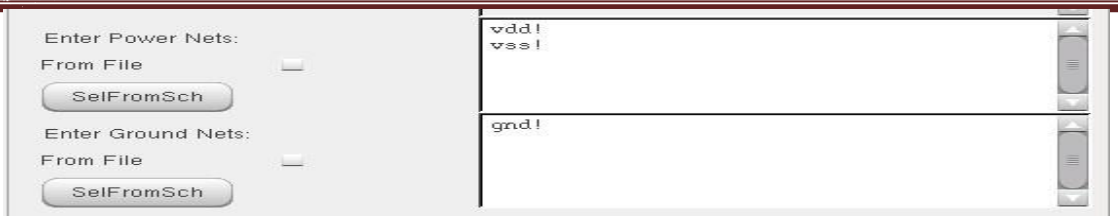


Figure1.27: Parasitic extraction Run Form

5. Click OK in the Assura parasitic extraction form when done. The RCX progress form appears, in the progress form click Watch log file to see the output log file.
5. When RCX completes, a dialog box appears, informs you that Assura RCX run Completed successfully.
6. You can open the av_extracted view from the library manager and view the parasitic. As shown in figure 1.28

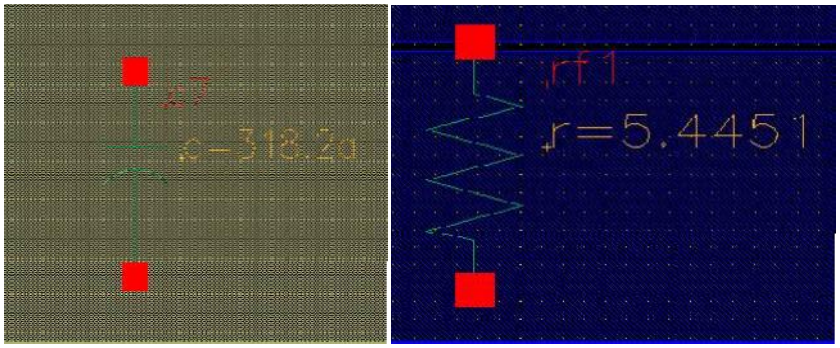
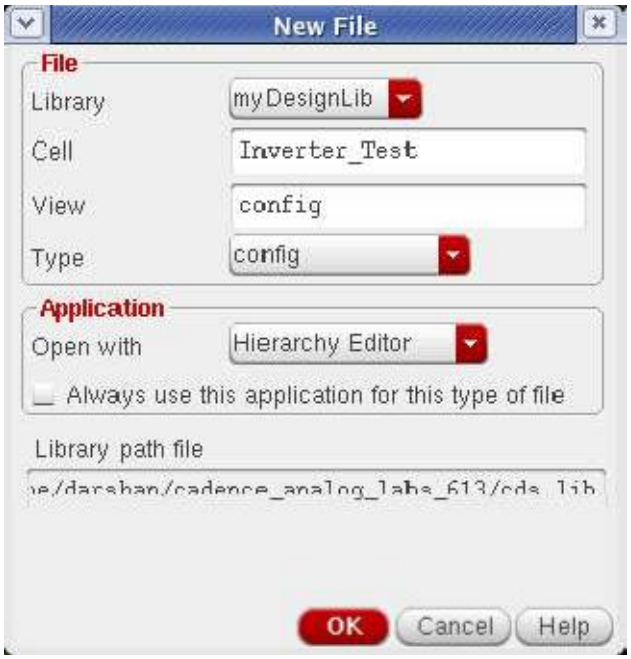


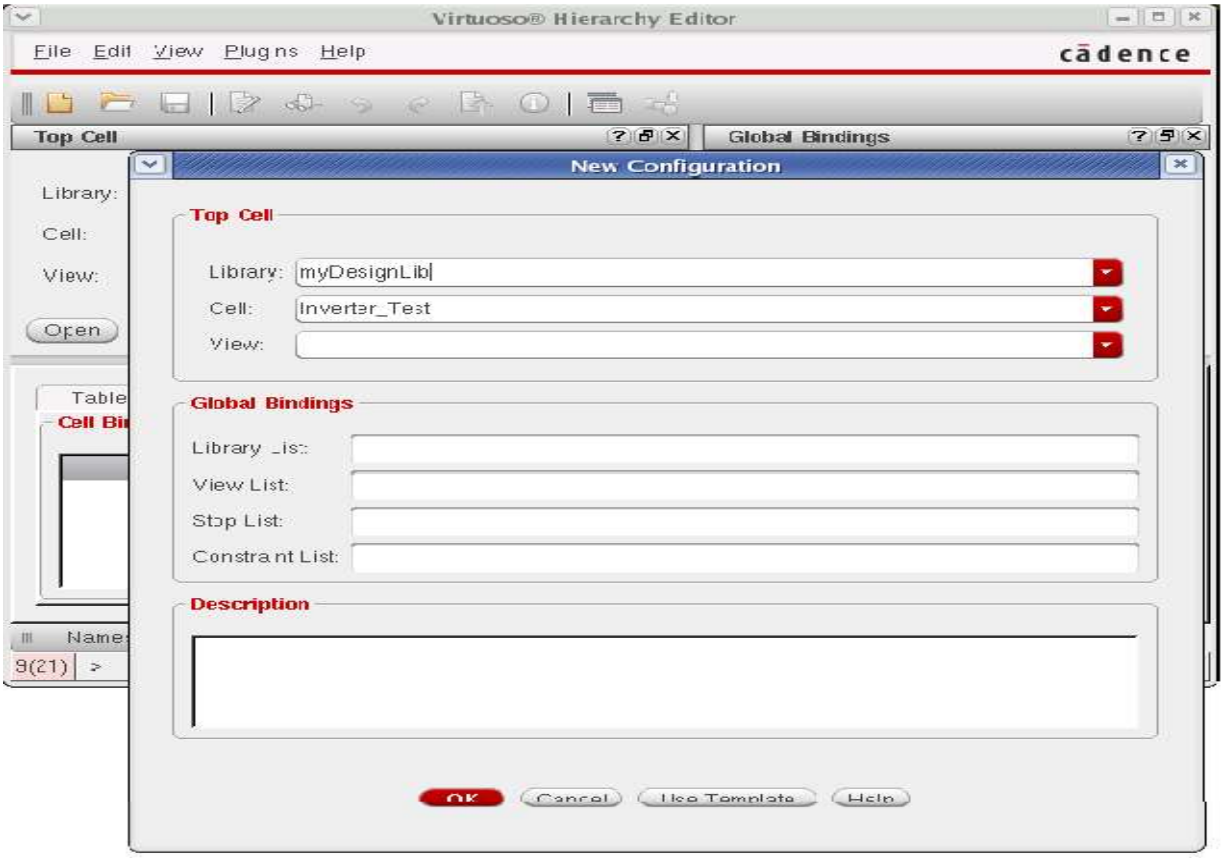
Figure 1.28: Parasitic extraction

Creating the Configuration View

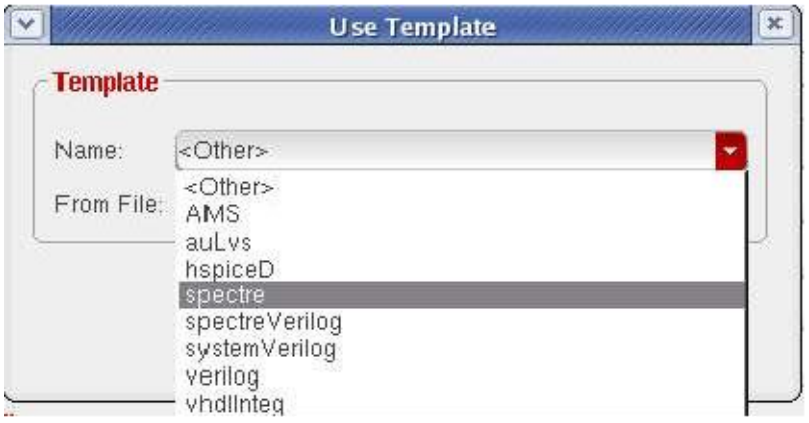
In this section we will create a config view and with this config view we will run the simulation with and without parasitic.
1. In the CIW or Library Manager, execute File – New – Cellview
2. In the Create New file form, set the following:



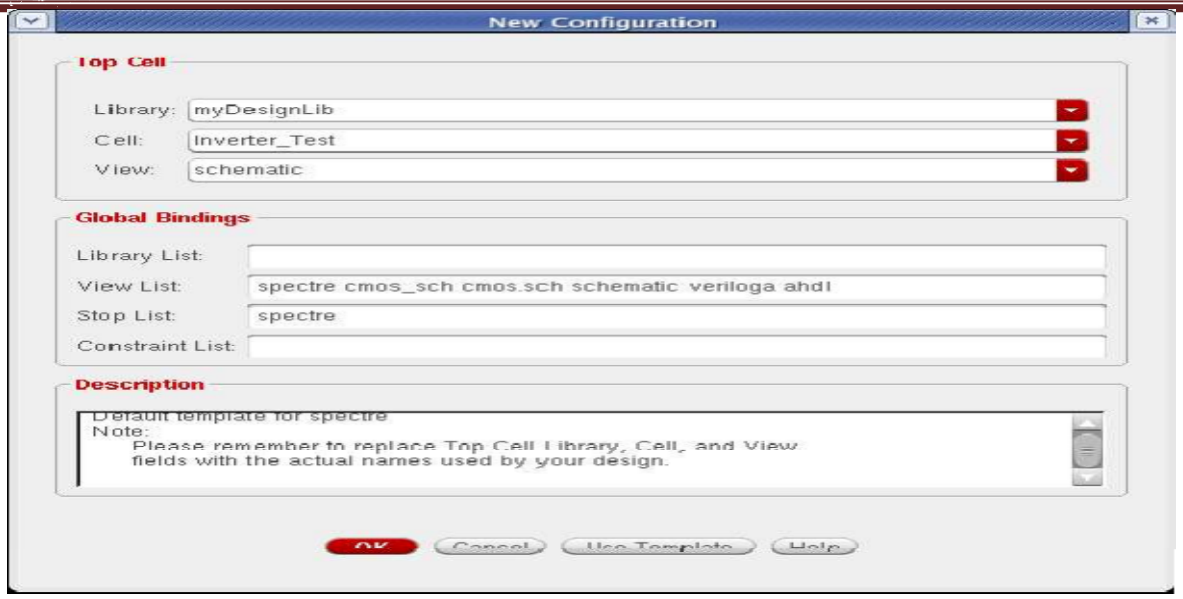
3. Click OK in createNew File form.
The Hierarchy Editor form opens and a New Configuration form opens in front of it.



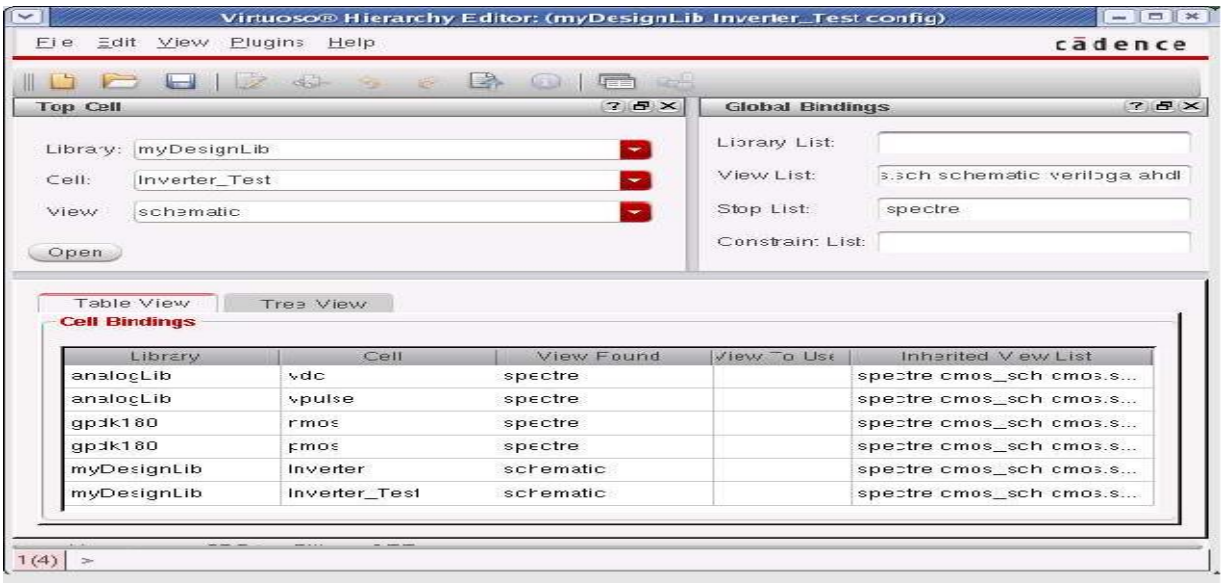
4. Click Use template at the bottom of the New Configuration form and select Spectrein the cyclic field and click OK.
The Global Bindings lists are loaded from the template.



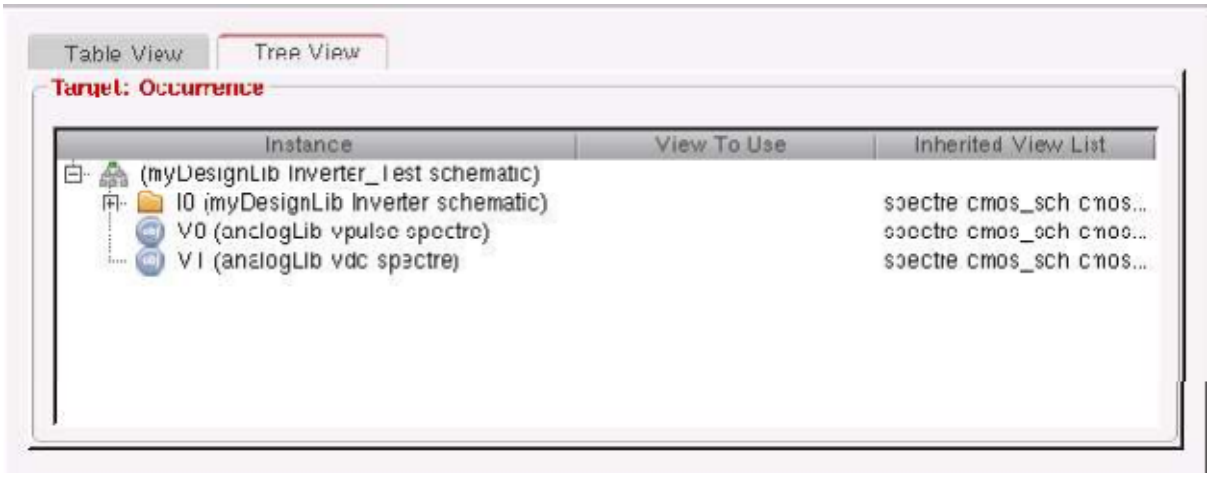
5. Change the Top Cell View to schematic and remove the default entry from the Library List field.
6. Click OK in the New Configuration form.



The hierarchy editor displays the hierarchy for this design using table format.



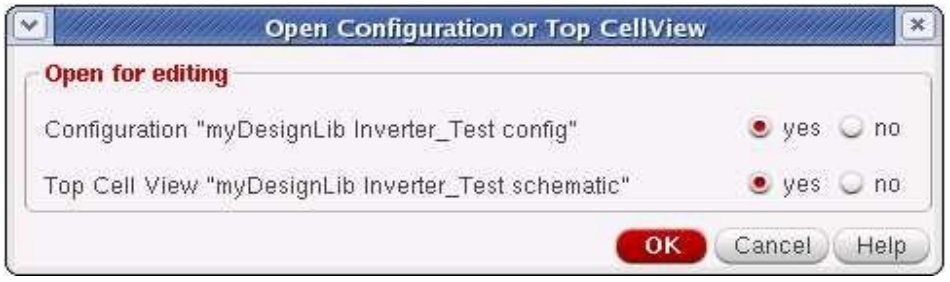
7. Click the Tree View tab. The design hierarchy changes to tree format. The form should look like this.



8. Save the current configuration. 
9. Close the Hierarchy Editor window. Execute File – Close Window.

To run the Circuit without Parasites
1. From the Library Manager open Inverter_TestConfig view.
Open Configuration or Top cellview form appears.

To run the Circuit without Parasites
1. From the Library Manager open Inverter_TestConfig view.
Open Configuration or Top cellview form appears.



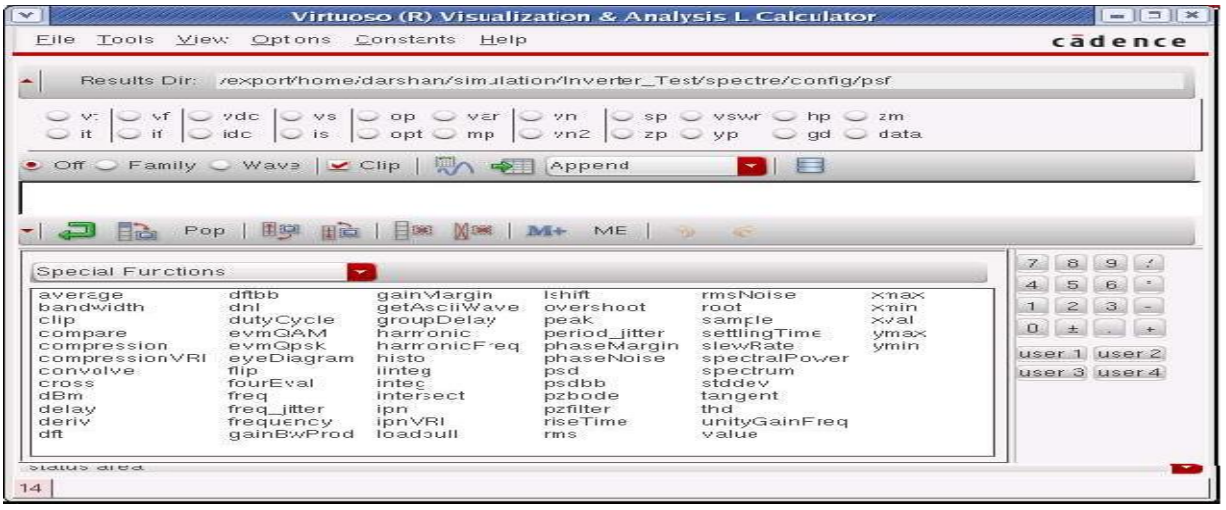
2. In the form, turn on the both cyclic buttons to Yes and click OK.
The Inverter_Test schematic and Inverter_Testconfig window appears. Notice the window banner of schematic also states Config: myDesignLibInverter_Testconfig.
3. Execute Launch – ADE L from the schematic window.
4. Now you need to follow the same procedure for running the simulation. Executing Session–Load state, the Analog Design Environment window loads the previous state.

5. Click Netlist and Run icon to start the simulation.  The simulation takes a few seconds and then waveform window appears.

6. In the CIW, note the netlisting statistics in the Circuit inventory section. This list includes all nets, designed devices, source and loads. There are no parasitic components. Also note down the circuit inventory section.

Measuring the Propagation Delay:

1. In the waveform window execute Tools – Calculator. 
The calculator window appears as shown in the figure



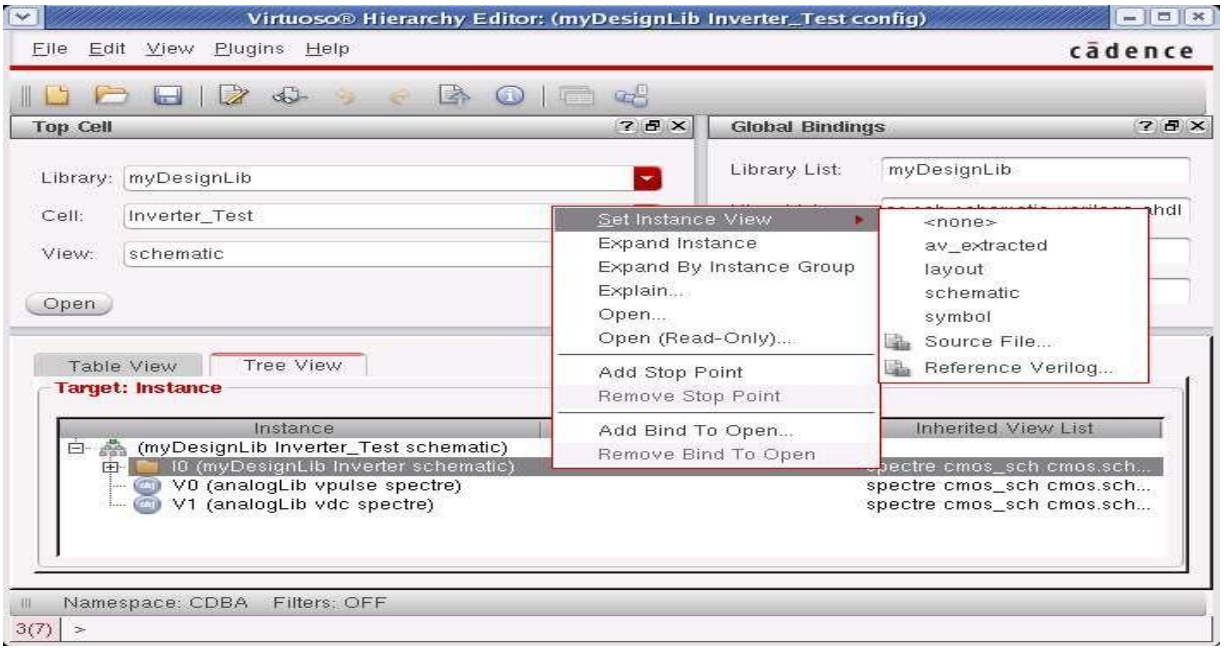
2. From the functions select delay, this will open the delay data panel.
3. Place the cursor in the text box for Signal1, select the wave button and select the input waveform from the waveform window.
4. Repeat the same for Signal2, and select the output waveform.

5. Set the Threshold value 1 and Threshold value 2 to 0.9; this directs the calculator to calculate delay at 50% i.e. at 0.9 volts.
6. Execute OK and observe the expression created in the calculator buffer.

7. Click on Evaluate the buffer icon  to perform the calculation, note down the value returned after execution.
8. Close the calculator window.

To run the Circuit with Parasites:

In this exercise, we will change the configuration to direct simulation of the av_extracted view which contains the parasites.
1. Open the same Hierarchy Editor form, which is already set for Inverter_Testconfig.
2. Select the Tree View icon: this will show the design hierarchy in the tree format.
3. Click right mouse on the Inverter schematic.
A pull down menu appears. Select av_extracted view from the Set Instance view menu, the View to use column now shows av_extracted view.



4. Click on the Recompute the hierarchy icon,  the configuration is now updated from schematic to av_extracted view.

6. From the Analog Design Environment window click Netlist and Run to  start the simulation again.
7. When simulation completes, note the Circuit inventory conditions, this time the list shows all nets, designed devices, sources and parasitic devices as well.
8. Calculate the delay again and match with the previous one. Now you can conclude how much delay is introduced by these parasites, now our main aim should to minimize the delay due to these parasites so number of iteration takes place for making an optimize layout.

Generating Stream Data

Streaming Out the Design

1. Select File – Export – Stream from the CIW menu and Virtuoso Xstream out form appears change the following in the form.

2. Click on the Options button.
3. In the StreamOut-Options form select Use Automatic Mapping under Layers tab and click OK.
4. In the Virtuoso XStream Out form, click Translate button to start the stream translator.
5. The stream file Inverter.gds is stored in the specified location.

Streaming In the Design

1. Select File – Import – Stream from the CIW menu and change the following in the form.

You need to specify the gpdk180_oa22.tf file. This is the entire technology file that has been dumped from the design library.

2. Click on the Options button.
3. In the StreamOut-Options form select under Layers tab and click OK.
4. In the Virtuoso XStream Out form, click Translate button to start the stream translator.
5. From the Library Manager open the Inverter cellview from the GDS_LIB library and notice the design.
6. Close all the windows except CIW window, which is needed for the next lab.

Experiment 2: NAND and NOR

Aim: Design the NAND and NOR with given specifications.

a) Draw the schematic and verify the following
   - DC Analysis
   - Transien tAnalysis

b) Draw the Layout and verify the DRC,ERC

c) Check for LVS

d) Extract RC and back annotate the same and verify the Design

NAND

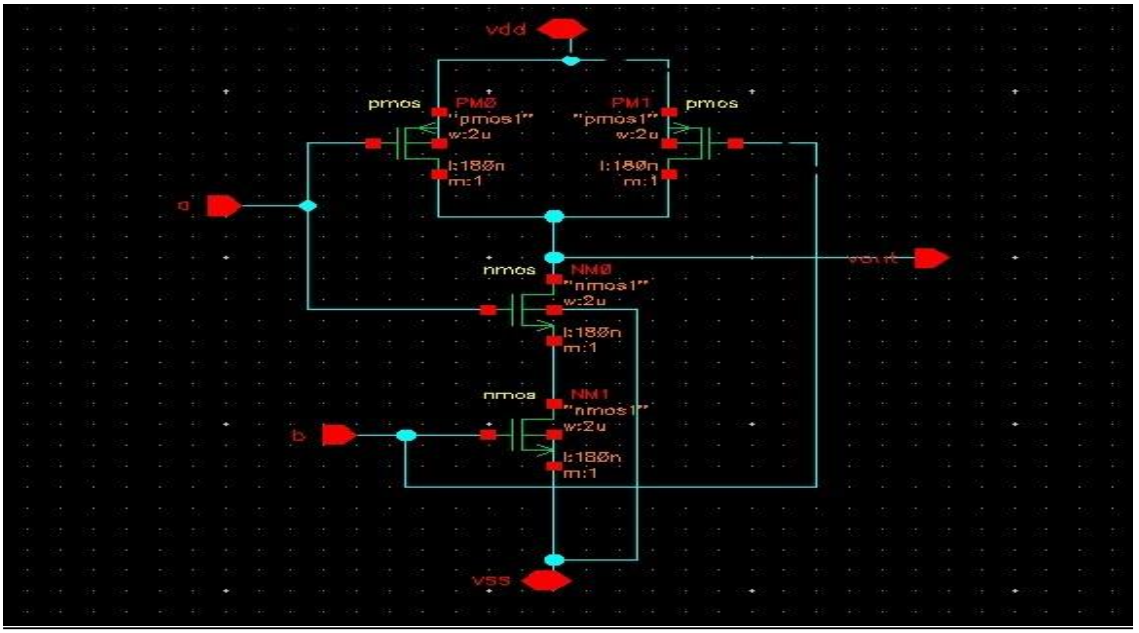


Figure 2.1: NAND Gate Schematic

Schematic Entry:

The table2.1 shows the list of components for building the NAND Gate schematic.

Library name	Cell Name	Properties/Comments
gpdk180	pmos	For M0: Model name = pmos1, **W= wp**, L=180n
gpdk180	nmos	For M1: Model name – nmos1, W– 2u, L=180n

Table2.1: the list of components for building the NAND Gate schematic.

Type the following in the ADD pin form in the exact order leaving space between the pin names.

Names	Direction
a b	Input
vout	Output
vddvss	Input

Symbol Creation:

Figure 2.2 shows the symbol creation of NAND Gate

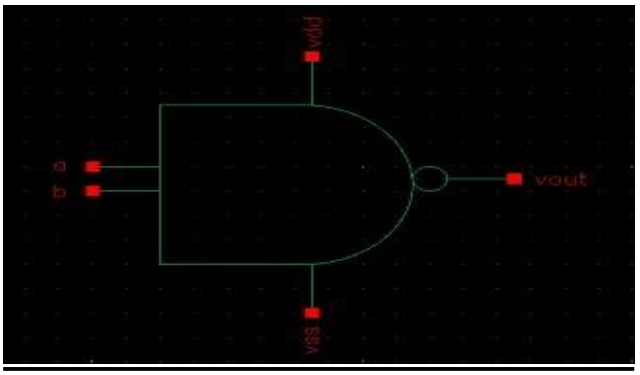


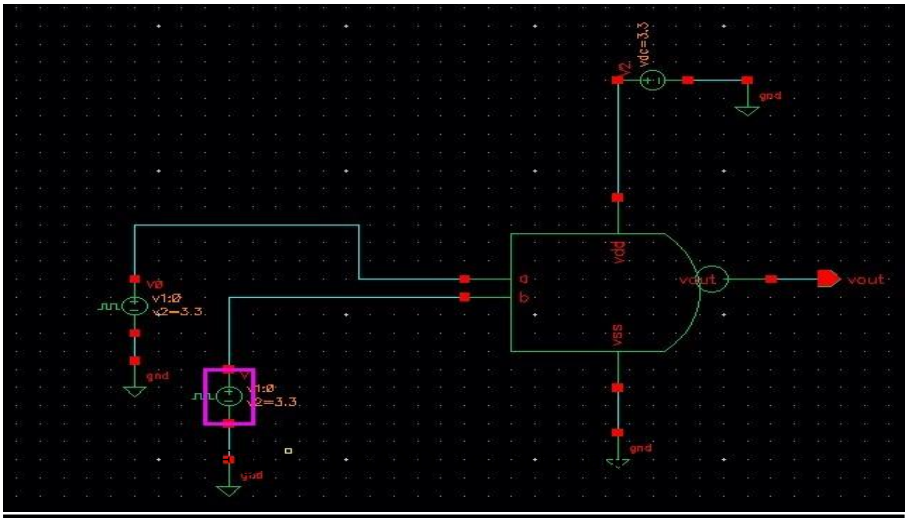
Figure 2.2: NAND Gate Symbol

Building the NAND Test Design:

Using the component list and Properties/Comments in the table1.2, build the NAND_test schematic as shown in figure 2.3

Library name	Cellview name	Properties/Comments
myDesignLib	NAND	Symbol
analogLib	vpulse	v1=0, v2=3.3, ton=2ns, T=10ns
analogLib	vpulse	v1=0, v2=3.3, ton=10ns, T=20ns
analogLib	vdc , gnd	vdd=3.3

Table 2.2: Components list for building NAND test Design



Simulation of NAND --ADE window and waveform as shown in figure2.4 and figure 2.5.

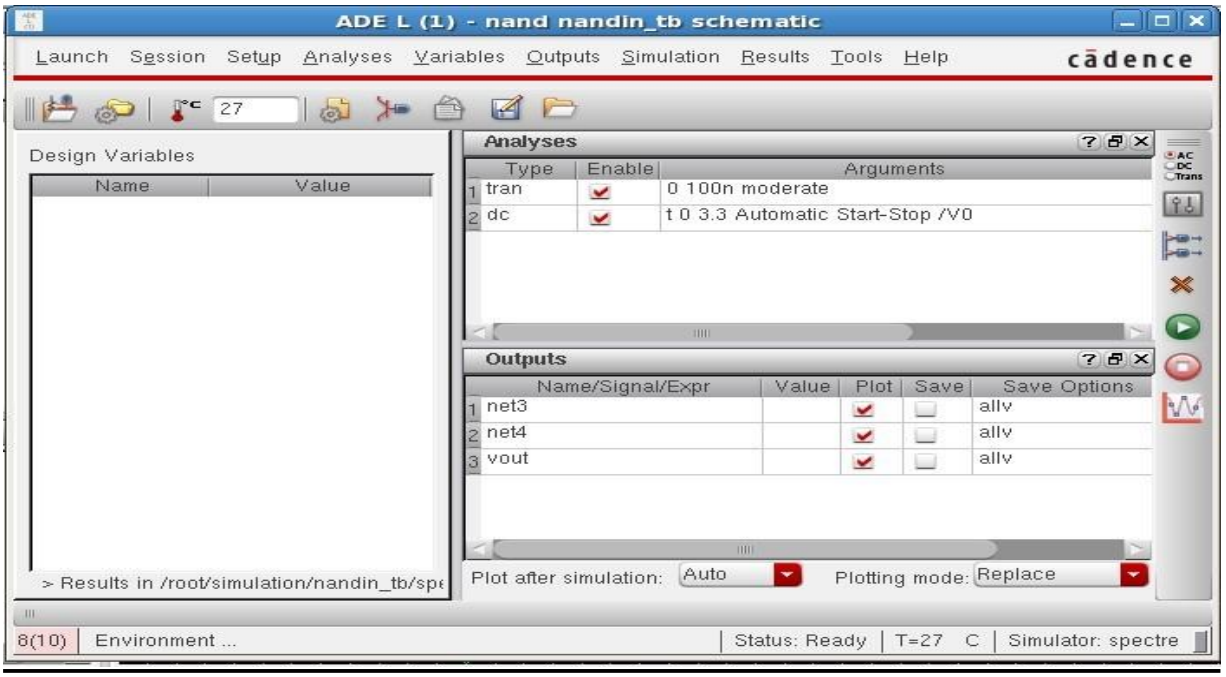


Figure 2.4: NAND Schematic ADE window

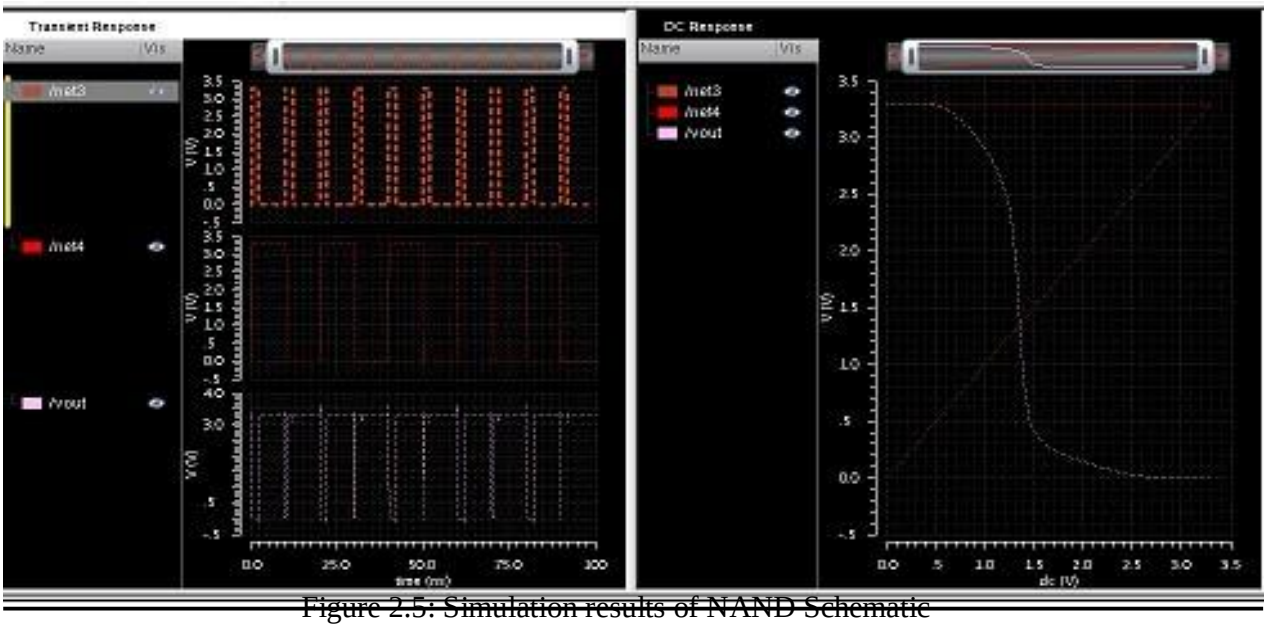


Figure 2.5: Simulation results of NAND Schematic

Creating a layout of NAND:

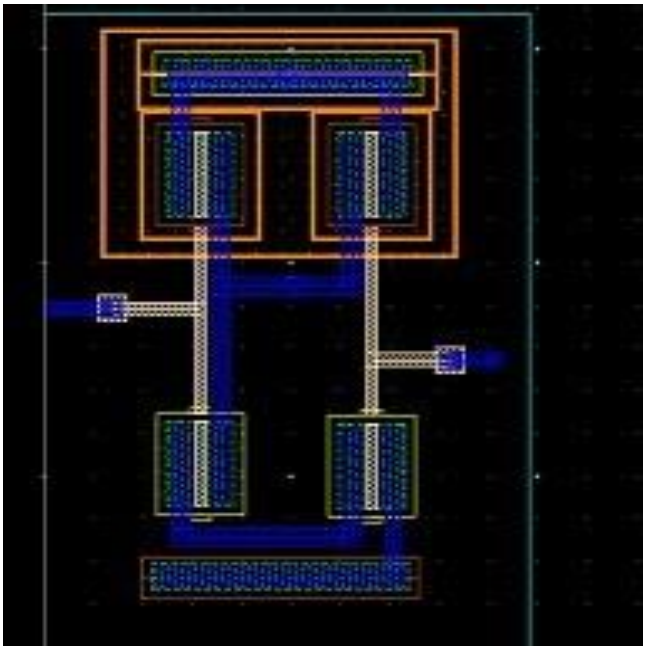


Figure 2.6: NAND Layout

NOR

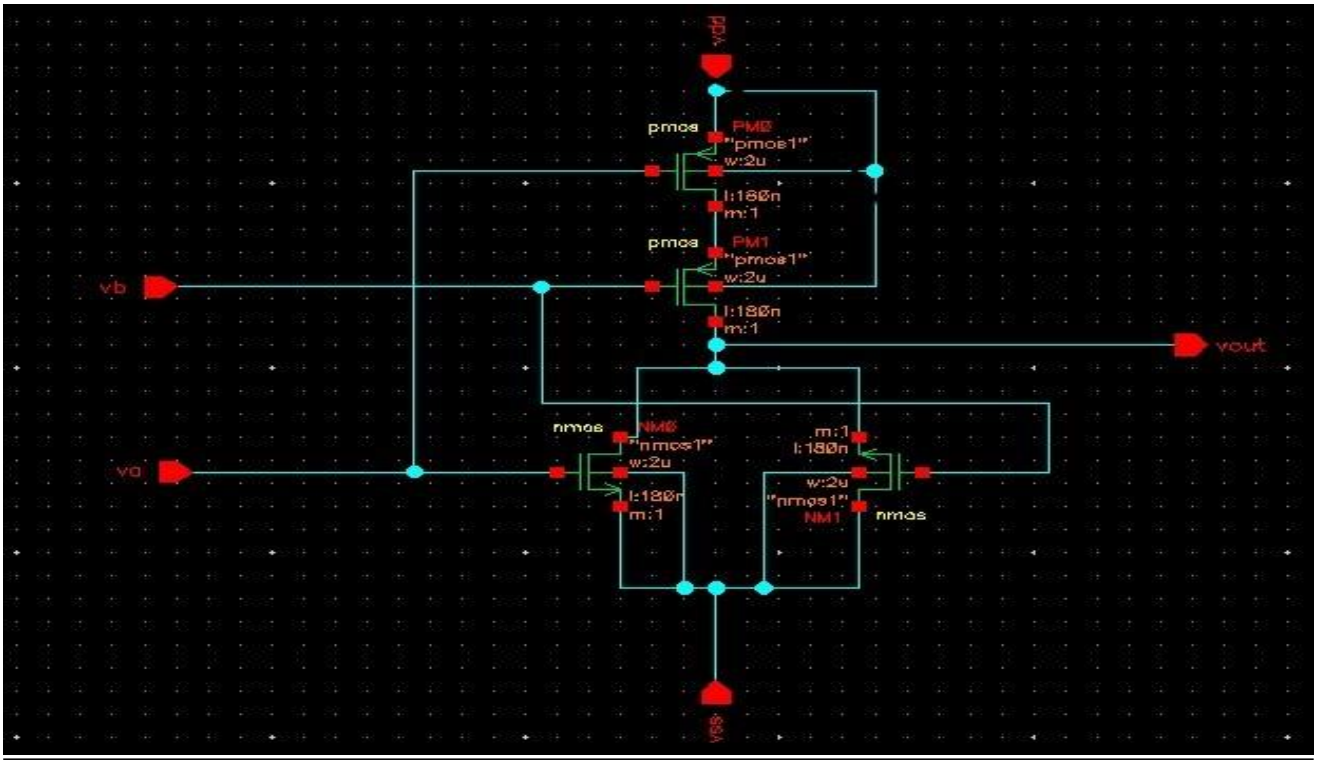


Figure2.7: NOR Schematic

Schematic Entry:

The table2.3 shows the list of components for building the NOR Gate schematic.

Library name	Cell Name	Properties/Comments
gpdk180	pmos	For M0: Model name = pmos1, W= wp, L=180n
gpdk180	nmos	For M1: Model name = nmos1, W= 2u, L=180n

Table2.3: The list of components for building the NOR Gate schematic.

Type the following in the ADD pin form in the exact order leaving space between the pin names.

Pin Names	Direction
vavb	Input
vout	Output
vddvss	Input

Symbol Creation:

Figure 2.8 shows the symbol creation of NOR Gate

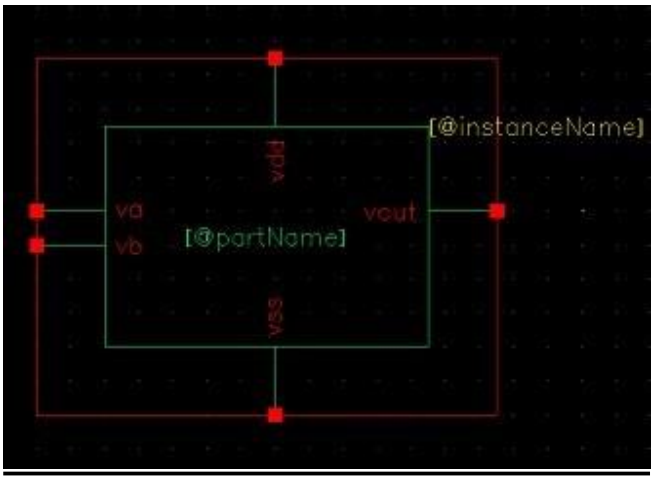


Figure2.8: NOR gate symbol

Building the NOR Test Design:

Using the component list and Properties/Comments in the table2.4, build the NOR_test schematic as shown in figure2.9.

Library name	Cellview name	Properties/Comments
myDesignLib	NOR	Symbol
analogLib	vpulse	v1=0, v2=1.8, ton=5us, T=10us
analogLib	vpulse	v1=0, v2=1.8, ton=10us, T=20us
analogLib	vdc , gnd	vdd=1.8

Table 2.4: Components list for building NOR_test Design

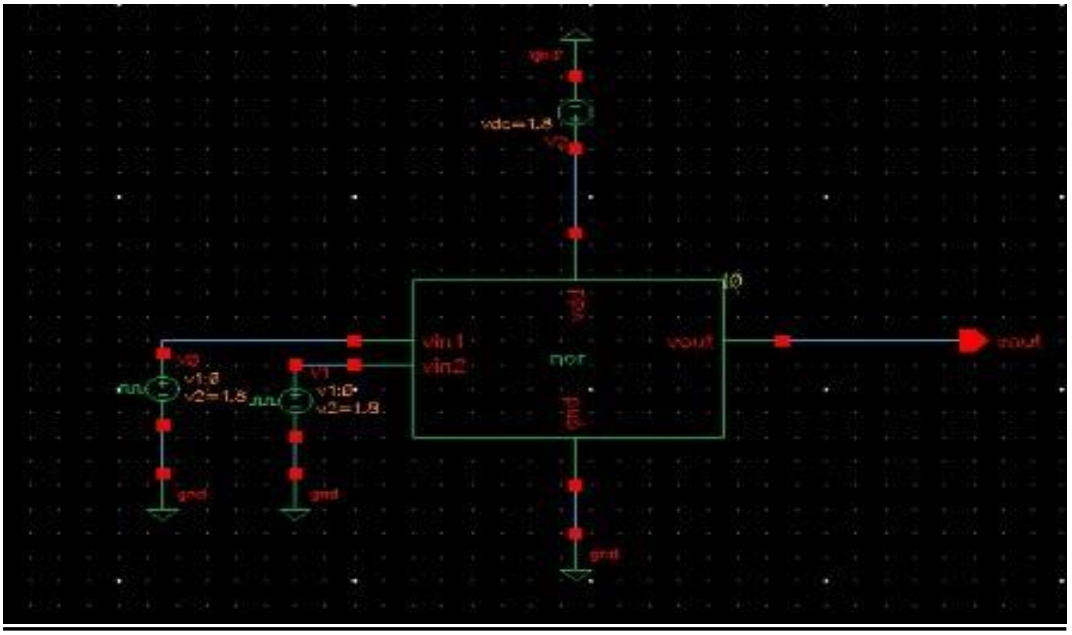


Figure 2.9: NOR_Test Schematic

Analog Simulation with Spectre:
The simulation of NOR, waveform is shown in the figure 2.10

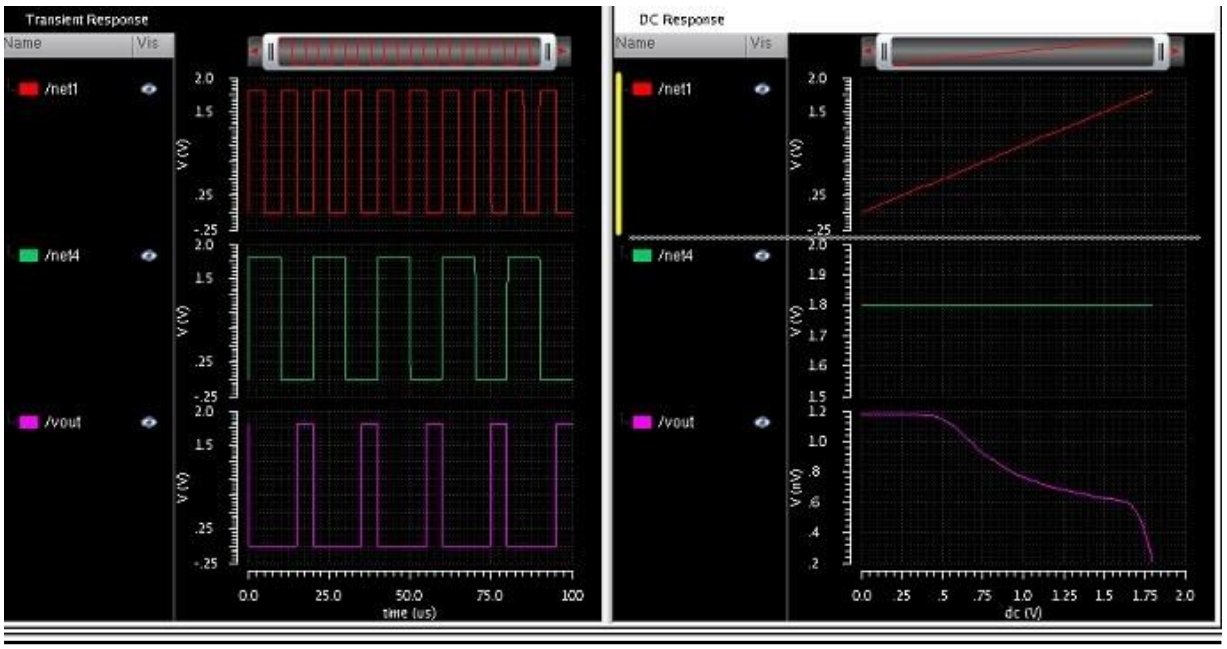


Figure2.10: Simulation results of NOR Schematic

Creating a layout of NOR:

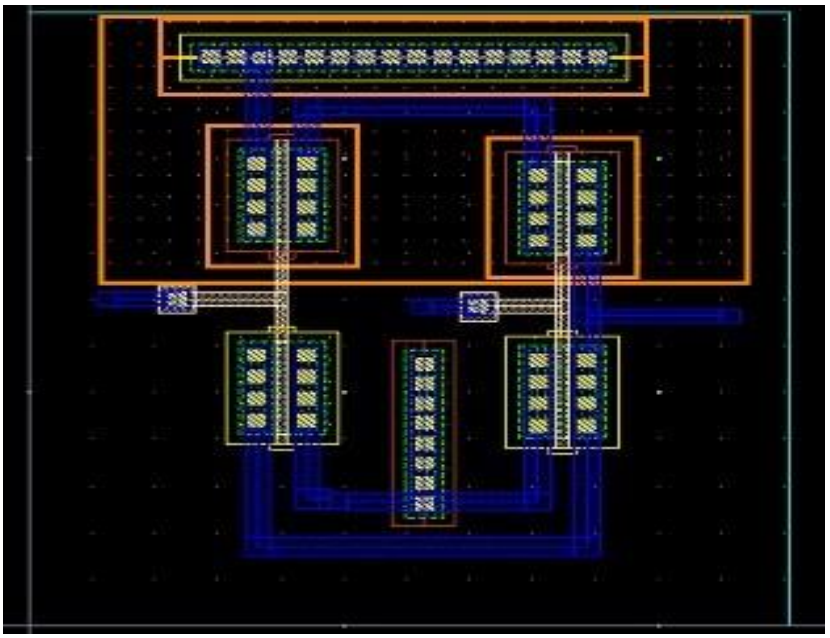


Figure2.11: NOR Layout

Experiment 3: COMMON SOURCE AND COMMON DRAIN AMPLIFIER.

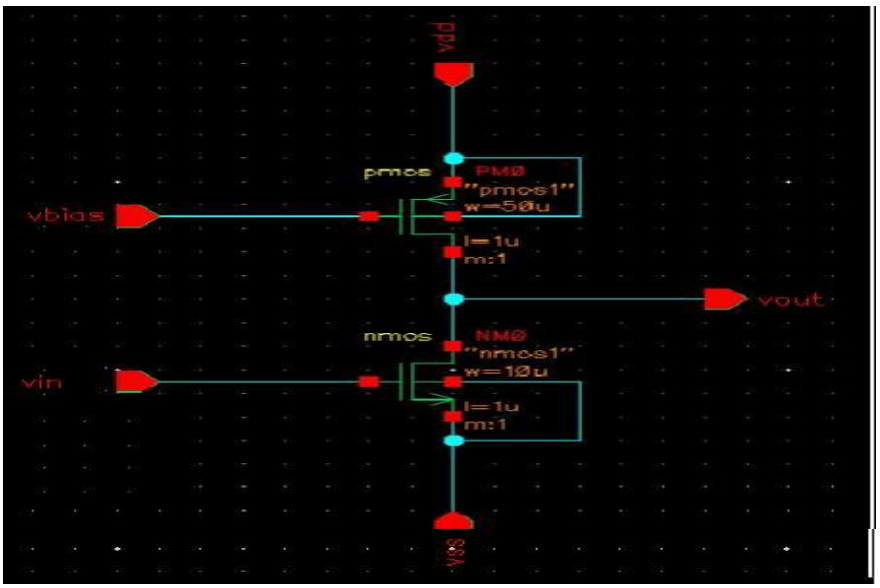
Aim: Design the following circuits with given specifications for common source and common drain amplifier completing the design flow mentioned below:

a) Draw the schematic and verify the following
   - DC Analysis
   - AC Analysis
   - Transient Analysis

b) Draw the Layout and verify the DRC,ERC

c) Check for LVS

Schematic Capture:



Schematic Entry:

Use the techniques learned in the Lab1 and Lab2 to complete the schematic of Common Source Amplifier.

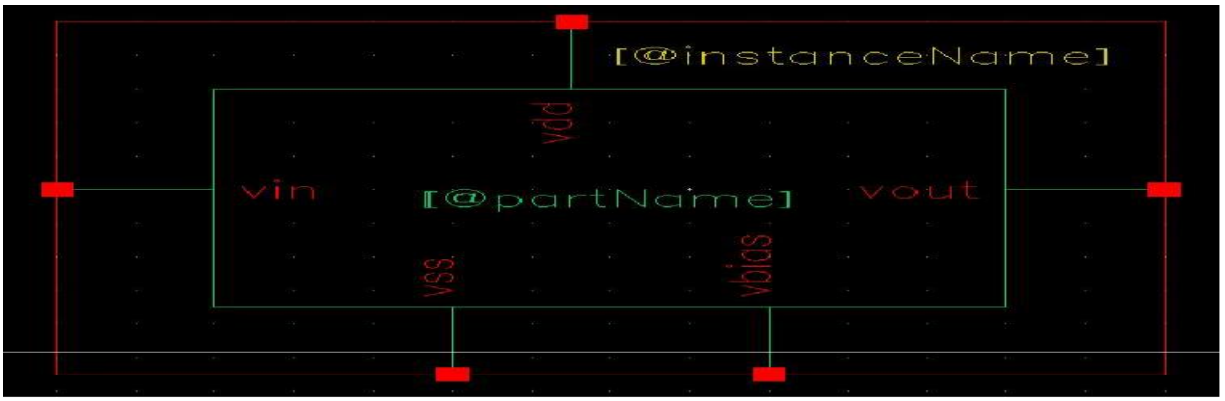
This is a table of components for building the Common Source Amplifier schematic.

Library name	Cell Name	Properties/Comments
gpdk180	Pmos	Model Name = pmos1; W= 50u ; L= 1u
gpdk180	Nmos	Model Name =nmos1; W= 10u ; L= 1u

Type the following in the ADD pin form in the exact order leaving space between the pin names.

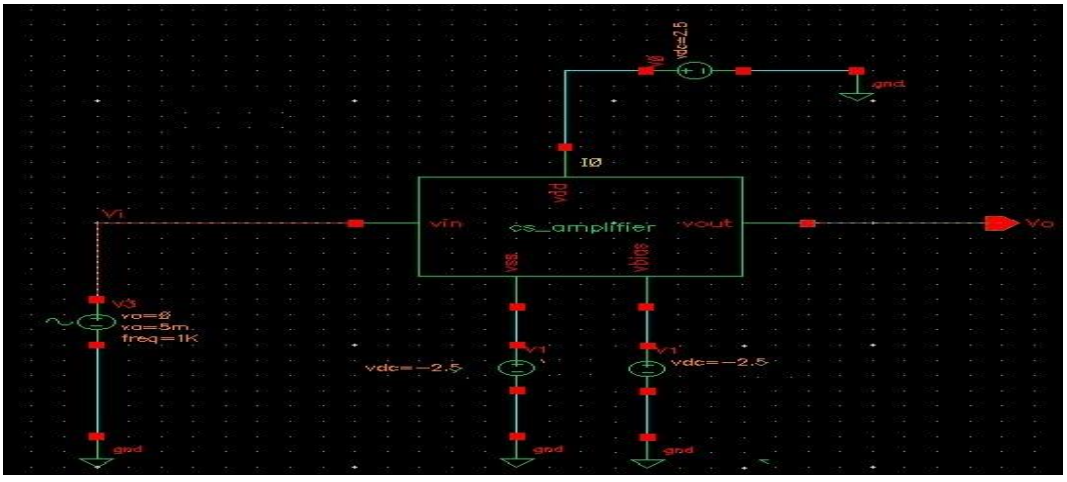
Pin Names	Direction
vin vbias	Input
vout	Output
vddvss	Input

Symbol Creation:
Use the techniques learned in the Lab1 and Lab2 to complete the symbol of cs-amplifier



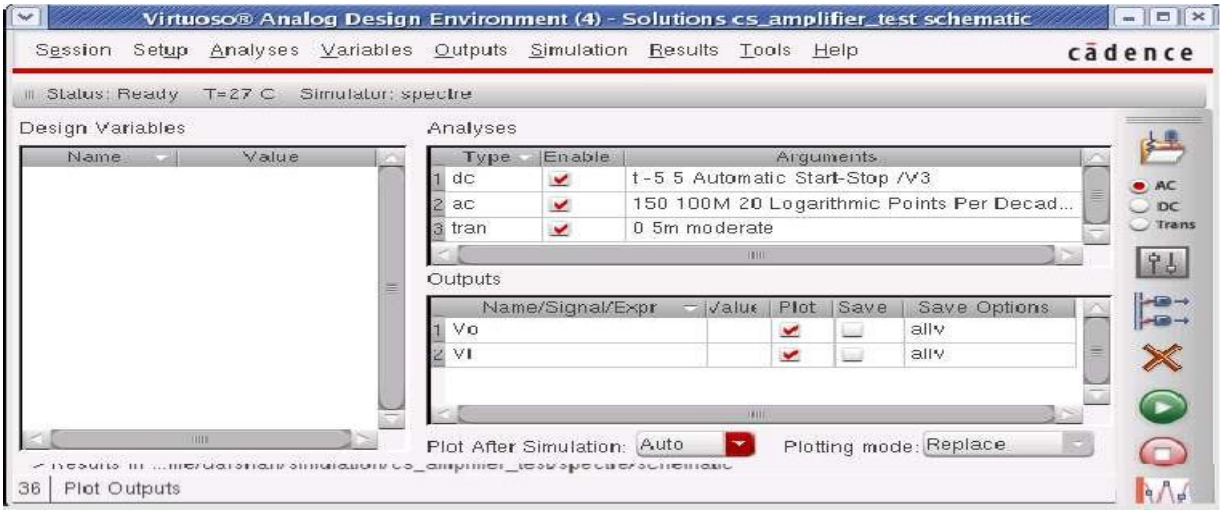
Building the Common Source Amplifier Test Design:
Using the component list and Properties/Comments in the table, build the cs-amplifier_test schematic as shown below.

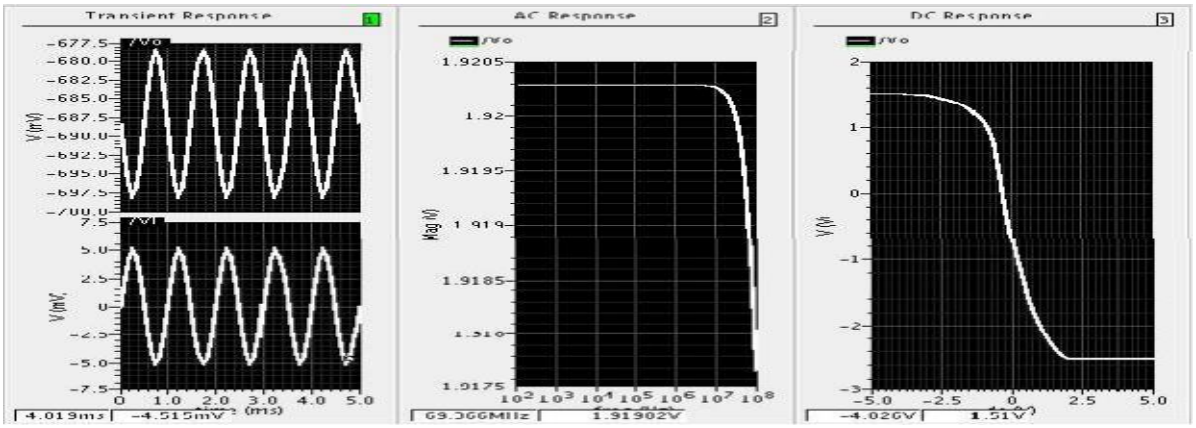
Library name	Cellview name	Properties/Comments
myDesignLib	cs_amplifier	Symbol
analogLib	vsin	Define pulse specification as AC Magnitude= 1; DC Voltage= 0; Offset Voltage= 0; Amplitude= 5m; Frequency= 1K
analogLib	vdd,vss,gnd	vdd=2.5 ; vss= -2.5 vbias=-2.5



Analog Simulation with Spectre:

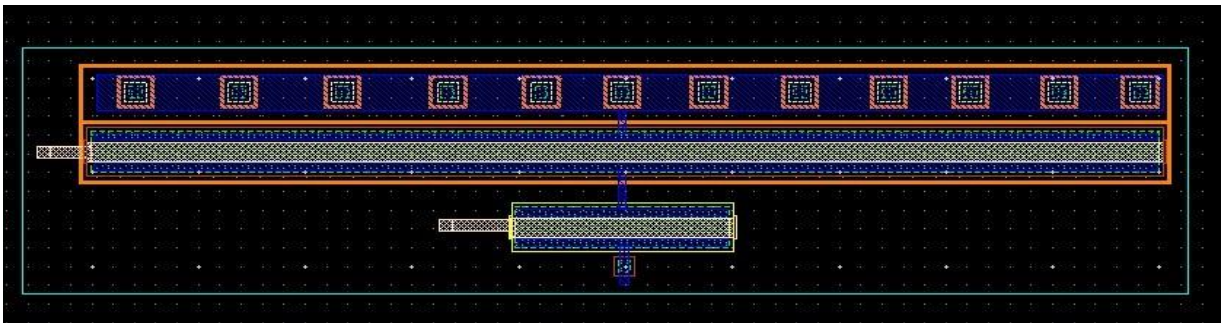
Use the techniques learned in the Lab1 and Lab2 to complete the simulation of cs_amplifier, ADE window and waveform should look like below.





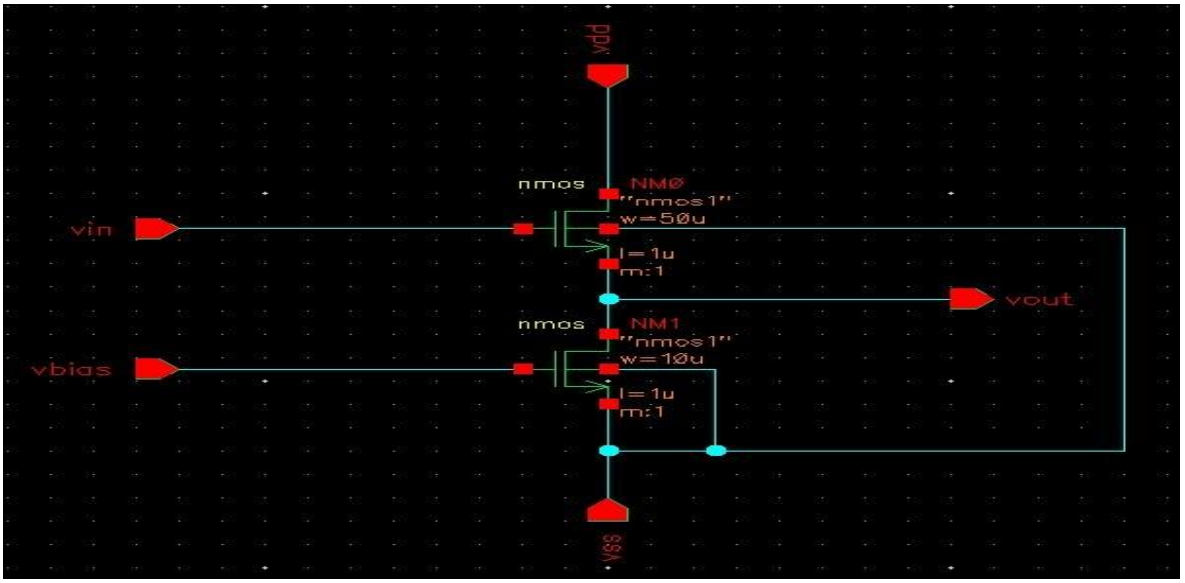
Creating a layout view of Common Source Amplifier:

Use the techniques learned in the Lab1 and Lab2 to complete the layout of cs_amplifier. Complete the DRC, LVS check using the assura tool. Extract RC parasites for back annotation and Re-simulation.



COMMON DRAIN AMPLIFIER:

Schematic Capture:



Schematic Entry:

Use the techniques learned in the Lab1 and Lab2 to complete the schematic of Common Drain Amplifier. This is a table of components for building the Common Drain Amplifier schematic.

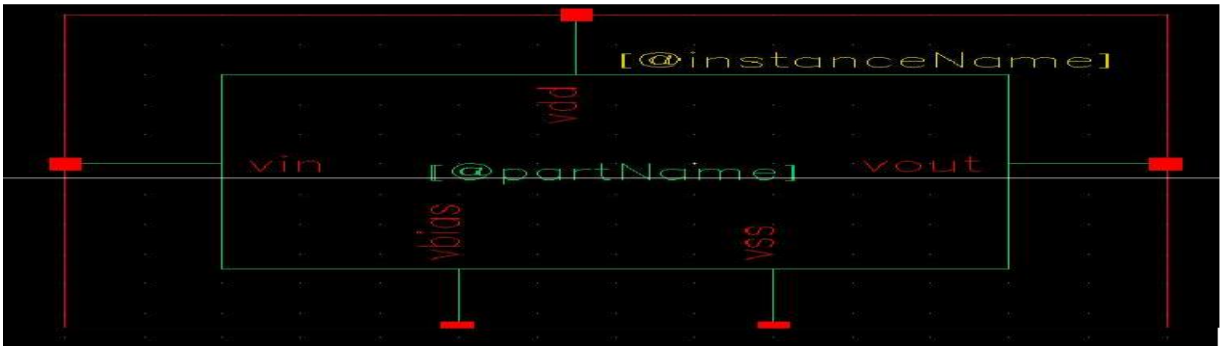
Library name	Cell Name	Properties/Comments
gpdk180	nmos	Model Name = nmos1; W= 50u ; L= 1u
gpdk180	nmos	Model Name = nmos1; W= 10u ; L= 1u

Type the following in the ADD pin form in the exact order leaving space between the pin names.

Pin Names	Direction
vin, vbias	Input
vout	Output
vddvss	Input

Symbol Creation:

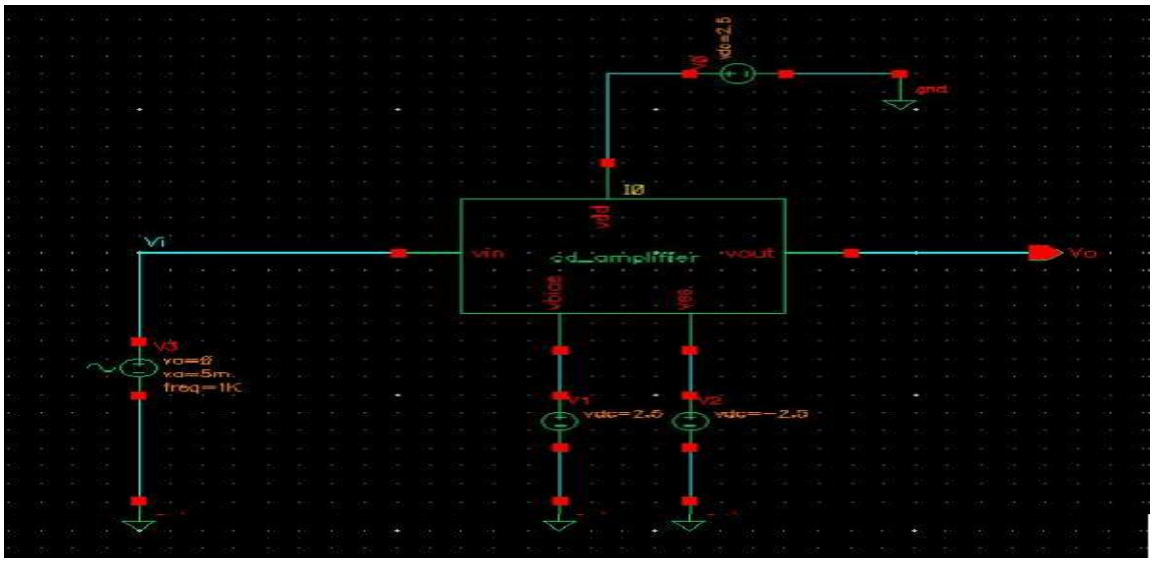
Use the techniques learned in the Lab1 and Lab2 to complete the symbol of cd-amplifier



Building the Common Drain Amplifier Test Design:

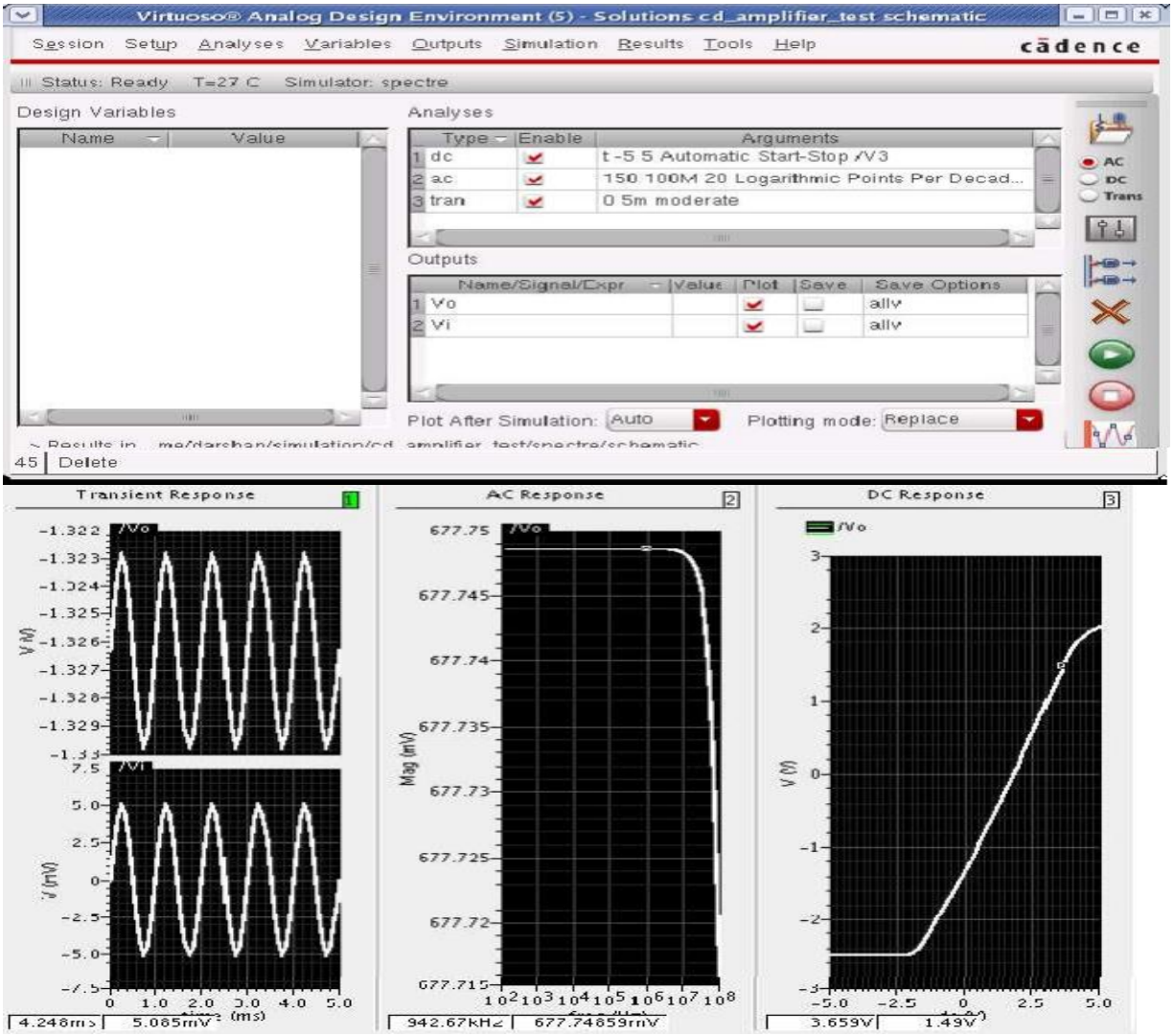
Using the component list and Properties/Comments in the table, build the cd-amplifier_test schematic as shown below.

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



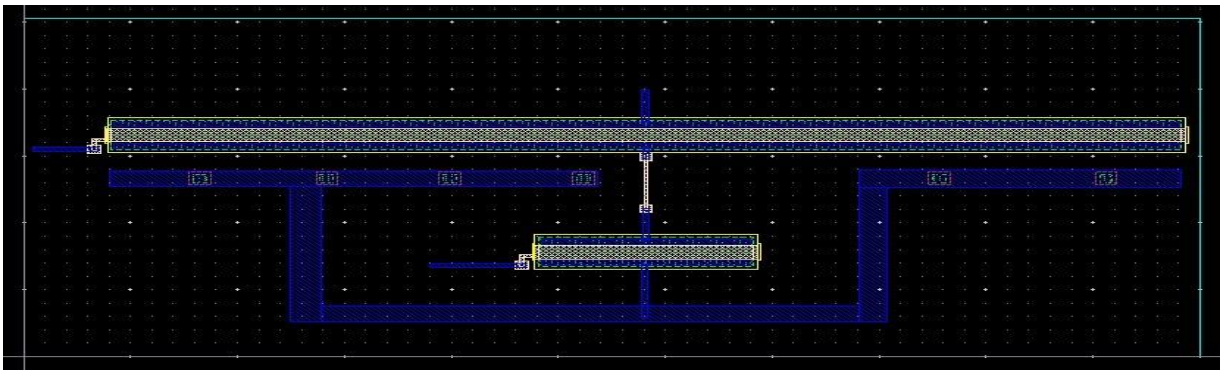
Analog Simulation with Spectre

Use the techniques learned in the Lab1 and Lab2 to complete the simulation of cd_amplifier, ADE window and waveform should look like below.



Creating a layout view of Common Drain Amplifier

Use the techniques learned in the Lab1 and Lab2 to complete the layout of cd_amplifier. Complete the DRC, LVS check using the assura tool. Extract RC parasites for back annotation and Re-simulation.



Experiment 4: SINGLE STAGE DIFFERENTIAL AMPLIFIER.

Aim: Design the following circuits with given specifications for single stage differential amplifier completing the design flow mentioned below:

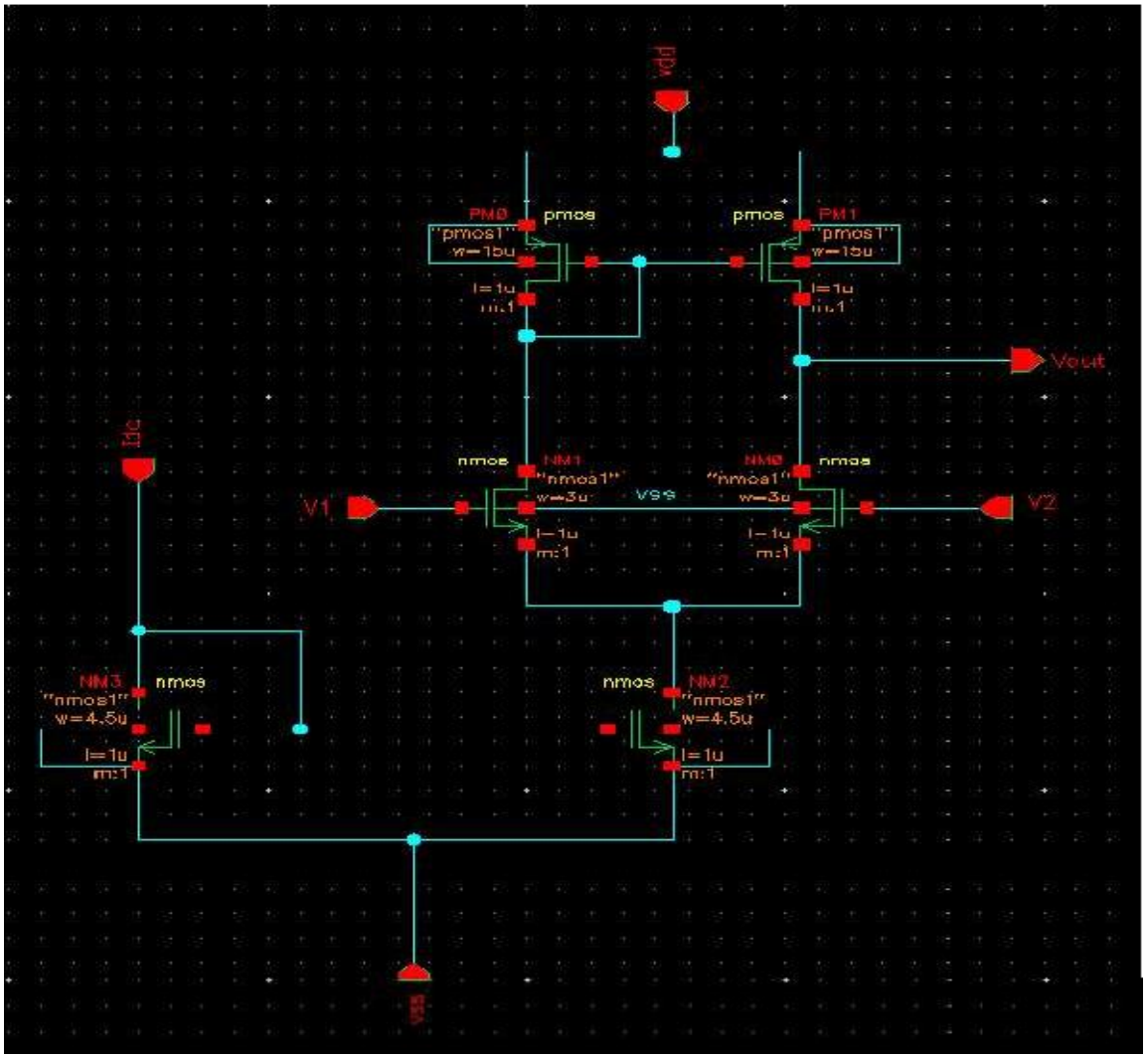
d) Draw th schematic and verify the following
   - DC Analysis
   - AC Analysis
   - Transient Analysis

e) Draw the Layout and verify the DRC,ERC

f) Check for LVS

g) Extract RC and back annotate the same and verify the Design.

Schematic Capture

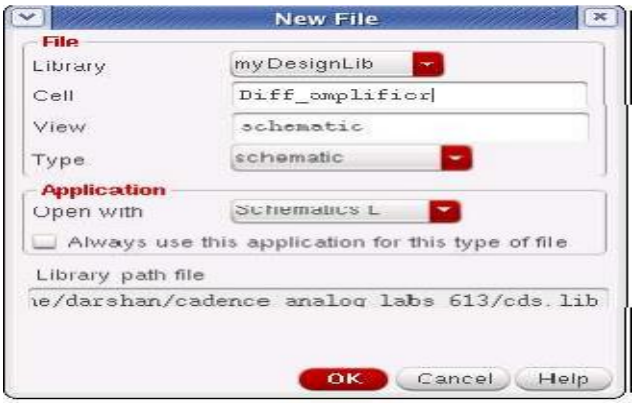


Schematic Entry:

Creating a Schematic cellview:

Open a new schematic window in the myDesignLiblibrary and build the Differential_Amplifier design.

1. In the CIW or Library manager, execute File – New – Cellview. Set up the Create New file form as follows:



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

Adding Components to schematic

1. In the Differential Amplifier schematic window, execute Create— Instance to display the Add Instance form.
2. Click on the Browse button. This opens up a Library browser from which you can select components and the Symbol view. You will update the Library Name, Cell Name, and the property values given in the table on the next page as you place each component.
3. After you complete the Add Instance form, move your cursor to the schematic window and click left to place a component.

This is a table of components for building the Differential Amplifier schematic.

Library name	Cell Name	Properties/Comment
gpdk180	nmos	Model Name = nmos1 (NM0, NM1) ; W= 3u ; L= 1u
gpdk180	nmos	Model Name =nmos1 (NM2, NM3) ; W= 4.5u ; L= 1u
gpdk180	pmos	Model Name =pmos1 (PM0, PM1); W= 15u ; L= 1u

4. After entering components, click Cancel in the Add Instance form or press Esc with your cursor in the schematic window.

Adding pins to Schematic:

Use Create – Pin or the menu icon to place the pins on the schematic window.
1. Click the Pin fixed menu icon in the schematic window.
You can also execute Create – Pin or press p. The Add pin form appears.
2. Type the following in the Add pin form in the exact order leaving space between the pin names.

Pin Names	Direction
Idc,V1,V2	Input
Vout	Output
vdd, vss,	InputOutput

Make sure that the direction field is set to input/ouput/inputoutputwhen placing the input/output/inoutpins respectively and the Usage field is set to schematic.
3. Select Cancel from the Add pin form after placing the pins. In the schematic window, execute View— Fit or press the f bindkey.

Adding Wires to a Schematic:

Add wires to connect components and pins in the design.
1. Click the Wire (narrow) icon in the schematic window.
You can also press the w key, or execute Create - Wire (narrow).
2. Complete the wiring as shown in figure and when done wiring press ESC key in the schematic window to cancel wiring.

Saving the Design:

1. Click the Check and Save icon in the schematic editor window.
2. Observe the CIW output area for any errors.

Symbol Creation

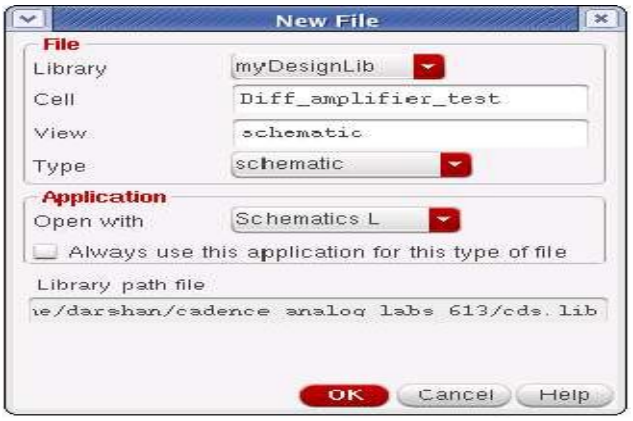
1. In the Differential Amplifier schematic window, execute
Create — Cellview— From Cellview.
The Cellview from Cellviewform appears. With the Edit Options function active, you can control the appearance of the symbol to generate.
2. Verify that the From View Name field is set to schematic, and the To View Name field is set to symbol, with the Tool/Data Type set as SchematicSymbol.
3. Click OK in the Cellview from Cellview form. The Symbol Generation Form appears.
4. Modify the Pin Specifications as in the below symbol.
5. Click OK in the Symbol Generation Options form.
6. A new window displays an automatically created Differential Amplifier symbol.
7. Modifying automatically generated symbol so that it looks like below Differential Amplifier symbol.
8. Execute Create— Selection Box. In the Add Selection Box form, click Automatic. A new red selection box is automatically added.

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

Building the Diff_amplifier_test Design

Creating the Differential Amplifier Test Cellview:

1. In the CIW or Library Manager, execute File— New— Cellview.
2. Set up the Create New File form as follows:



3. Click OK when done. A blank schematic window for the Diff_ amplifier_test design appears.

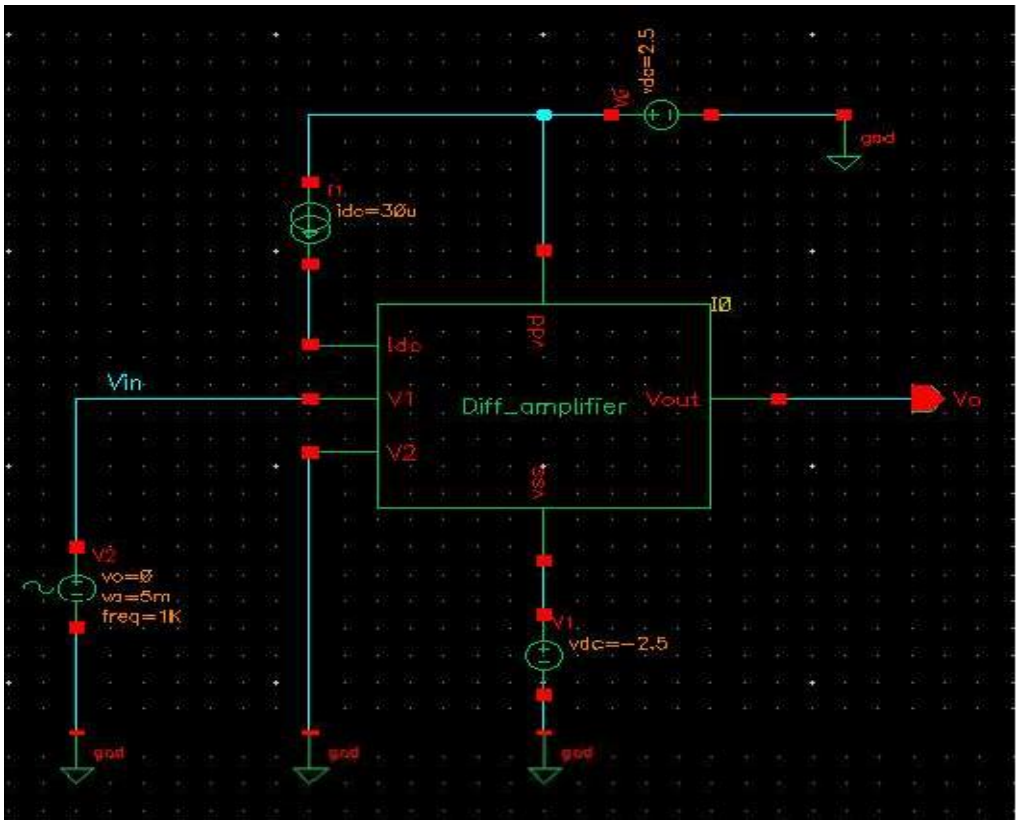
Building the Diff_amplifier_testCircuit:

1. Using the component list and Properties/Comments in this table, build the Diff_amplifier_test schematic.

Library name	Cellview name	Properties/Comments
myDesignLib	Diff_amplifier	Symbol
analogLib	vsin	Define specification as AC Magnitude= 1; Amplitude= 5m; Frequency= 1K
analogLib	vdd, vss, gnd	Vdd=2.5 ; Vss= -2.5
analogLib	Idc	Dc current = 30u

Note: Remember to set the values for VDD and VSS. Otherwise your circuit will have no power.

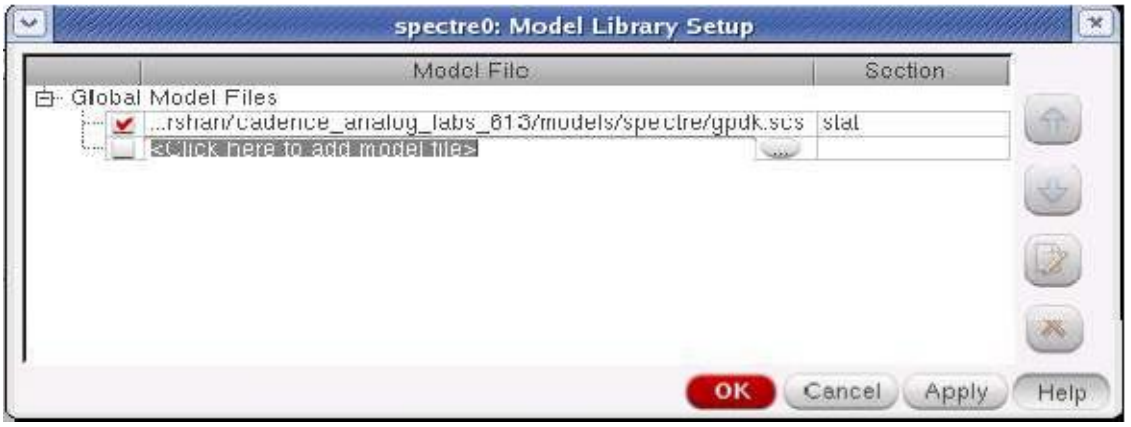
3. Click the Wire (narrow) icon and wire your schematic.
Tip: You can also press the w key, or execute Create— Wire (narrow).
4. Click on the Check and save icon to save the design.
5. The schematic should look like this.



6. Leave your Diff_amplifier_test schematic window open for the next section.

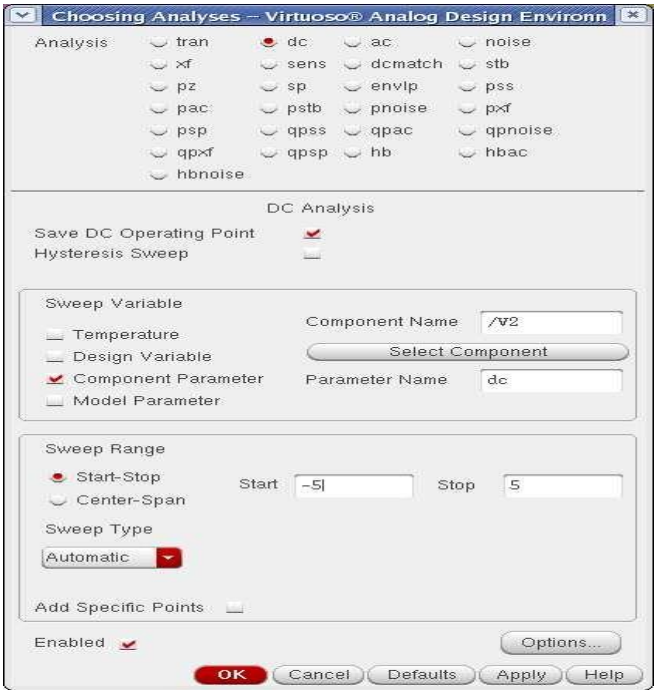
Analog Simulation with Spectre:
In this section, we will run the simulation for Differential Amplifier and plot the transient, DC and AC characteristics.
Starting the Simulation Environment
1. In the Diff_amplifier_test schematic window, execute Launch – ADE L. The Analog Design Environment simulation window appears.
Choosing a Simulator:
1. In the simulation window or ADE, execute
Setup— Simulator/Directory/Host.
2. In the Choosing Simulator form, set the Simulator field to spectre
(Not spectreS) and click OK.

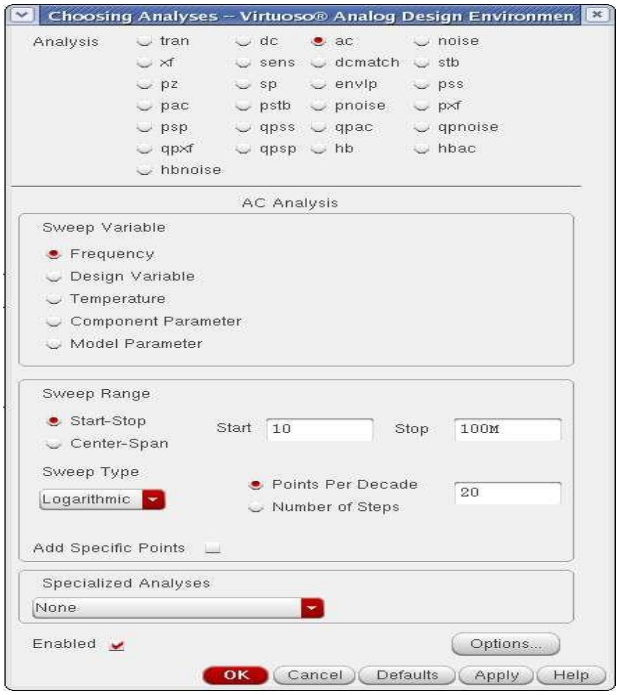
Setting the Model Libraries:
1. Click Setup - Model Libraries.
Note: Step 2 should be executed only if the model file not loaded by default.
2. In the Model Library Setup form, click Browse and find the gpdk180.scs file
in the ./models/spectredirectory.
Select stat in Section field, click Add and click OK.



Choosing Analyses:

1. In the Simulation window, click the Choose - Analyses icon.
You can also execute Analyses - Choose.
The Choosing Analysis form appears. This is a dynamic form, the bottom of the form Changes based on the selection above.
2. To setup for transient analysis
a. In the Analysis section select tran
b. Set the stop time as 5m
c. Click at the moderate or Enabledbutton at the bottom, and then click Apply.
3. To set up for DC Analyses:
a. In the Analyses section, select dc.
b. In the DC Analyses section, turn on Save DC Operating Point.
c. Turn on the Component Parameter
d. Double click the Select Component, Which takes you to the schematic window.
e. Select input signal Vsinfor dc analysis.
f. In the analysis form, select start and stop voltages as -5 to 5 respectively.
g. Check the enable button and then click Apply.

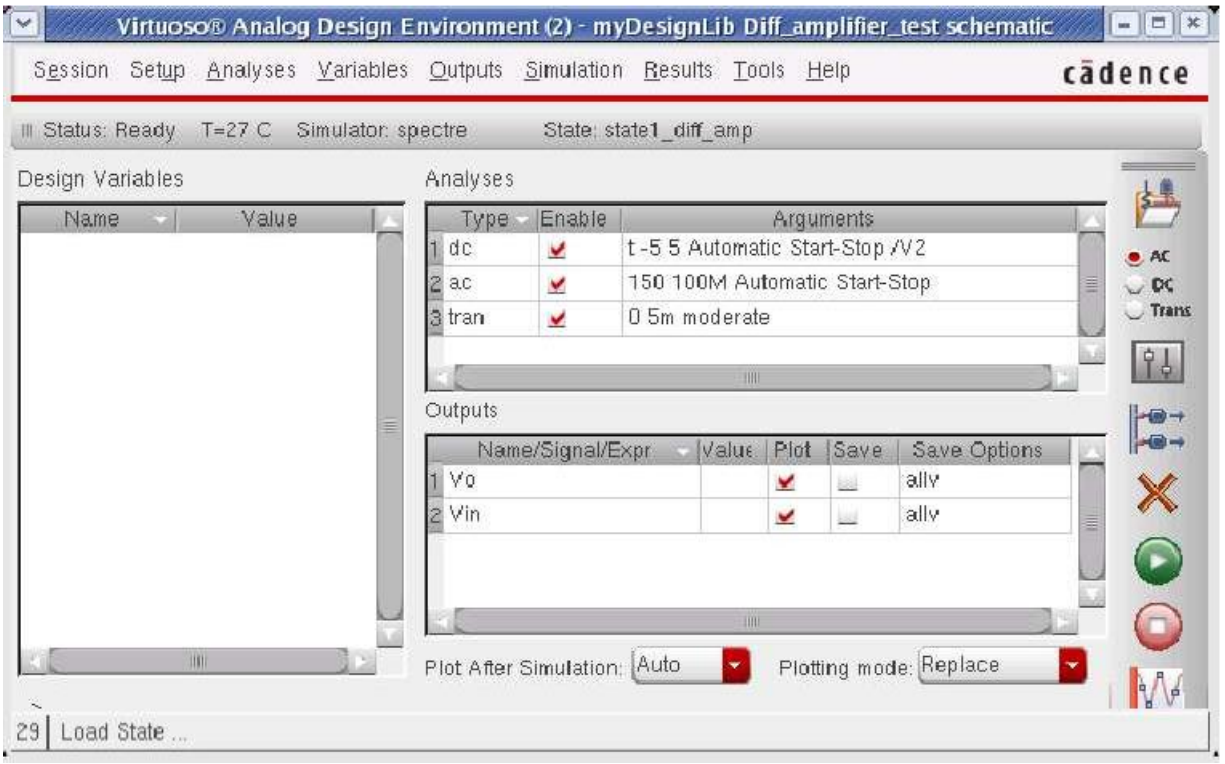




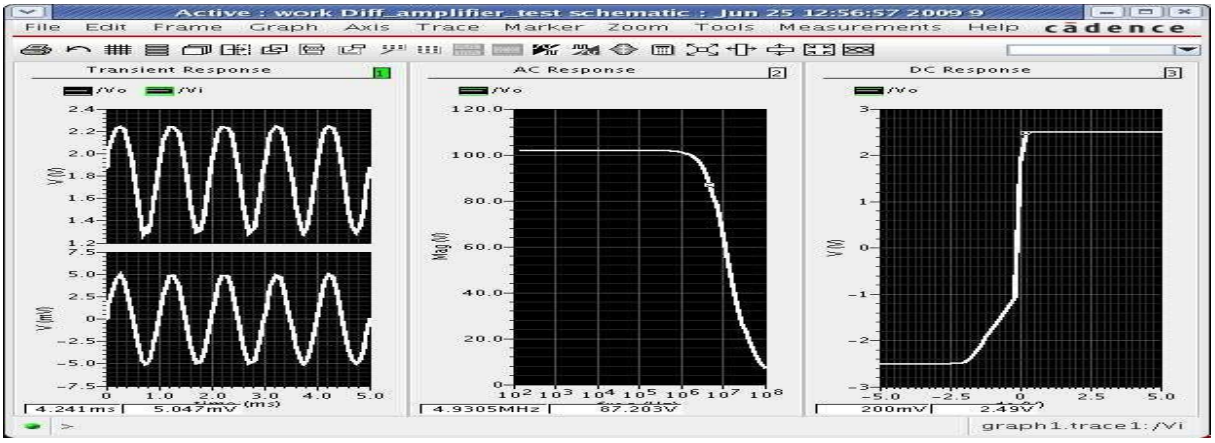
4. To set up for AC Analyses form is shown in the previous page. a. In the Analyses section, select ac.

b. In the AC Analyses section, turn on Frequency.
c. In the Sweep Range section select start and stop frequencies as 150 to 100M
d. Select Points per Decade as 20.
e. Check the enable button and then click Apply.
5. Click OK in the Choosing Analyses Form.

Selecting Outputs for Plotting:
Select the nodes to plot when simulation is finished.
1. Execute Outputs – To be plotted – Select on Schematic in the simulation window.
2. Follow the prompt at the bottom of the schematic window, Click on output net Vo, input net Vin of the Diff_amplifier. Press ESC with the cursor in the schematic after selecting node.

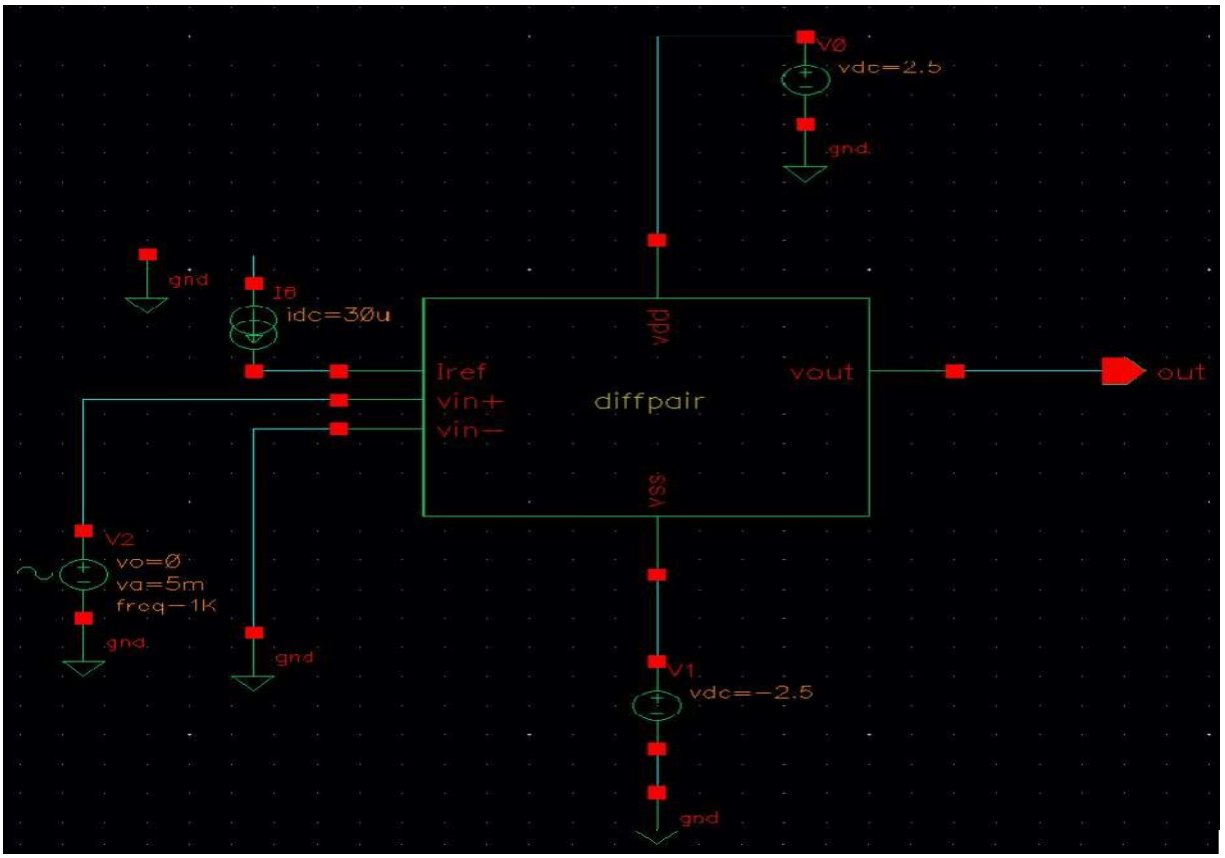


Running the Simulation:
1. Execute Simulation – Netlist and Run in the simulation window to start the simulation, this will create the netlist as well as run the simulation.
2. When simulation finishes, the Transient, DC and AC plots automatically will be popped up along with netlist.

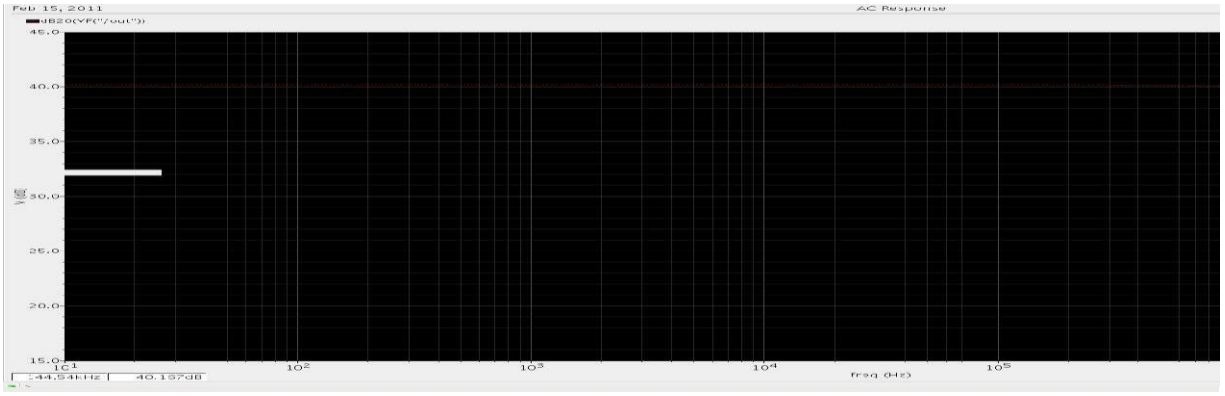


To Calculate the gain of Differential pair:

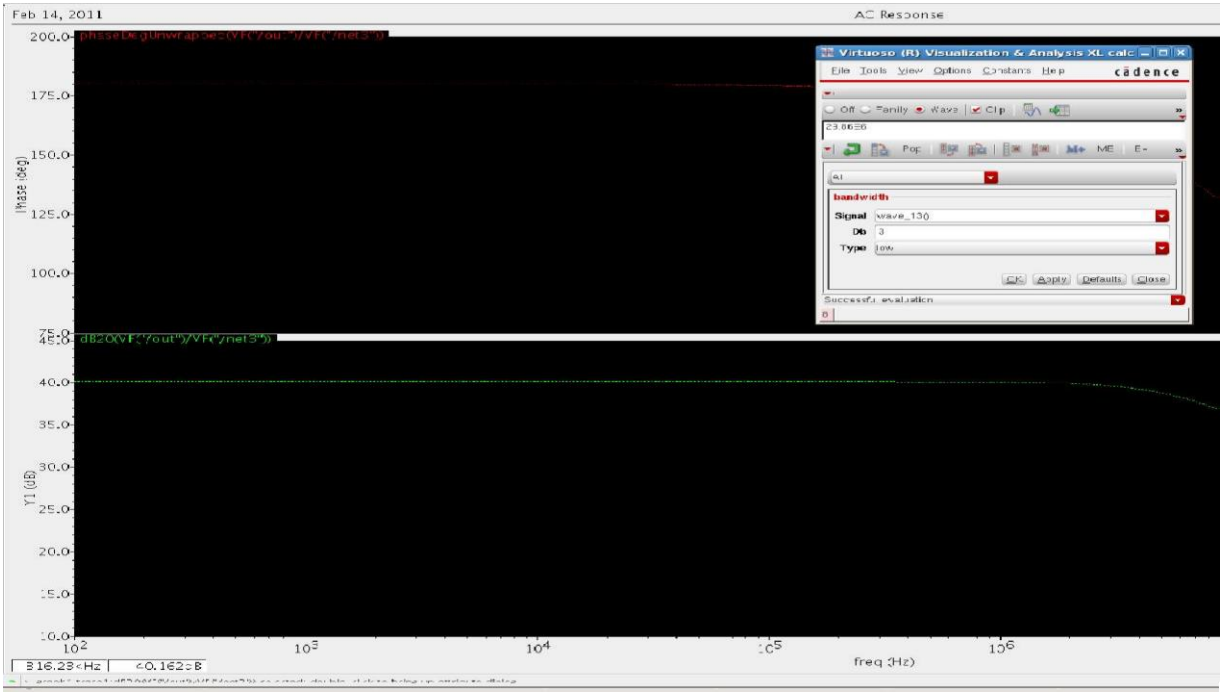
Configure the Differential pair schematic as shown below –



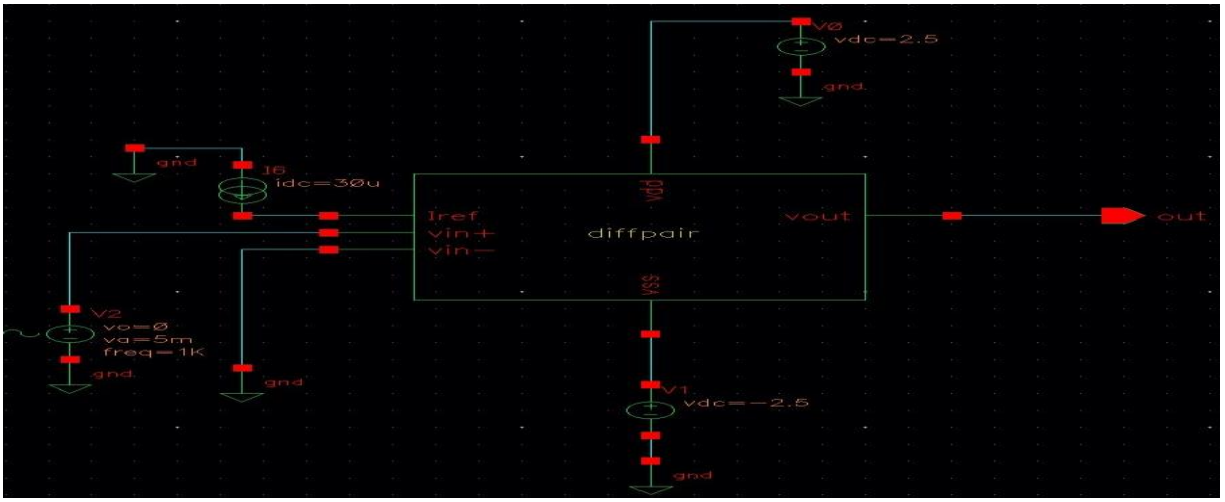
Now, open the ADE L, from LAUNCH ADE L, choose the analysis set the ac response and run the simulation, from Simulation Run. Next go to Results Direct plot select AC dB20 and output from the schematic and press escape. The following waveform appears as shown below –



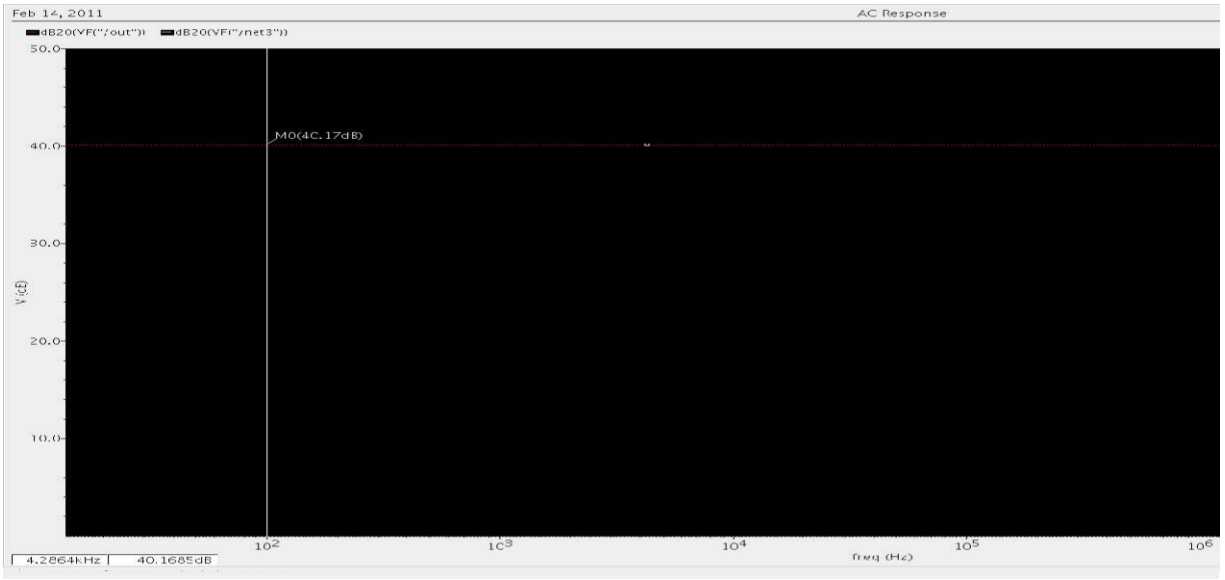
To Calculate the BW of the Differential pair:
Open the calculator and select the bandwidth option, select the waveform of the gain in dB and press Evaluate the buffer –



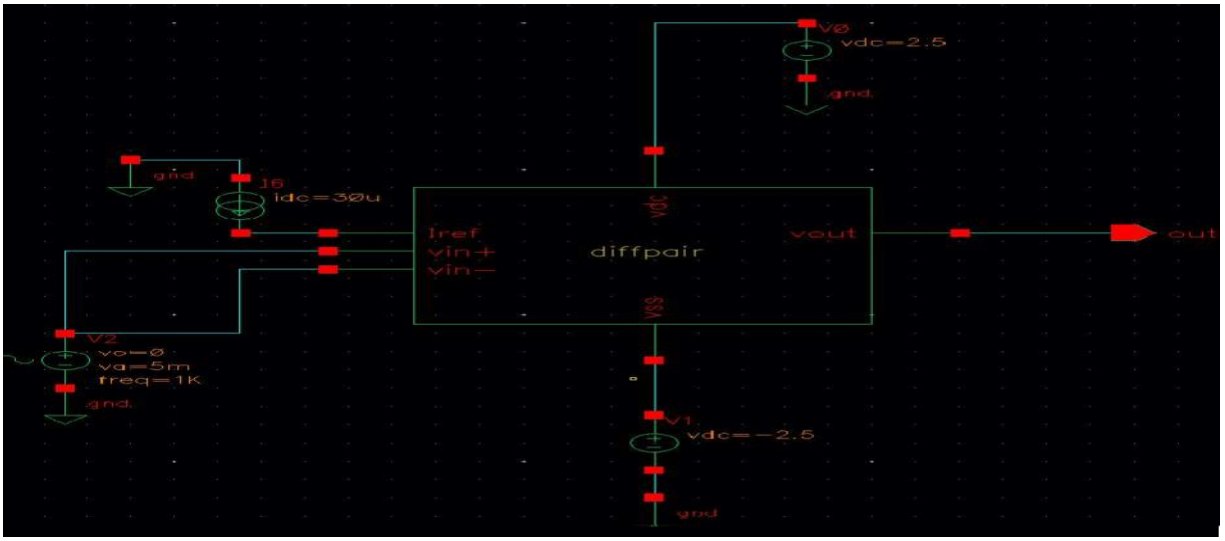
To Calculate the CMRR of the Differential pair:
Configure the Differential pair schematic to calculate the differential gain as shown below –



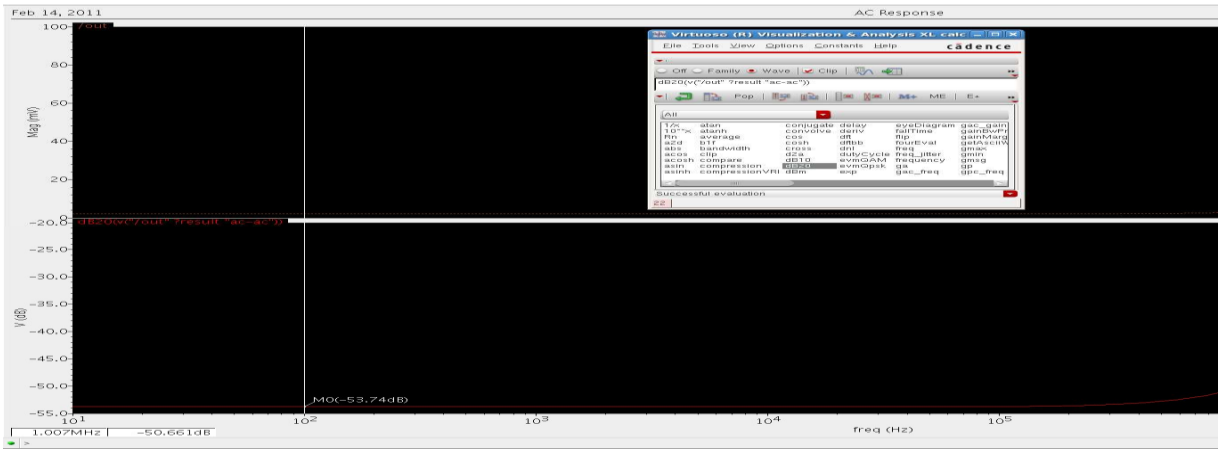
In the ADE L, plot the ac response with gain in dB. Measure the gain at 100hz and at 100Mhz,note down the value of the gain in dB, as shown below –



Configure the Differential pair schematic to calculate the common-mode gain as shown below –



In the ADE L, plot the ac response with gain in dB. Measure the gain at 100hz and at 100Mhz, note down the value of the gain in dB, as shown below –



Calculate the CMRR = || , add the gains in dB i.e., Ad – (-Ac). For the output impedance note down the output resistance of the pmos and nmos transistors at the ouput side, and use the necessary equation like r01|| r02 .

Saving the Simulator State:

We can save the simulator state, which stores information such as model library file, outputs, analysis, variable etc. This information restores the simulation environment without having to type in all of setting again.

1. In the Simulation window, execute Session – Save State.

The Saving State form appears.

2. Set the Save as field to state1_diff_amp and make sure all options are selected under what to save field. Click OK in the saving state form. The Simulator state is saved.

Creating a Layout View of Diff_ Amplifier:

1. From the Diff_amplifier schematic window menu execute

Launch – Layout XL. A Startup Option form appears.

2. Select Create New option. This gives a New Cell View Form
3. Check the Cellname(Diff_amplifier), Viewname(layout).
4. Click OK from the New Cellview form.

LSW and a blank layout window appear along with schematic window.

Adding Components to Layout:

1. Execute Connectivity – Generate – All from Source or click the icon in the layout editor window, Generate Layout form appears. Click OK which imports the schematic components in to the Layout window automatically.
2. Re arrange the components with in PR-Boundary as shown in the next page.
3. To rotate a component, Select the component and execute Edit –Properties. Now select the degree of rotation from the property edit form.

4. To Move a component, Select the component and execute Edit -Move command.

Making interconnection:

1. Execute Connectivity –Nets – Show/Hide selected Incomplete Nets or click the icon in the Layout Menu.
2. Move the mouse pointer over the device and click LMB to get the connectivity information, which shows the guide lines (or flight lines) for the inter connections of the components.
3. From the layout window execute Create – Shape – Path or Create – Shape – Rectangle (for vdd and gnd bar) and select the appropriate Layers from the LSW window and Vias for making the inter connections

Creating Contacts/Vias

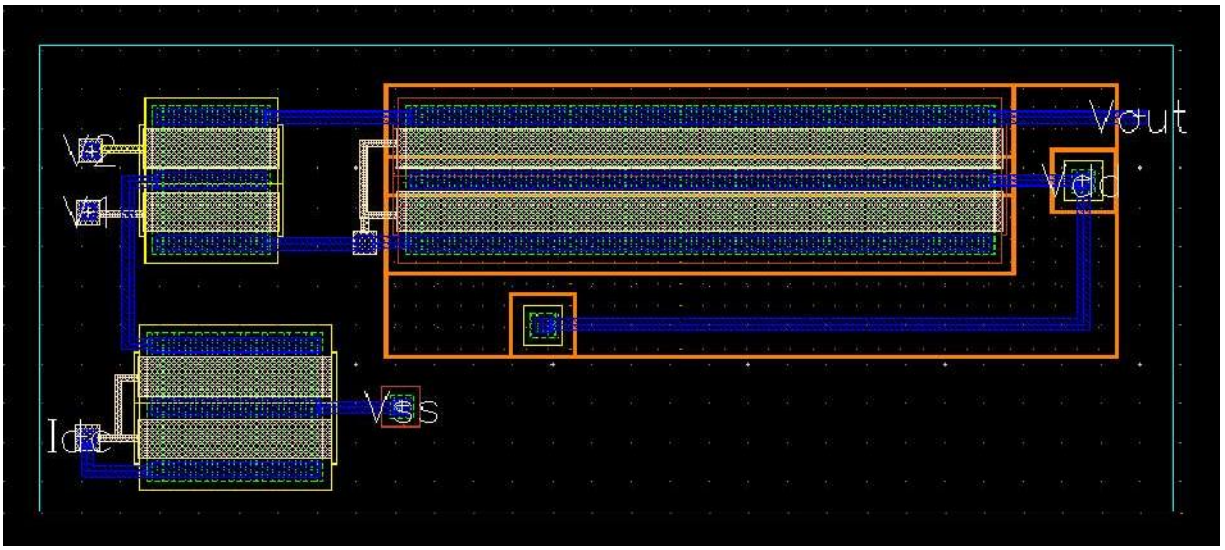
You will use the contacts or vias to make connections between two different layers.

1. Execute Create — Viaor select  command to place different Contacts, as given in below table.

Connection	Contact Type
For Metal1- Poly Connection	Metal1-Poly
For Metal1- Psubstrate Connection	Metal1-Psub
For Metal1- Nwell Connection	Metal1-Nwell

Saving the design

1. Save your design by selecting File — Saveor click  to save the layout and layout should appear as below



Physical Verification:

Assura DRC:

Running a DRC:

1. Open the Differential_Amplifier layout form the CIW or library manger if you have closed that.Pressshift – f in the layout window to display all the levels.
2. Select Assura - Run DRC from layout window.

The DRC form appears. The Library and Cellname are taken from the current design window, but rule file may be missing. Select the Technology as gpdk180. This automatically loads the rule file.

3. Click OK to start DRC.
4. A Progress form will appears. You can click on the watch log file to see the log file.
5. When DRC finishes, a dialog box appears asking you if you want to view your DRC results, and then click Yes to view the results of this run.

6. If there any DRC error exists in the design View Layer Window (VLW) and Error Layer Window (ELW) appears. Also the errors highlight in the design itself.
7. Click View – Summary in the ELW to find the details of errors.
8. You can refer to rule file also for more information, correct all the DRC errors and Re – run the DRC.
9. If there are no errors in the layout then a dialog box appears with No DRC errors foundwritten in it, click on close to terminate the DRC run.

ASSURA LVS

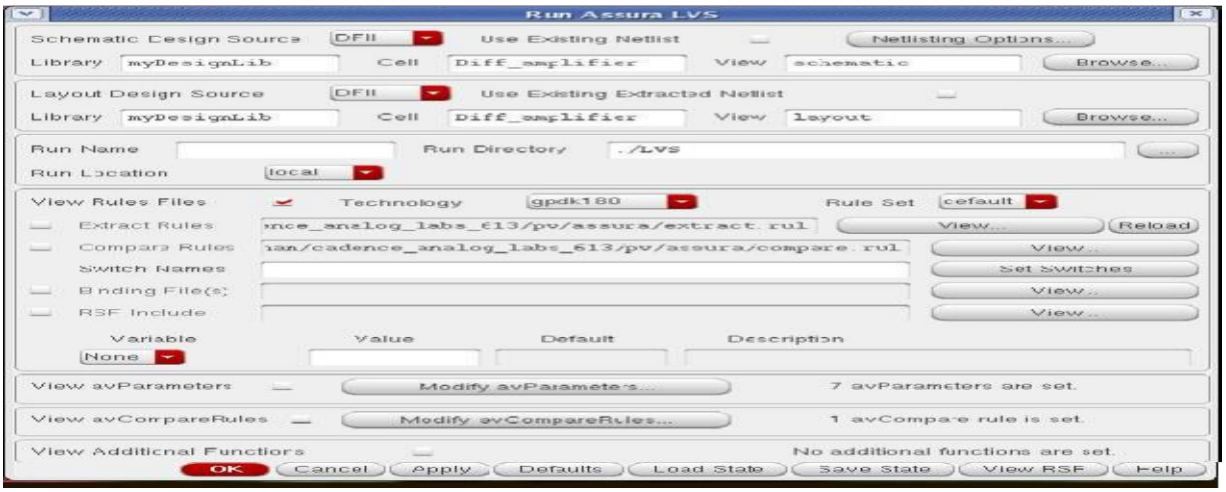
In this section we will perform the LVS check that will compare the schematic netlist and the layout netlist.

Running LVS:

1. Select Assura – Run LVS from the layout window.

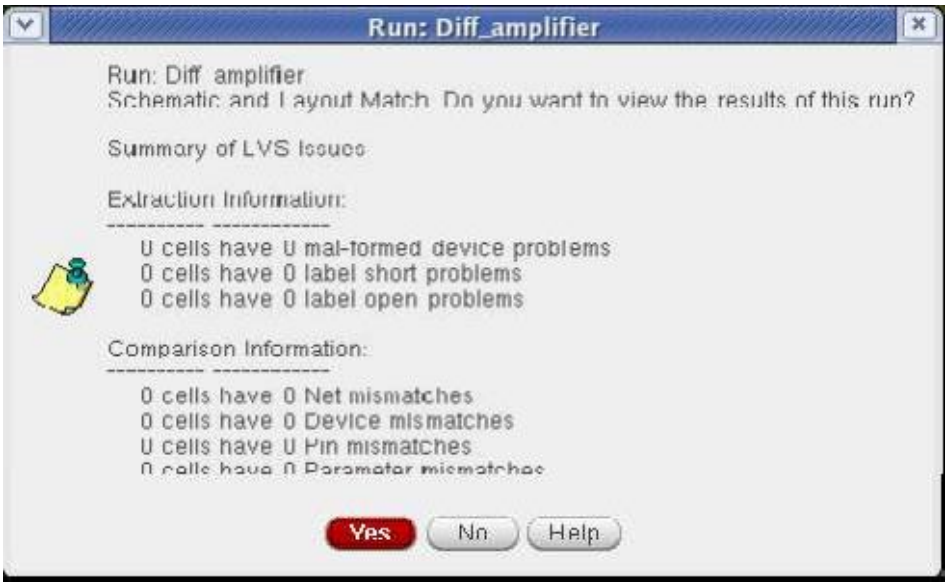
The Assura Run LVS form appears. The layout name is already in the form. Assura fills in the layout name from the cellview in the layout window.

2. Verify the following in the Run Assura LVS form.

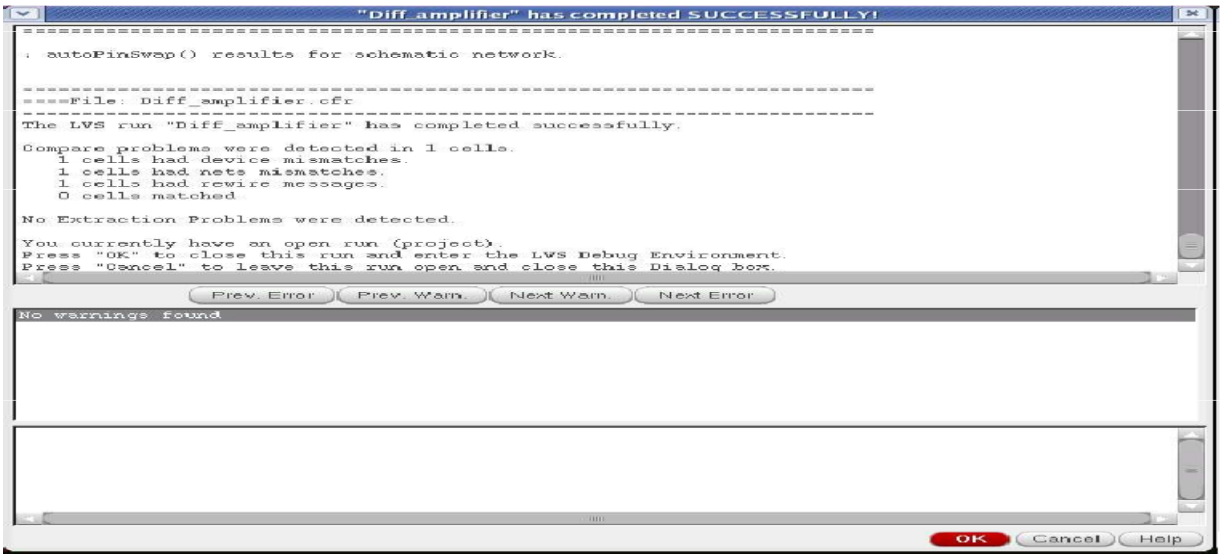


3. The LVS begins and a Progress form appears.

4. If the schematic and layout matches completely, you will get the form displaying Schematic and Layout Match.



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

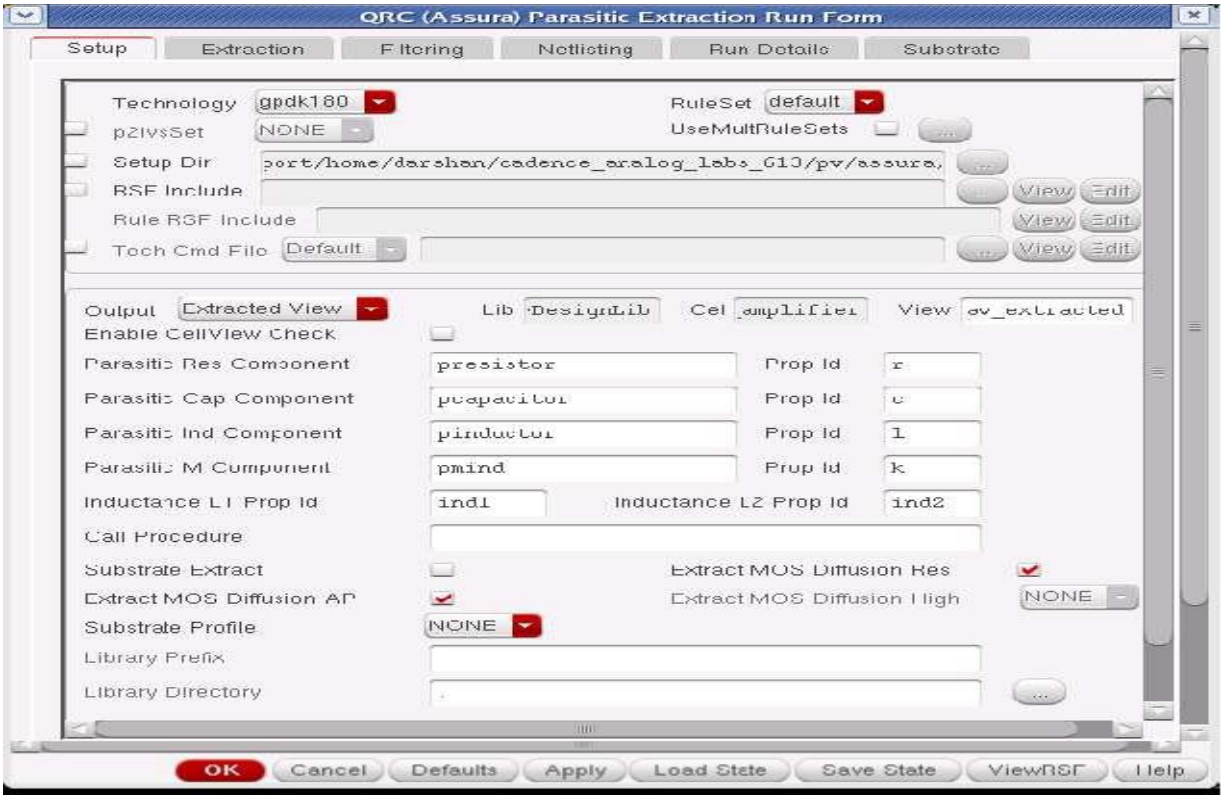


6. Click Yes in the form.LVS debug form appears, and you are directed into LVS debug environment.
7. In the LVS debug form you can find the details of mismatches and you need to correct all those mismatches and Re – run the LVS till you will be able to match the schematic with layout

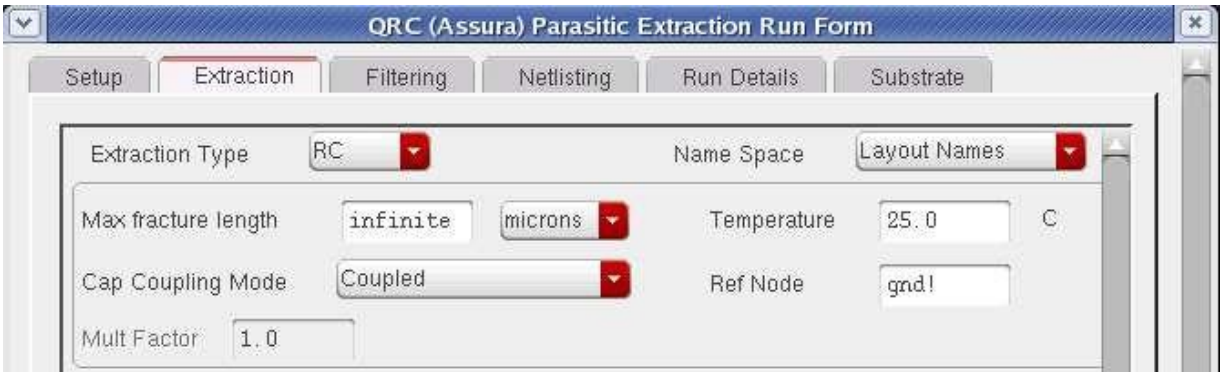
Assura RCX:

In this section we will extract the RC values from the layout and perform analog circuit simulation on the designs extracted with RCX.
Before using RCX to extract parasitic devices for simulation, the layout should match with schematic completely to ensure that all parasites will be backannoted to the correct schematic nets.

Running RCX:
1. From the layout window execute Assura – Run RCX.
2. Change the following in the Assura parasitic extraction form. Select output type under Setup tab of the form.



3. In the Extraction tab of the form, choose Extraction type, Cap Coupling Mode and specify the Reference node for extraction.



4. In the Filtering tab of the form, Enter Power Nets as vdd!,vss! andEnter Ground Nets as gnd!

5. Click OK in the Assura parasitic extraction form when done.
The RCX progress form appears, in the progress form click Watch log file to see the output log file.
5. When RCX completes, a dialog box appears, informs you that Assura RCX run completed successfully.

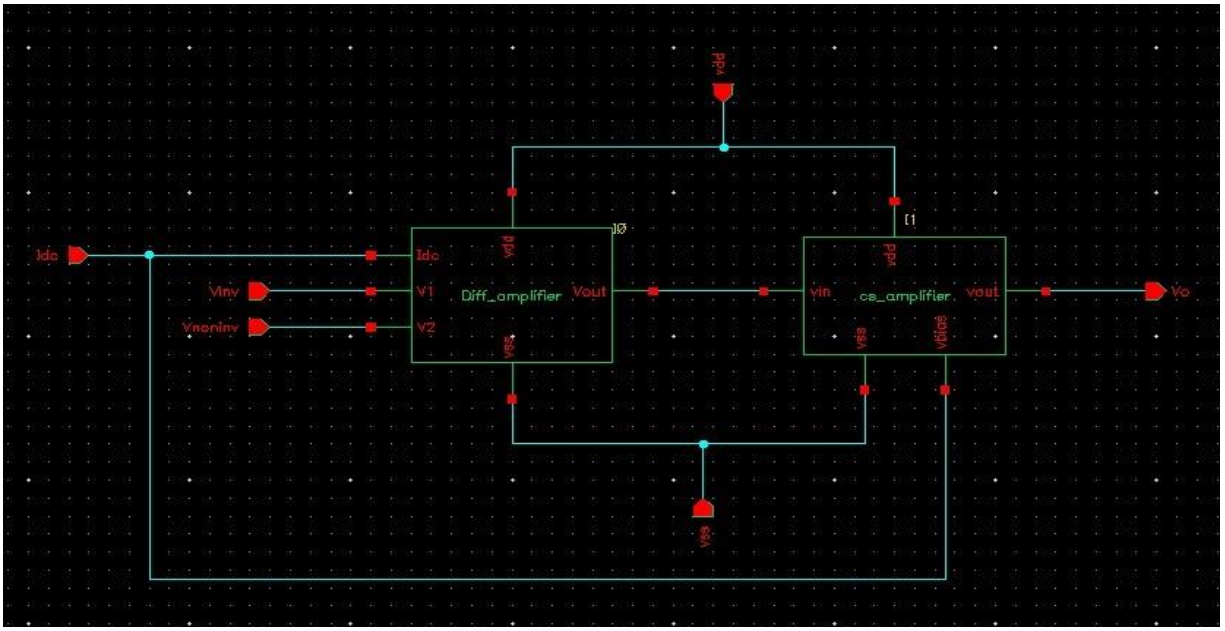
6. You can open the av_extractedview from the library manager and view the parasitic.

Experiment 5: OPERATIONAL AMPLIFIER.

Aim: Design an op- amp with given specification using given differential amplifier Common source and Common Drain amplifier in library and completing the design flow mentioned below:

a) Draw the schematic and verify the following
   - DC Analysis
   - AC Analysis
   - Transient Analysis
b) Draw the Layout and verify the DRC,ERC
c) Check for LVS
d) Extract RC and back annotate the same and verify the Design.

Schematic Capture:



Schematic Entry:

Use the techniques learned in the Lab1 and Lab2 to complete the schematic of Operational Amplifier. This is a table of components for building the Operational Amplifier schematic.

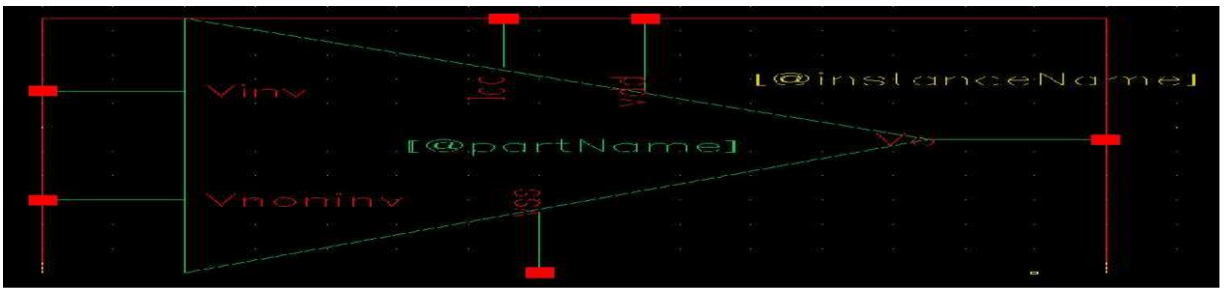
Library name	Cell Name	Properties/Comments
myDesignLib	Diff_amplifier	Symbol
myDesignLib	cs_amplifier	Symbol

Type the following in the ADD pin form in the exact order leaving space between the pin names.

Pin Names	Direction
Idc,Vinv,Vnoninv	Input
Vo	Output
vdd, vss	Input

Symbol Creation:

Use the techniques learned in the Lab1 and Lab2 to complete the symbol of op-amp.

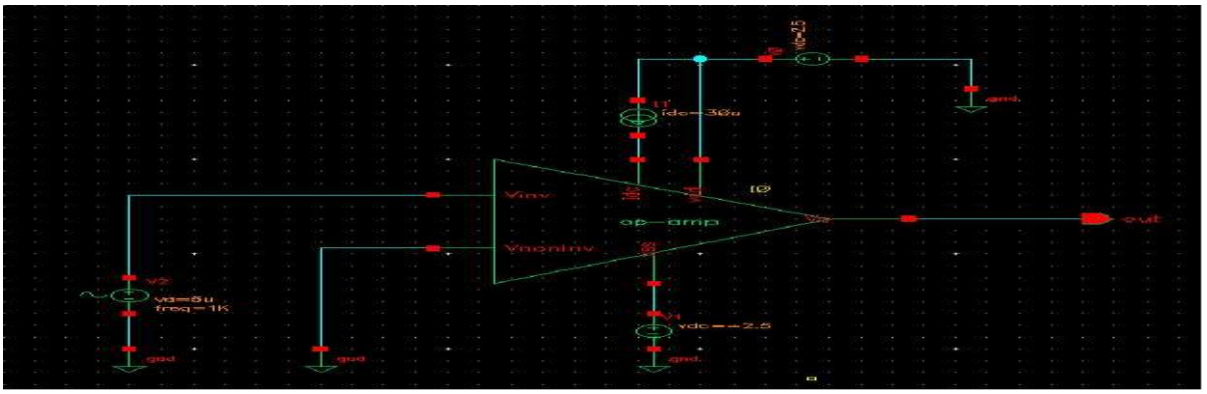


Building the Operational Amplifier Test Design:

Using the component list and Properties/Comments in the table, build the op-amp_test schematic as shown below.

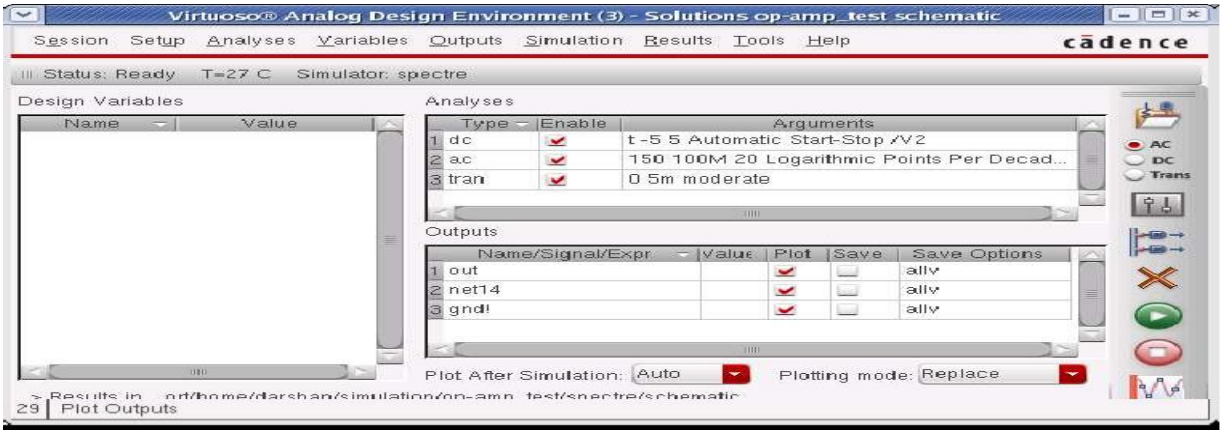
Library name	Cellview name	Properties/Comments
myDesignLib	op-amp	Symbol
analogLib	vsin	Define pulse specification as AC Magnitude= 1; DC Voltage= 0; Offset Voltage= 0; Amplitude= 5m; Frequency= 1K
analogLib	vdc, gnd	vdd=2.5 ; vss= -2.5
analogLib	Idc	Dc current = 30u

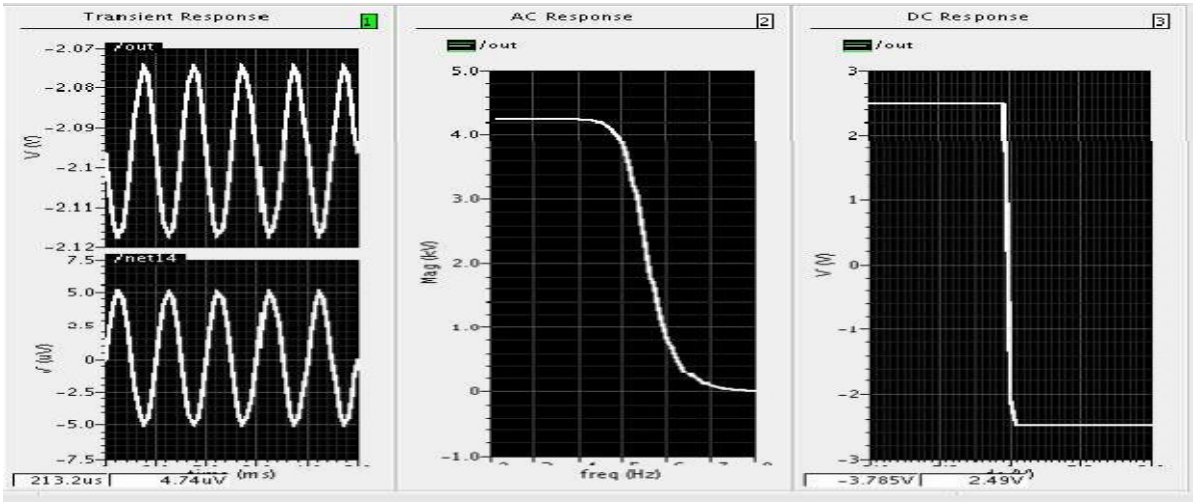
Note: Remember to set the values for vddand vss. Otherwise your circuit will have no power.



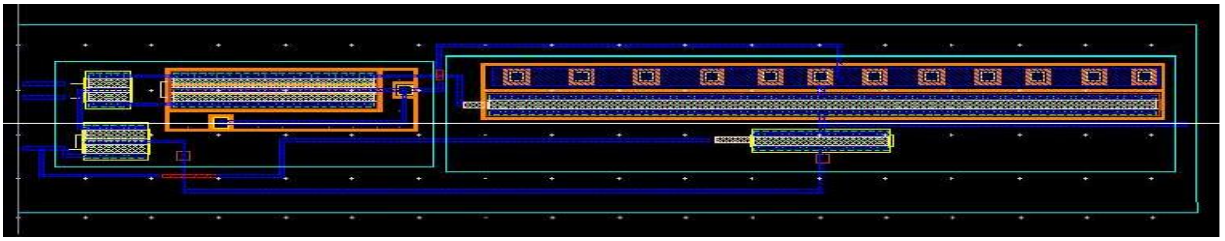
Analog Simulation with Spectre:

Use the techniques learned in the Lab1 and Lab2 to complete the simulation of op-amp, ADE window and waveform should look like below.





Creating a layout view of Operational Amplifier: Use the techniques learned in the Lab1 and Lab2 to complete the layout of op-amp. Complete the DRC, LVS check using the assura tool. Extract RC parasites for back annotation and Re-simulation.



B: DIGITAL VLSI DESIGN

ASIC-Digital Design Flow

Experiment 1: Inverter and Tristate Buffer

Inverter

Data flow model

```
module in1(a, b);
        input a; output b;
        assign b=~a;
endmodule
```

Behavioural model

```
module in1(a, b);
        input a;
        output reg b;
always @(a)
begin
        b=~a;
end
endmodule
```

TestBench

```
        module iin_v; reg
        a; // Inputs wire
        b; // Outputs
        // Instantiate the Unit Under Test (UUT)
        in1 uut (.a(a),.b(b) );
        initial begin
        a = 0;
        #100; // Wait 100 ns for global reset to finish
        // Add stimulus here
        a = 1; #100; // Wait 100 ns for global reset to finish
        end
        endmodule
```

Tristate Buffer

```
module  tristate_buffer(input_x, enable, output_x);
input  input_x;
input  enable;
output output_x;
assign  output_x = enable?  input_x : 'bz;
endmodule
```

Experiment 2: TRANSMISSION GATE

```
module transmission_gate(A,IN,OUT);
        input A,IN;
        output OUT;
        wire Abar;
        assign Abar=~A;
        pmos(OUT,Abar,IN);
        nmos(OUT,A,IN);
endmodule
```

TestBench

```
module tg_tb_v;
reg A; reg IN; // Inputs
wire OUT; // Outputs
transmission_gate uut ( .A(A), .IN(IN), .OUT(OUT) );
initial begin
A = 0; IN = 0;#100;
A = 0; IN = 1;#100;
A = 1; IN = 0;#100;
A = 1; IN = 1;#100;
end endmodule
```

Data flow model

```
module gat(c,d,a,o,na,no,x);
        input c,d ;
        output a,o,x,no,na;
        assign o=(c|d); //o -> or gate output assign
        no=~(c|d); // no-> nor gate output assign
        a=(c&d); // a-> and gate output assign
        na=~(c&d); // na-> nand gate output assign
        x=(c^d); // x-> xor gate output
endmodule
```

TestBench

```
module basi_v;
reg c, d;
wire a, o, na, no, x; // Outputs
gat uut (.c(c), .d(d), .a(a), .o(o), .na(na), .no(no), .x(x) );
initial begin
c = 0; d = 0;#100;
c = 0; d = 1;#100;
c = 1; d = 0;#100;
c = 1; d = 1;#100;
end
endmodule
```

Experiment 3: Ripple Carry Adder

Structural Model: Half Adder

```
module half_adder(
 output S,C,
input A,B );
xor(S,A,B);
and(C,A,B);
endmodule
```

Structural Model : Full Adder

```
module full_adder(
   output S,Cout,
   input A,B,Ci );
wire s1,c1,c2;
half_adder HA1(s1,c1,A,B);
half_adder HA2(S,c2,s1,Cin);
or OG1(Cout,c1,c2);
endmodule
```

Structural Model : 4 Bit Ripple Carry Adder

```
module ripple_adder_4bit(
   output [3:0] Sum,
   output Cout,
   input [3:0] A,B,
   input Cin );
wire c1,c2,c3;
full_adder FA1(Sum[0],c1,A[0],B[0],Cin),
FA2(Sum[1],c2,A[1],B[1],c1),
FA3(Sum[2],c3,A[2],B[2],c2),
FA4(Sum[3],Cout,A[3],B[3],c3);
endmodule
```

TestBench

```
module test_ripple_adder_4bit;
reg [3:0] A;
reg [3:0] B;
reg Cin;
wire [3:0] Sum;
wire Cout;
ripple_adder_4bit uut ( .Sum(Sum), .Cout(Cout), .A(A), .B(B), .Cin(Cin) );
initial begin
A = 0; B = 0;nCin = 0; #100;
A=4'b0001;B=4'b0000;Cin=1'b0; #10;
A=4'b1010;B=4'b0011;Cin=1'b0; #10;
A=4'b1101;B=4'b1010;Cin=1'b1;
end
initial begin
$monitor("time=",$time,, "A=%b B=%b Cin=%b : Sum=%b Cout=%b",A,B,Cin,Sum,Cout);
end
endmodule
```

Experiment 4: Carry Look Ahead Adder

```
module cpu_wb_cla_adder (in1, in2, carry_in, sum, carry_out);
parameterDATA_WID = 32;
input [DATA_WID - 1:0] in1;
input [DATA_WID - 1:0] in2;
inputcarry_in;
output [DATA_WID - 1:0] sum;
outputcarry_out;
//assign {carry_out, sum} = in1 + in2 + carry_in;
wire [DATA_WID - 1:0] gen;
wire [DATA_WID - 1:0] pro;
wire [DATA_WID:0] carry_tmp;
genvar j, i;
generate
//assume carry_tmp in is zero
assigncarry_tmp[0] = carry_in;
//carry generator
for(j = 0; j <DATA_WID; j = j + 1) begin: carry_generator
assign gen[j] = in1[j] & in2[j];
assign pro[j] = in1[j] | in2[j];
assigncarry_tmp[j+1] = gen[j] | pro[j] &carry_tmp[j];
end
//carry out
assigncarry_out = carry_tmp[DATA_WID];
//calculate sum
//assign sum[0] = in1[0] ^ in2 ^ carry_in;
for(i = 0; i <DATA_WID; i = i+1) begin: sum_without_carry
assign sum[i] = in1[i] ^ in2[i] ^ carry_tmp[i];
end
endgenerate
endmodule
```

TestBench

```
regcarry_in; // To cla1 of cla_adder.v
reg [DATA_WID-1:0] in1; // To cla1 of cla_adder.v
reg [DATA_WID-1:0] in2; // To cla1 of cla_adder.v
wirecarry_out; // From cla1 of cla_adder.v
wire [DATA_WID-1:0] sum; // From cla1 of cla_adder.v
```

```
cla_adder cla1( .sum (sum[DATA_WID-1:0]), .carry_out (carry_out), .in1 (in1[DATA_WID-1:0]), .in2 (in2[DATA_WID-1:0]), .carry_in (carry_in));
initial begin
 in1 = 16'd0;
 in2 = 16'd0;
carry_in = 1'b0;
end
initial begin
#(`DELAY)
#(`DELAY) in1 = 16'd10;
#(`DELAY) in1 = 16'd20;
#(`DELAY) in2 = 16'd10;
#(`DELAY) in2 = 16'd20;
#(`DELAY) in2 = 16'd0;
#(`DELAY*3) in1 = 16'hFFFF; in2 = 16'hFFFF;
#(`DELAY*3) in1 = 16'h7FFF; in2 = 16'hFFFF;
#(`DELAY*3) in1 = 16'hBFFF; in2 = 16'hFFFF;
end
endmodule
```

Carry Skip/Bypass Adder

```
module carry_skip_16bit(a, b, cin, sum, cout);
input [15:0] a,b;
input cin;
output cout;
output [15:0] sum;
wire [2:0] c;
carry_skip_4bit csa1(.a(a[3:0]), .b(b[3:0]), .cin(cin), .sum(sum[3:0]), .cout(c[0]));
carry_skip_4bit csa2 (.a(a[7:4]), .b(b[7:4]), .cin(c[0]), .sum(sum[7:4]), .cout(c[1]));
carry_skip_4bit csa3(.a(a[11:8]), .b(b[11:8]), .cin(c[1]), .sum(sum[11:8]), .cout(c[2]));
carry_skip_4bit csa4(.a(a[15:12]), .b(b[15:12]), .cin(c[2]), .sum(sum[15:12]), .cout(cout));
endmodule
```

TestBench

```
module carry_skip_4bit(a, b, cin, sum, cout);
input [3:0] a,b;
input cin;
output [3:0] sum;
output cout;
wire [3:0] p;
wire c0;
wire bp;
ripple_carry_4_bit rca1 (.a(a[3:0]), .b(b[3:0]), .cin(cin), .sum(sum[3:0]), .cout(c0));
```

```
generate_p p1(a,b,p,bp);
mux2X1 m0(.in0(c0),.in1(cin),.sel(bp),.out(cout));
endmodule
```

BCD ADDER

```
module  BCD_adder(S, C, A, B, C0);
input  [3:0] A, B;
input   C0;
output  [3:0] S;
output  C;
wire  C1, C2, C3, C4, C5;
wire  [3:0]X, Z;
and  (C1, Z[3], Z[2]);
and  (C2, Z[3], Z[1]);
or   (C, C3, C1,C2);
xor (C5, C, C);
assign  X[2] = C;
assign  X[1] = C;
assign  X[3] = C5;
assign  X[0] = C5;
four_bit_adder F_1 (Z, C3, A, B, C0);
four_bit_adder F_2 (S, C4, X, Z, C0);
endmodule
```

TestBench

```
module  t_BCD_adder;
wire  [3:0] S;
wire   C;
reg   [3:0]A, B;
reg    C0;
BCD_adder F1(S,  C, A, B, C0);
Initial  begin
 A[3:0] = 4'b0000; B = 4'b0000; C0 = 1'b0;  #100;
 A[3:0] = 4'b1001; B = 4'b1001; C0 = 1'b0; #100;
 A[3:0] = 4'b1000; B = 4'b0011; C0 = 1'b0; #100
 A[3:0] = 4'b0100; B = 4'b0011; C0 = 1'b0;
end  endmodule
```

Array Multiplication 8-bit

```
module  ha(sum,c_out,x,y);  //half adder
  input  x,y;
  output sum,c_out;
  assign  {c_out,sum}=x+y;
endmodule // ha
```

```
module  fa(sum,c_out,c_in,x,y);  //full adder
  input  x,y,c_in;
  output sum,c_out;
  assign  {c_out,sum}=x+y+c_in;
endmodule
module  partial(x,z,r0,r1,r2,r3,md,mr);
  input[3:0] x,z;
  input[3:0] mr,md;
  output [7:0]  r0,r1,r2,r3;
  reg  [7:0]    r0,r1,r2,r3;
  reg  [3:0]  comp;
  reg   [7:0]  tmp;

   always@(x  or  z or mr  or  md)
     begin
   comp=~mr+1;
   tmp=comp<<1;
  //r0
   if (~(x[0]|z[0]))
     r0=0;
    if (~x[0]&z[0])
     begin
       if(mr[3]) r0=mr|8'b11110000;
       else  r0=mr;
     end
   else  if (x[0]&z[0])
     begin
         if(comp[3])  r0=comp|8'b11110000;
       else  r0=comp;
     end
   if (~(x[1]|z[1]))
     r1=0;
   else  if (~x[1]&z[1])
     begin
       if(mr[3]) r1=(mr|8'b11110000)<<1;
       else  r1=mr<<1;
     end
   else  if (x[1]&z[1])
     begin
       if(comp[3])  r1=(comp|8'b11110000)<<1;
       else  r1=comp<<1;
     end
   if (~(x[2]|z[2]))
     r2=0;
```

```
   else if (~x[2]&z[2])
    begin
      if(mr[3]==1) r2=(mr|8'b11110000)<<2;
      else r2=mr<<2;
    end
   else if (x[2]&z[2])
    begin
        if(comp[3])  r2=(comp|8'b11110000)<<2;
      else r2=comp<<2;
    end

  if (~(x[3]|z[3]))
    r3=0;
   else if (~x[3]&z[3])
    begin
      if(mr[3]) r3=(mr|8'b11110000)<<3;

      else r3=mr<<3;
    end
   else if (x[3]&z[3])
    begin
        if(comp[3])  r3=(comp|8'b11110000)<<3;
      else r3=comp<<3;
    end
    end
endmodule
```

Experiment 5 : Booth Multiplication

```
module booth_encoder(mr,md,x,z);
  input[3:0] mr,md;
  output [3:0] x,z;
  //reg [3:0] mr,md;
  reg [3:0] x,z; reg
  [1:0] i;
  always@(mr or md)
  begin
   x[0]=md[0];
   z[0]=md[0];
   x[1]=md[1]&~md[0];
   z[1]=md[1]^md[0];
   x[2]=md[2]&~md[1];
   z[2]=md[2]^md[1];
   x[3]=md[3]&~md[2];
   z[3]=md[3]^md[2];
   end
endmodule

module wallace(r0,r1,r2,r3,result);
  input[7:0] r0,r1,r2,r3;
  output [7:0] result;
  wire [7:0] result;
   wire w1,w2,w3,w4,w5,w6,w7,w8,w9,w10,w11,w12,w13,w14,w15,w16,w17,w18,w19;
  wire o1,o2,o3,o4,o5,o6,o7,o8,o9,o10,o11;
  wire tmp;
   ha ha1(o1,w1,r1[2],r2[2]);
  fa fa2(o2,w2,r2[3],r3[3],r1[3]);
  fa fa3(o3,w3,r1[4],r2[4],r3[4]);
  fa fa4(o4,w4,r1[5],r2[5],r3[5]);
  fa fa5(o5,w5,r1[6],r2[6],r3[6]);
  fa fa17(o10,w16,r0[7],r1[7],r2[7]);
  ha ha6(o6,w6,o2,r0[3]);
  fa fa7(o7,w7,w2,o3,r0[4]);
  fa fa8(o8,w8,w3,o4,r0[5]);
  fa fa9(o9,w9,w4,o5,r0[6]);
  fa fa18(o11,w17,w5,o10,r3[7]);
  not not1(w19,r0[0]);
  not not2(result[0],w19);
  ha ha10(result[1],w10,r0[1],r1[1]);
  fa fa11(result[2],w11,w10,r0[2],o1);
  fa fa12(result[3],w12,w11,w1,o6);
```

```
  fa    fa13(result[4],w13,w12,w6,o7);
  fa    fa14(result[5],w14,w13,w7,o8);
  fa    fa15(result[6],w15,w14,w8,o9);
  fa fa16(result[7],w18,w15,w9,o11);
endmodule
```

Magnitude Comparator

```
module  comparator(a,b,eq,lt,gt);
input  [3:0] a,b;
output reg eq,lt,gt;
always  @(a,b)
begin
if (a==b)
begin
eq = 1'b1; lt = 1'b0; gt = 1'b0;
end
else if (a>b)
begin
eq = 1'b0; lt = 1'b0; gt = 1'b1;
end
else
begin
eq = 1'b0; lt = 1'b1; gt = 1'b0;
end  end
endmodule
```

TestBench

```
module  comparator_tst;
reg [3:0] a,b;
wire  eq,lt,gt;
comparator DUT (a,b,eq,lt,gt);
initial  begin
  a = 4'b1100;    b = 4'b1100;    #10;
  a = 4'b0100;    b = 4'b1100;    #10;
  a = 4'b1111;   b = 4'b1100;   #10;
  a = 4'b0000;   b = 4'b0000;   #10;
end
endmodule
```

LFSR

```
module lfsr (out, clk, rst);
output reg [3:0] out;
input clk, rst;
wire feedback;
assign feedback = ~(out[3] ^ out[2]);
always @(posedge clk, posedge rst)
begin
if (rst)
out = 4'b0;
else
out = {out[2:0],feedback};
end
endmodule
```

TestBench

```
module lfsr_tb();
reg clk_tb, rst_tb;
wire [3:0] out_tb;
lfsr DUT(out_tb,clk_tb,rst_tb);
initial begin
 clk_tb = 0;    rst_tb = 1;   #15;
rst_tb = 0;   #200;
end
always begin
   #5;  clk_tb = ~ clk_tb;
end endmodule
```

Experiment 6: Universal Shift Register

```
module  uni_shift_8b(op,clk,rst_a, load,sh_ro_lt_rt,ip);
 output reg [7:0] op;
 input  load;
 input  [1:0] sh_ro_lt_rt;
 input  [7:0] ip;
 input  clk, rst_a;
 reg [7:0]temp;
   always  @(posedge clk or posedge  rst_a)
 begin
  if  (rst_a)
     op  =  0;
     else
    case(load)
     1'b1:
     begin                    //Load Input
       temp  = ip;
     end
     1'b0:                    //Operation
     case (sh_ro_lt_rt)
       2'b00:  op = temp<<1;     //Left Shift
       2'b01:  op = temp>>1;     //Right Shift
       2'b10:  op = {temp[6:0],temp[7]}; //Rotate Left
       2'b11:  op = {temp[0], temp[7:1]};  //Rotate Right

       default:  $display("Invalid Shift Control  Input");
     endcase
            default:  $display("Invalid Load Control  Input");
    endcase
   end
   endmodule
```

TestBench

```
module  uni_shift_8b_tst;
 reg [7:0] ip;
 reg [1:0] sh_ro_lt_rt;
 reg       load,rst_a,clk;
 wire [7:0] op;
  uni_shift_8b u1 (.op(op), .ip(ip), .sh_ro_lt_rt(sh_ro_lt_rt), .load(load) , .rst_a(rst_a),
.clk(clk));
```

```
initial begin
    clk=1'b1;
    forever #50 clk=~clk;
  end
    initial begin
   ip = 8'b11001100;    rst_a = 1'b1;   load = 1'b1;   sh_ro_lt_rt = 2'b00;    #100;
   ip = 8'b10001100;   rst_a = 1'b0;   load = 1'b1;   sh_ro_lt_rt = 2'b01;   #100;
   ip = 8'b11001100;   load = 1'b0;   sh_ro_lt_rt = 2'b01;   #100;
   ip = 8'b10101101;   load = 1'b1;   sh_ro_lt_rt = 2'b01;   #100;
   ip = 8'b11001101;   load = 1'b0;   sh_ro_lt_rt = 2'b01;   #100;
   ip = 8'b11101100;   load = 1'b1;   sh_ro_lt_rt = 2'b10;   #100;
   ip = 8'b11110000;   load = 1'b0;   sh_ro_lt_rt = 2'b10;   #100;
   ip = 8'b11001100;   load = 1'b1;   sh_ro_lt_rt = 2'b11;   #100;
   ip = 8'b11001101;   load = 1'b0;   sh_ro_lt_rt = 2'b11;   #100;
   ip = 8'b11001000;   load = 1'b1;   sh_ro_lt_rt = 2'b11;   #100;
   $stop;
    end
  endmodule
```