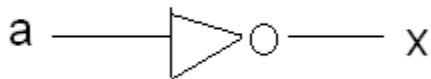




**PART A (DIGITAL DESIGN)****EXPERIMENT 1: Synthesis and Simulation of Inverter Using Verilog Code****THEORY:**

In digital logic, an **inverter** or **NOT gate** is a logic gate which implements logical negation.

This represents perfect switching behavior, which is the defining assumption in Digital electronics. In practice, actual devices have electrical characteristics that must be carefully considered when designing inverters. In fact, the non-ideal transition region behavior of a CMOS inverter makes it useful in analog electronics as a class A amplifier (e.g., as the output stage of an operational amplifier).

**SYMBOL:****TRUTH TABLE:**

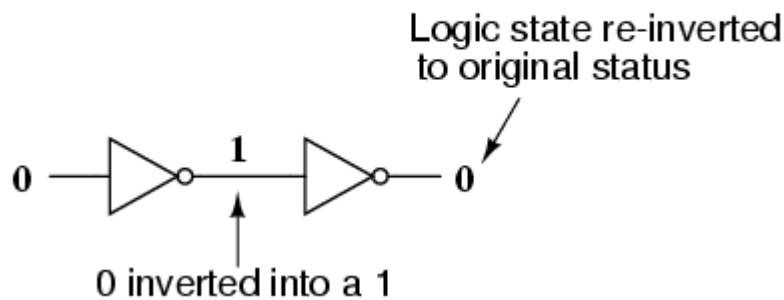
Inputs	Outputs
<b>a</b>	<b>x</b>
0	1
1	0

Verilog code	Test bench
<pre>module inv (a,b); input a; output b; assign b = ~a; endmodule</pre>	<pre>module inv_tb; reg a; wire b; initial begin     #10 a = 1'b0;     #10 a = 1'b1;     #10 a = 1'b0; end inv inv_t(a,b,); endmodule</pre>

**RESULTS:**

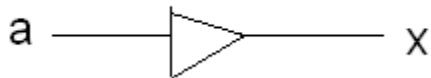
**EXPERIMENT 2: Synthesis and Simulation of Buffer Using Verilog Code****THEORY:**

If we were to connect two inverter gates together so that the output of one fed into the input of another, the two inversion functions would "cancel" each other out so that there would be no inversion from input to final output:

*Double inversion*

While this may seem like a pointless thing to do, it does have practical application. Remember that gate circuits are signal amplifiers, regardless of what logic function they may perform. A weak signal source (one that is not capable of sourcing or sinking very much current to a load) may be boosted by means of two inverters like the pair shown in the illustration. The logic level is unchanged, but the full current-sourcing or -sinking capabilities of the final inverter are available to drive a load resistance if needed.

For this purpose, a special logic gate called a buffer is manufactured to perform the same function as two inverters. Its symbol is simply a triangle, with no inverting "bubble" on the output terminal.

**SYMBOL:****TRUTH TABLE:**

Inputs	Outputs
<b>a</b>	<b>x</b>
0	0
1	1

Verilog code	Test bench
<pre>module buff (a,b); input a; output b; assign b = a; endmodule</pre>	<pre>module buff_tb; reg a; wire b; initial begin a = 1'b0; #10 a = 1'b1; #10 a = 1'b0; end buff buff_t(a,b); endmodule</pre>

**Results:**

**EXPERIMENT 3: Synthesis and Simulation of Basic Gates/Universal gates Using Verilog Code**

**THEORY:** The gate is a digital circuit with one or more input voltages but only one output voltage. By connecting the different gates in different ways, we can build circuits that perform arithmetic and other functions.

Logic gates are the basic elements that make up a digital system. The Electronic gate is a circuit that is able to operate on a number of binary inputs in order to perform a particular logic function. The types of gates available are the NOT, AND, OR, NAND, NOR, EXCLUSIVE-OR, and EXCLUSIVE-NOR.

**TRUTH TABLE:**

Inputs		Outputs						
a	b	C(0) AND	C(1) OR	C(2) XOR	C(3) NOT a	C(4) NAND	C(5) NOR	C(6) XNOR
0	0	0	0	0	1	1	1	1
0	1	0	1	1	1	1	0	0
1	0	0	1	1	0	1	0	0
1	1	1	1	0	0	0	0	1

Verilog code and	Test bench
<pre> module andgate (c,a,b); input a,b; output c;  assign c = a &amp; b;  endmodule </pre>	<pre> module and_t; reg a,b; wire c; andgate and_test(c,a,b);  initial begin #10 a = 1'b0;b = 1'b0; #10 a = 1'b0;b = 1'b1; #10 a = 1'b1;b = 1'b0; #10 a= 1'b1;b = 1'b1; end  endmodule </pre>
Verilog code nand	Test bench
<pre> module nandgate (c,a,b); input a,b; output c; reg d;  always @ (a or b) begin  d &lt;= ~(a &amp; b);  end assign c =d;  endmodule </pre>	<pre> module nand_t; reg a,b; wire c; nandgate nand_test(c,a,b);  initial begin #10 a = 1'b0;b = 1'b0; #10 a = 1'b0;b = 1'b1; #10 a = 1'b1;b = 1'b0; #10 a= 1'b1;b = 1'b1; end  endmodule </pre>

Verilog code nor	Test bench
<pre> module norgate (c,a,b); input a,b; output c; reg d;  always @ (a or b) begin  d &lt;= ~(a   b);  end  assign c = d;  endmodule </pre>	<pre> module nor_t; reg a,b; wire c; norgate nor_test(c,a,b);  initial begin #10 a = 1'b0;b = 1'b0; #10 a = 1'b0;b = 1'b1; #10 a = 1'b1;b = 1'b0; #10 a= 1'b1;b = 1'b1; end  endmodule </pre>

Verilog code or	Test bench
<pre> module orgate (c,a,b); input a,b; output c;  assign c = a   b;  endmodule </pre>	<pre> module or_t; reg a,b; wire c; orgate or_test(c,a,b);  initial begin #10 a = 1'b0;b = 1'b0; #10 a = 1'b0;b = 1'b1; #10 a = 1'b1;b = 1'b0; #10 a= 1'b1;b = 1'b1; end  endmodule </pre>

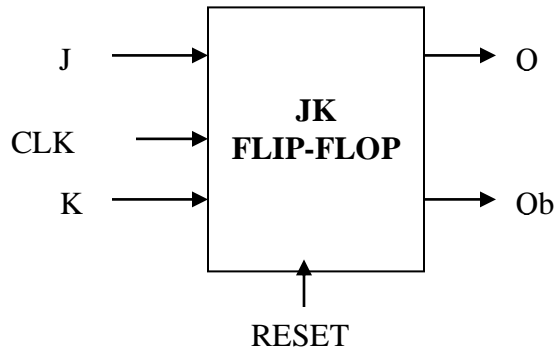
Verilog code or	Test bench
<pre>module xor (c,a,b); input a,b; output c;  always @ (a or b) begin  c &lt;= a ^ b;  end  endmodule</pre>	<pre>module xor_t; reg a,b; wire c; and xor_test(c,a,b);  initial begin #10 a = 1'b0;b = 1'b0; #10 a = 1'b0;b = 1'b1; #10 a = 1'b1;b = 1'b0; #10 a= 1'b1;b = 1'b1; end  endmodule</pre>

Results:



**EXPERIMENT 4: Synthesis and Simulation of JK, MSJK Flip-flops Using Verilog Code****AIM: Develop the HDL code for J K flip-flop.**

**THEORY:** A JK flip-flop is a refinement of the RS flip-flop in that the indeterminate state of the RS flip-flop is defined in the JK flip-flop. Inputs J and K behave like inputs S and R to set and clear the flip-flop. The JK flip-flop behaves like an RS flip-flop except when both J and K are equal to 1. When both J and K are 1, the clock pulse is transmitted through one AND gate only-the one whose  $Q=1$ , the output of upper AND gate becomes 1, upon application of clock pulse, and the flip-flop is cleared. If  $Q'$  is 1, the output of lower AND gate becomes a 1, and the flip-flop is set.

**BLOCK DIAGRAM:****TRUTH TABLE:****JK-F/F Truth table**

J	K	$Q^*$	Action
0	0	Q	No Change
0	1	0	Reset
1	0	1	Set
1	1	Q	Toggle

Verilog code	Test bench
<pre> module jkflop(j,k,clk,rst,q); input j,k,clk,rst; output q; reg q; always @(posedge clk)begin     if(j==1 &amp; k==1 &amp; rst==0)begin         q &lt;= #2 ~q;     end     else if(j==1 &amp; k==0 &amp; rst==0)begin         q &lt;= #2 1;     end     else if(j==0 &amp; k==1)begin         q &lt;= #2 0;     end end always @(posedge rst)begin     q &lt;= 0; end endmodule </pre>	<pre> module main; reg j,k,clk,rst; wire q; jkflop jk(j,k,clk,rst,q); initial begin     forever begin         clk=0;         #5         clk=1;         #5         clk=0;     end end initial begin     j=0; k=0; rst=1;     #4     j=1; k=1; rst=0;     #40     rst=1;     #10     j=0; k=1;     #10     rst=0;     #10     j=1; k=0; end endmodule </pre>

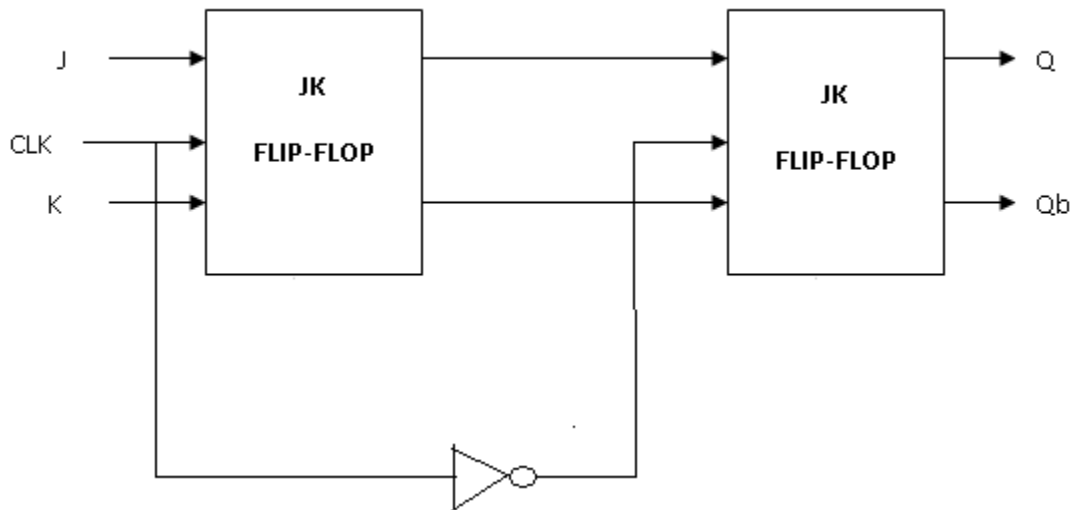
## RESULTS:

**AIM: Develop the HDL code for MSJK flip-flop.**

**THEORY:**

When both the inputs of JK f.f. are high then toggling condition occurs i.e. output changes from 0 to 1 and 1 to 0 because of clock delay. The output Q becomes uncertain at the end of clock pulse this is called race around condition. This can be removed by Master slave JK flip flop.

**BLOCK DIAGRAM:**



**TRUTH TABLE:**

J	K	Q*	Action
0	0	Q	No Change
0	1	0	Reset
1	0	1	Set
1	1	Q	Toggle

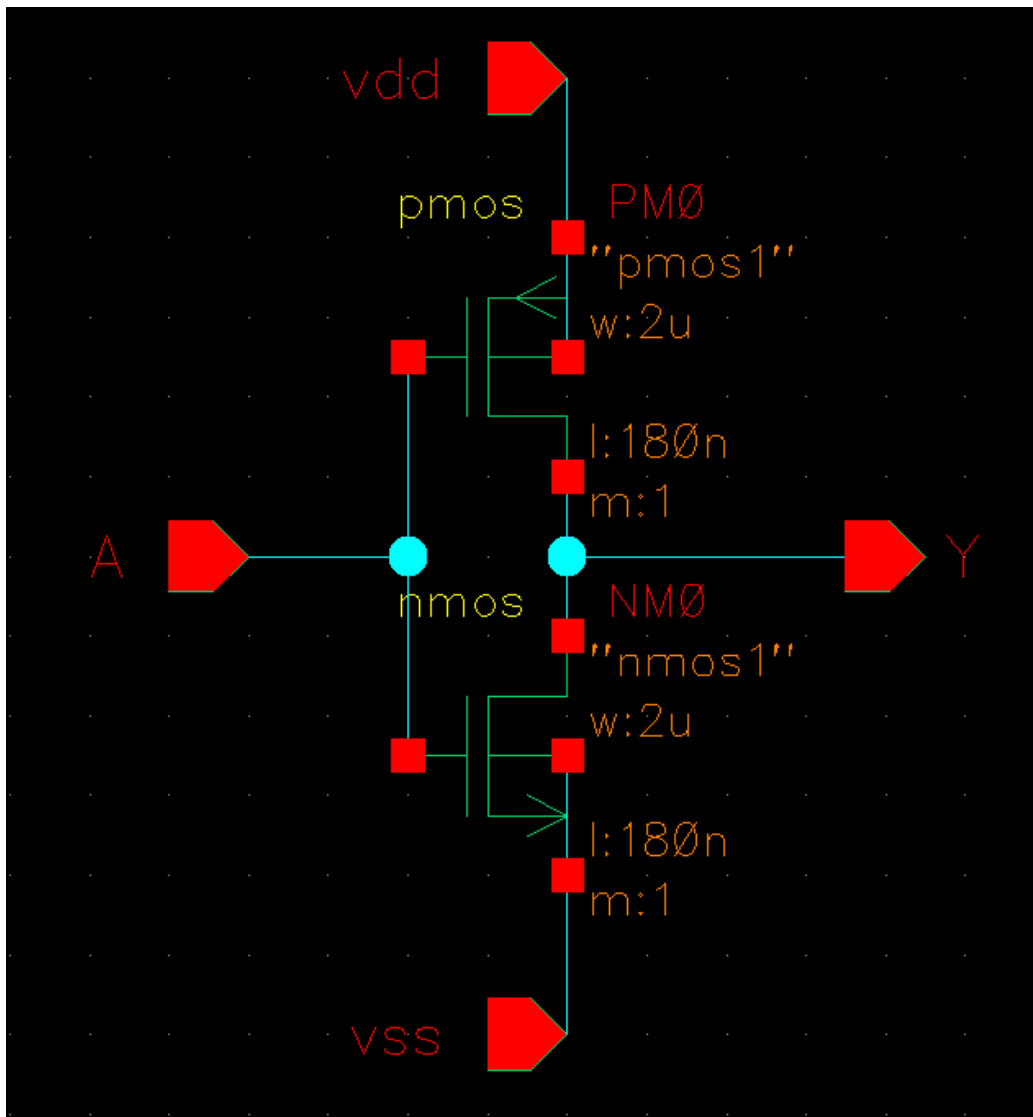
Verilog code	Test bench
<pre> module ms_jkff(q,q_bar,clk,j,k);     output q,q_bar;     input clk,j,k;     reg tq,q,q_bar;      always @(clk)     begin         if (!clk)             begin                 if (j==1'b0 &amp;&amp; k==1'b1)                     tq &lt;= 1'b0;                 else if (j==1'b1 &amp;&amp; k==1'b0)                     tq &lt;= 1'b1;                 else if (j==1'b1 &amp;&amp; k==1'b1)                     tq &lt;= ~tq;             end         if (clk)             begin                 q &lt;= tq;                 q_bar &lt;= ~tq;             end         end     endmodule      tq = 1'b0;     else if (j == 1'b1 &amp;&amp; k == 1'b0)         tq = 1'b1;     else if (j == 1'b1 &amp;&amp; k == 1'b1)         tq = ~tq;     end     if (clk)         begin             q = tq;             qb = ~tq;         end     end endmodule </pre>	<pre> module tb_ms_jkff;     reg clk,j,k;     wire q,q_bar;      wire clk2,j2,k2;      ms_jkff inst(q,q_bar,clk,j,k);      assign clk2=clk;     assign j2=j;     assign k2=k;      initial         clk = 1'b0;      always #10         clk = ~clk;      initial     begin         j = 1'b0; k = 1'b0;         #60 j = 1'b0; k = 1'b1;         #40 j = 1'b1; k = 1'b0;         #20 j = 1'b1; k = 1'b1;         #40 j = 1'b1; k = 1'b0;         #5 j = 1'b0; #20 j = 1'b1;         #10 ;     end endmodule </pre>

**RESULTS:**

## PART B (ANALOG DESIGN)

### EXPERIMENT 5: DRC and LVS analysis of CMOS Inverter Layout

## Schematic Capture



## Schematic Entry

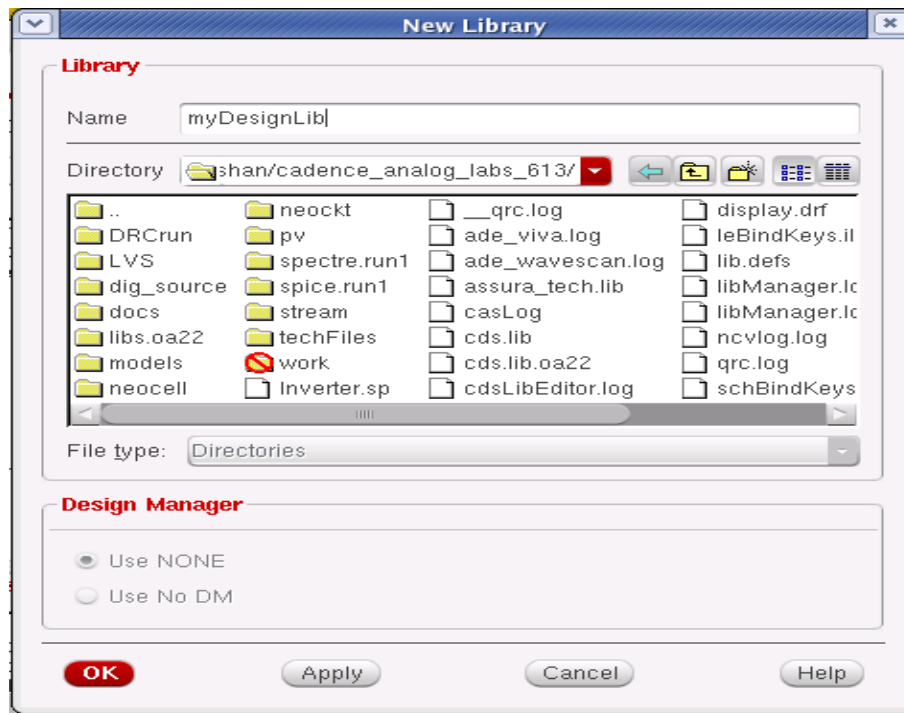
**Objective:** To create a library and build a schematic of an Inverter

---

Below steps explain the creation of new library “**myDesignLib**” and we will use the same throughout this course for building various cells that we going to create in the next labs. Execute **Tools – Library Manager** in the CIW or Virtuoso window to open Library Manager.

## Creating a New library

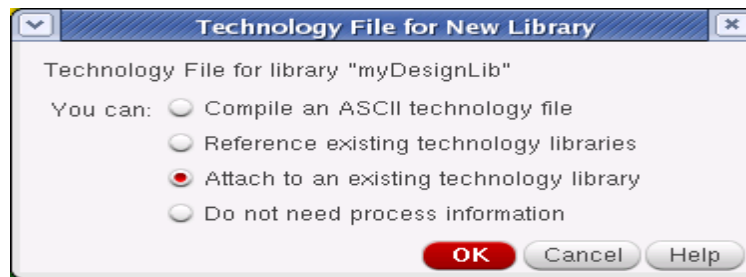
1. In the Library Manager, execute **File - New – Library**. The new library form appears.
2. In the “New Library” form, type “**myDesignLib**” in the Name section.



4. In the field of Directory section, verify that the path to the library is set to **~/Database/cadence\_analog\_labs\_613** and click **OK**.

**Note:** A technology file is not required if you are not interested to do the layouts for the design

5. In the next “**Technology File for New library**” form, select option **Attach to an existing techfile** and click **OK**.

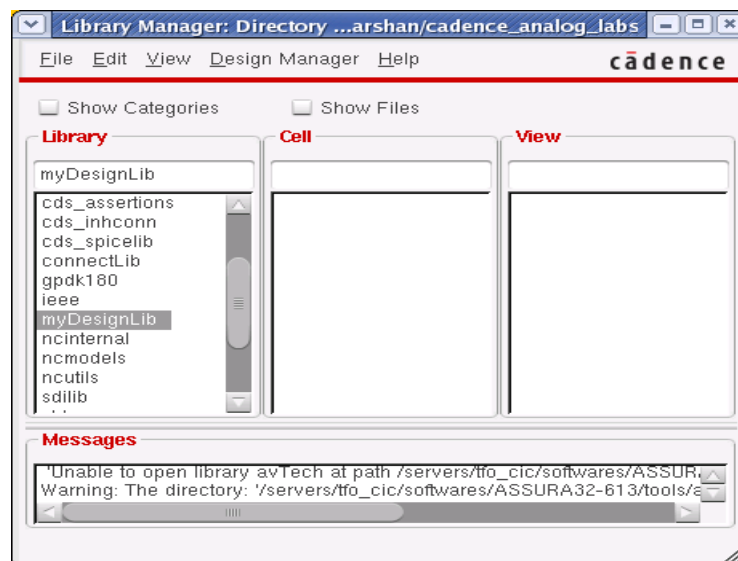


6. In the “**Attach Design Library to Technology File**” form, select **gpdK180** from the cyclic field and click **OK**.



7. After creating a new library you can verify it from the library manager.

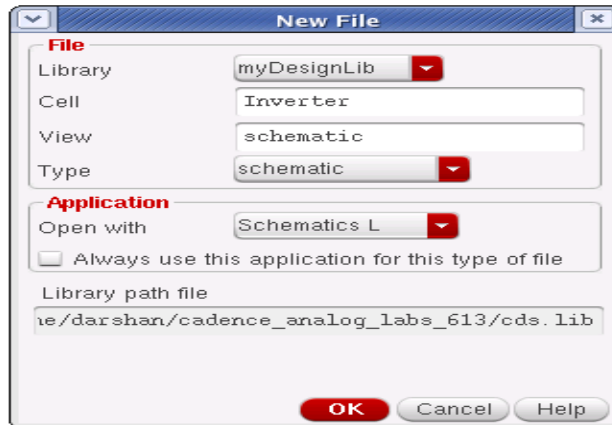
8. If you right click on the “**myDesignLib**” and select properties, you will find that **gpdK180** library is attached as techlib to “**myDesignLib**”.



## 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 at the start of this lab.

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



Do not edit the **Library path file** and the one above might be different from the path shown in your form.

3. Click **OK** when done the above settings. A blank schematic window for the **Inverter** design appears.

## Adding Components to schematic



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



**Tip:** You can also execute **Create — Instance** or press **i**.

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 Inverter schematic.



Library name	Cell Name	Properties/Comments
gpd180	pmos	For M0: Model name = pmos1, W= wp, L=180n
gpd180	nmos	For M1: Model name = nmos1, W= 2u, L=180n

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** 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
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 form after placing the pins.

In the schematic window, execute **Window— 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. 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 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

**Objective: To create a symbol for the Inverter**

---

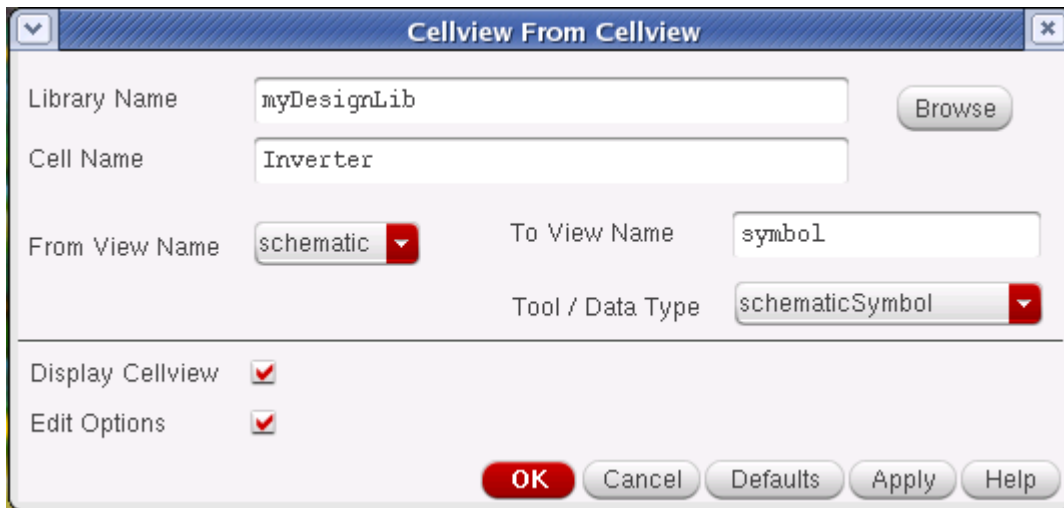
In this section, you will create a symbol for your inverter design so you can place it in a test circuit for simulation. A symbol view is extremely important step in the design process. The symbol view must exist for the

schematic to be used in a hierarchy. In addition, the symbol has attached properties (cdsParam) that facilitate the simulation and the design of the circuit.

1. In the Inverter schematic window, execute **Create — Cellview— From Cellview**.

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

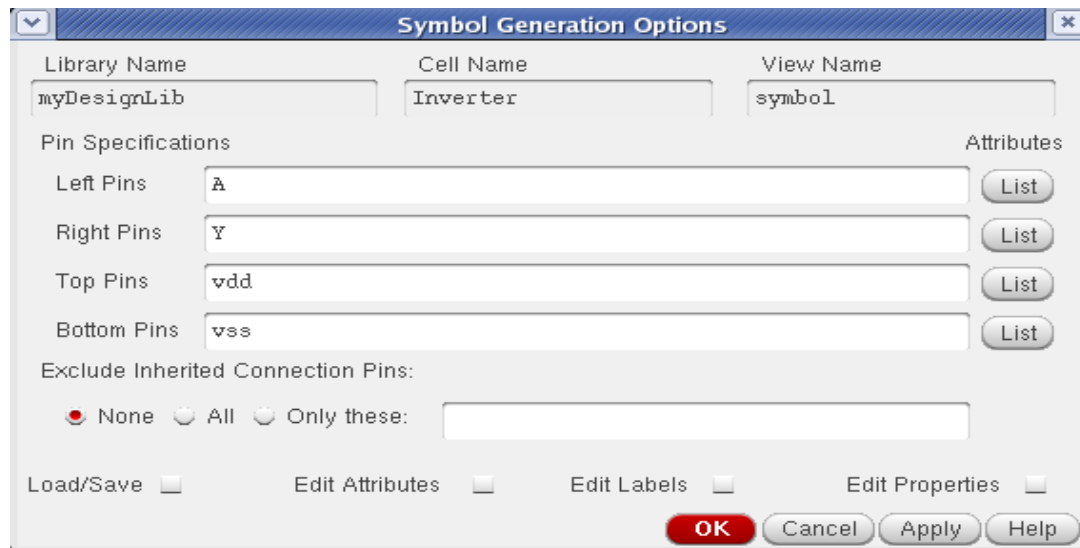
2. Verify that the **From View Name** field is set to **schematic**, and the **To View Name** field is set to **symbol**, with the **Tool/Data Type** set as **SchematicSymbol**.



3. Click **OK** in the **Cellview From Cellview** form.

The Symbol Generation Form appears.

4. Modify the **Pin Specifications** as follows:

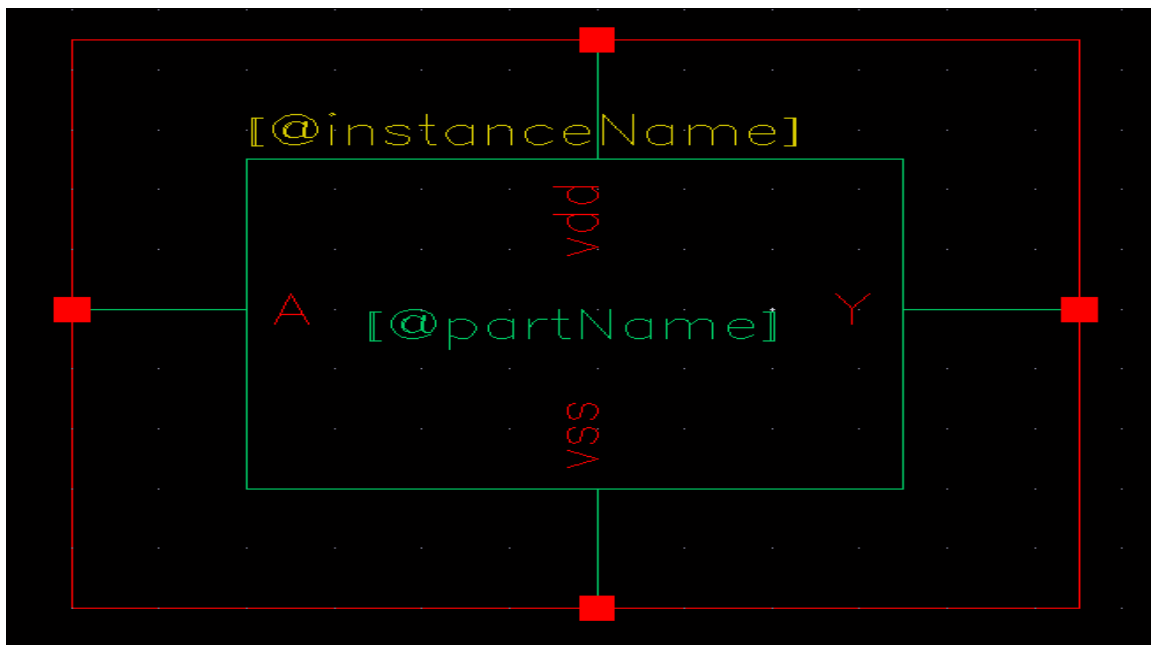


The dialog box titled "Symbol Generation Options" contains the following fields and controls:

- Library Name:** myDesignLib
- Cell Name:** Inverter
- View Name:** symbol
- Pin Specifications:**
  - Left Pins: A
  - Right Pins: Y
  - Top Pins: vdd
  - Bottom Pins: vss
- Attributes:** Four "List" buttons corresponding to the pin specifications.
- Exclude Inherited Connection Pins:** Radio buttons for "None" (selected), "All", and "Only these:" followed by an empty text field.
- Checkboxes:** Load/Save, Edit Attributes, Edit Labels, and Edit Properties (all unchecked).
- Buttons:** OK (highlighted in red), Cancel, Apply, and Help.

5. Click **OK** in the Symbol Generation Options form.

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

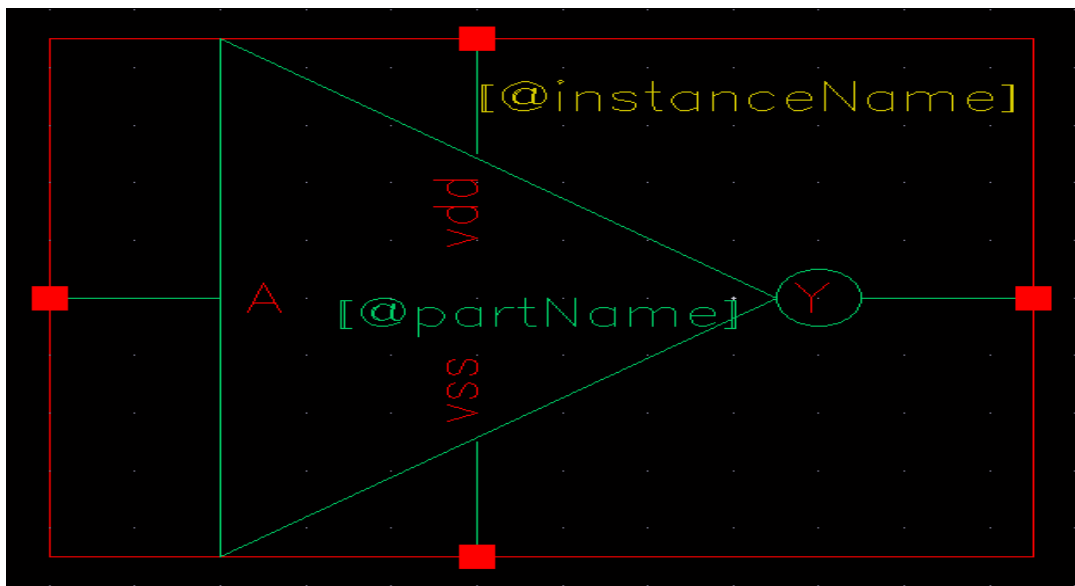


## Editing a Symbol

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.



## Building the Inverter\_Test Design

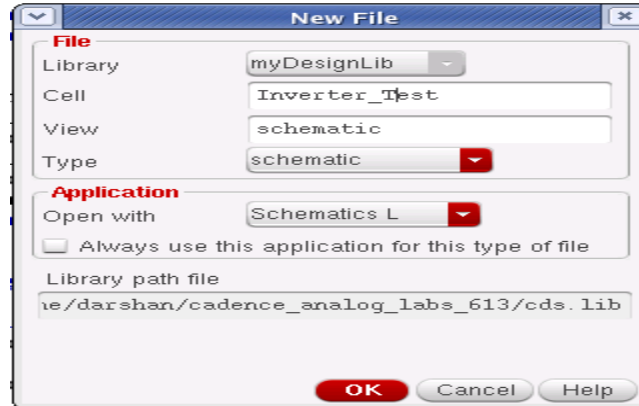
**Objective:** To build an Inverter Test circuit using your Inverter

---

### Creating the Inverter\_Test Cellview

You will create the Inverter\_Test cellview 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, execute **File— New— Cellview**.
2. Set up the New File form as follows:



3. Click **OK** when done. A blank schematic window for the **Inverter\_Test** design appears.

### Building the Inverter\_Test Circuit

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

**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)**.

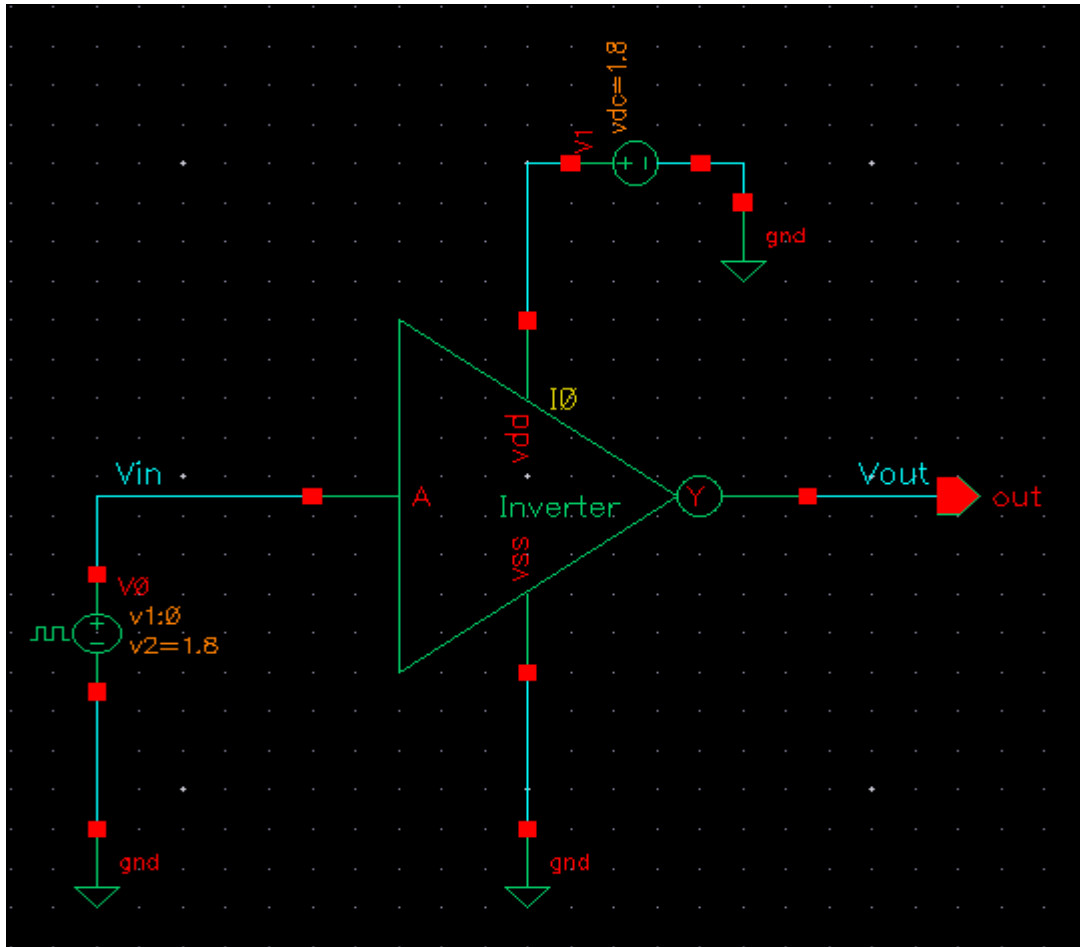
4. Click **Create — Wire Name** or press **L** to name the input (**Vin**) and output (**Vout**) wires as in the below schematic.



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



5. The schematic should look like this.



6. Leave your **Inverter\_Test** schematic window open for the next section.

## Analog Simulation with Spectre

**Objective:** To set up and run simulations on the Inverter\_Test design

---

In this section, we will run the simulation for Inverter and plot the transient, DC characteristics and we will do Parametric Analysis after the initial simulation.

### Starting the Simulation Environment

Start the Simulation Environment to run a simulation.

1. In the **Inverter\_Test** schematic window, execute **Launch – ADE L**

The **Virtuoso Analog Design Environment (ADE)** simulation window appears.

## Choosing a Simulator


Set the environment to use the **Spectre® tool**, a high speed, highly accurate analog simulator. Use this simulator with the **Inverter\_Test** design, which is made-up of analog components.

1. In the simulation window (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

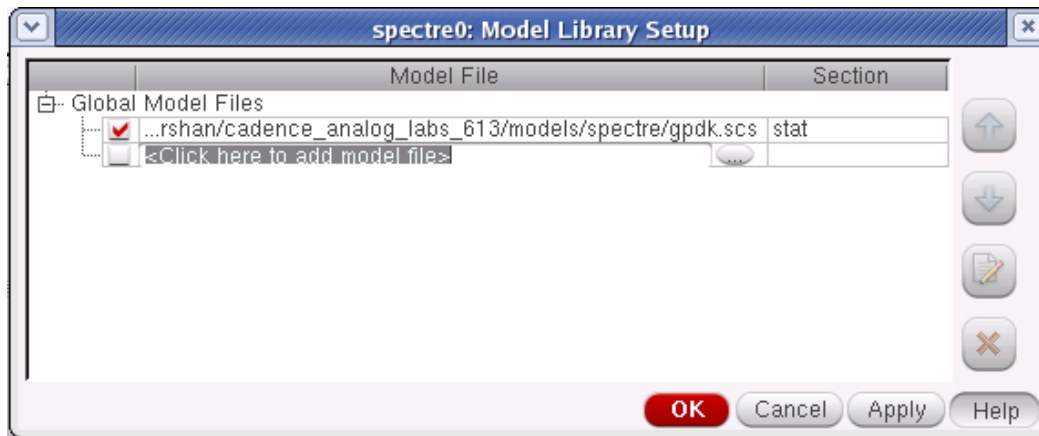
The Model Library file contains the model files that describe the nmos and pmos devices during simulation.

1. In the simulation window (ADE), Execute **Setup - Model Libraries**.

The Model Library Setup form appears. Click the **browse** button  to add **gpdk.scs** if not added by default as shown in the **Model Library Setup** form. Remember to select the section type as **stat** in front of the gpdk.scs file.

Your Model Library Setup window should now look like the below figure.





To view the model file, highlight the expression in the Model Library File field and Click **Edit File**.



2. To complete the Model Library Setup, move the cursor and click **OK**.

The Model Library Setup allows you to include multiple model files.  
It also allows you to use the Edit button to view the model file.

## Choosing Analyses

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

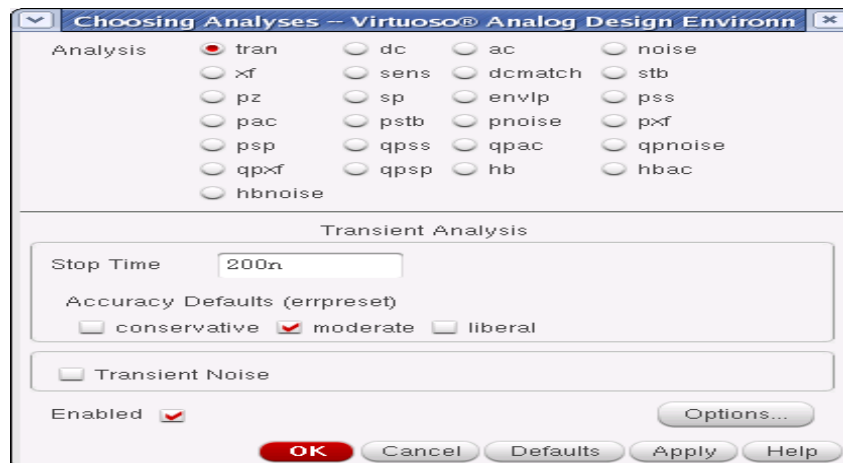


1. In the Simulation window (ADE), click the **Choose - Analyses** icon.  
You can also execute **Analyses - Choose**.

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

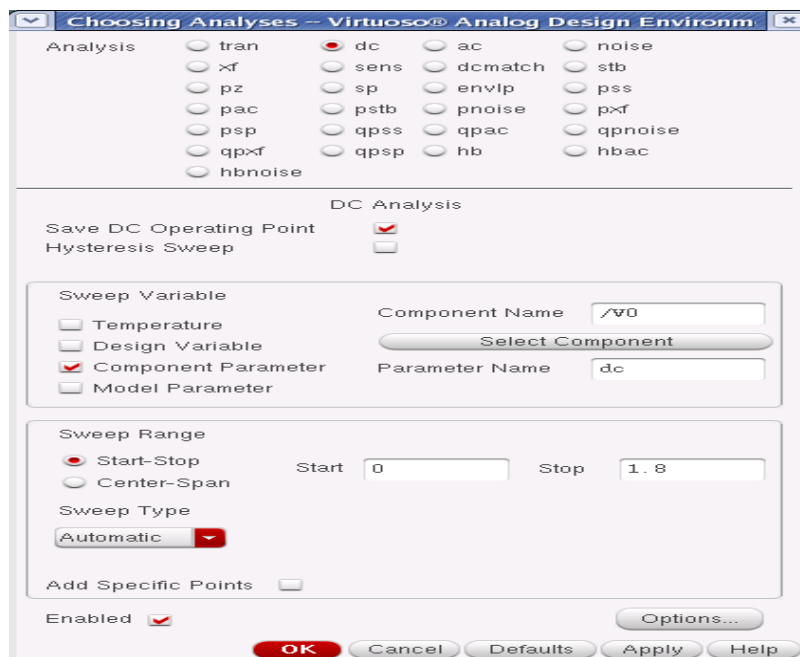
2. To setup for transient analysis

- a. In the Analysis section select **tran**
- b. Set the stop time as **200n**
- c. Click at the **moderate** or **Enabled** button at the bottom, and then click **Apply**.



3. To set up for DC Analyses:

- In the Analyses section, select **dc**.
- In the DC Analyses section, turn on **Save DC Operating Point**.
- Turn on the **Component Parameter**.
- Double click the **Select Component**, Which takes you to the schematic window.
- Select input signal **vpulse source** in the test schematic window.
- Select **“DC Voltage”** in the **Select Component Parameter** form and click OK.
- In the analysis form type **start** and **stop** voltages as **0** to **1.8** respectively.
- Check the enable button and then click **Apply**.



4. 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.

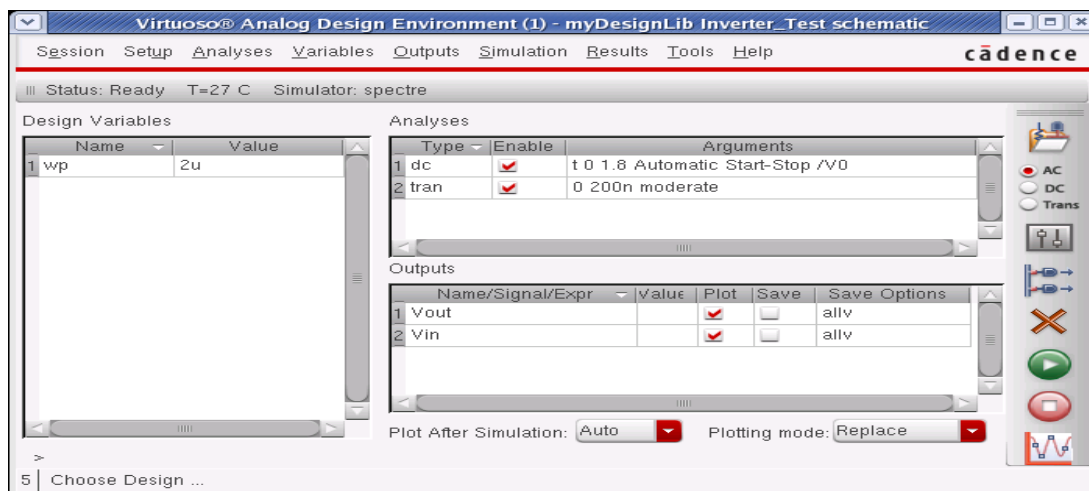
3. Click **OK** or **Cancel** in the Editing Design Variables window.

## Selecting Outputs for Plotting

1. 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 **Vout**, input net **Vin** of the Inverter. Press **ESC** with the cursor in the schematic after selecting it.

Does the simulation window look like this?



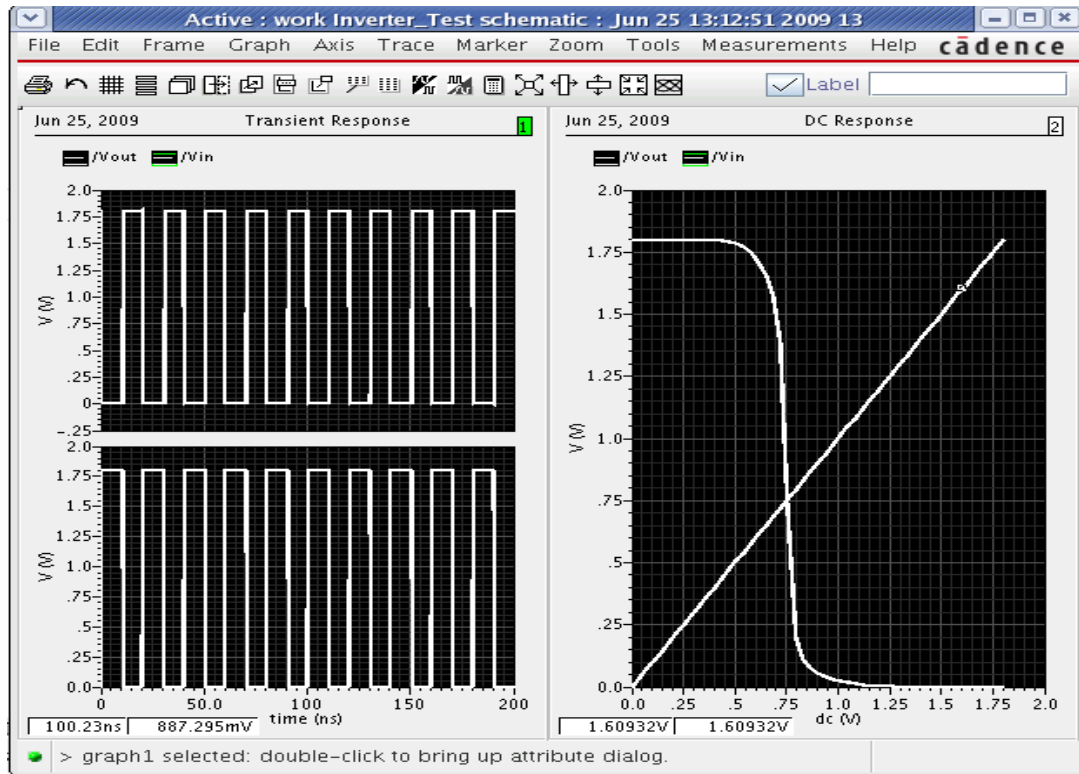
## Running the Simulation



1. Execute **Simulation – Netlist and Run** in the simulation window to start the

Simulation 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 file.



## 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.

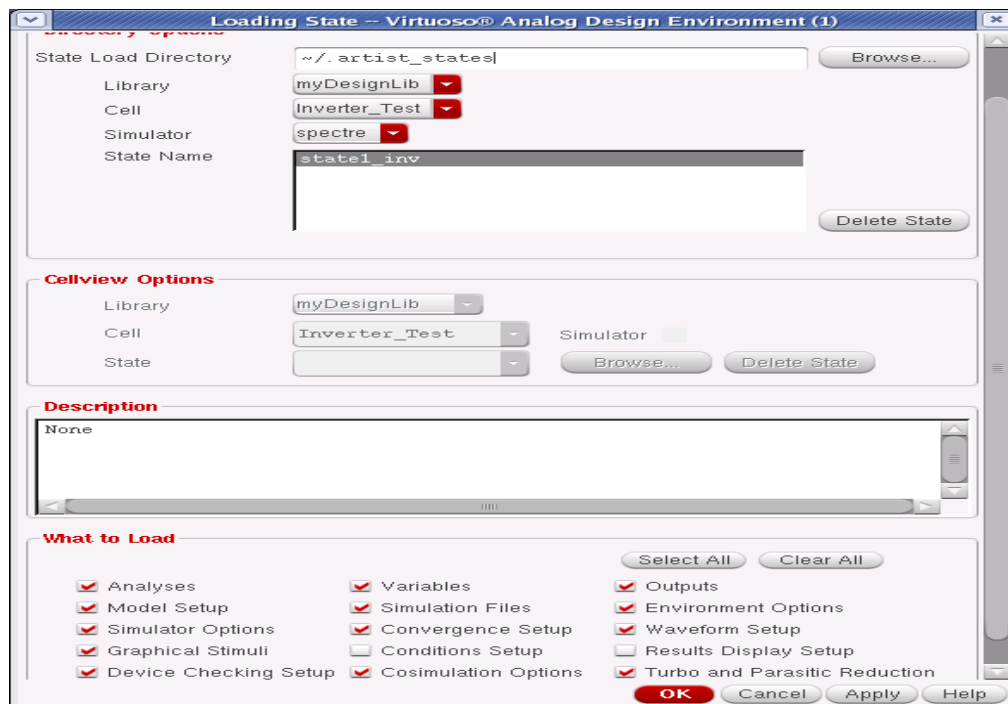
2. Set the **Save as** field to **state1\_inv** and make sure all options are selected under what to save field.

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



3. Click **OK** in the Loading State 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 edit property option. 

## Creating Layout View of Inverter

1. From the **Inverter** 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 (**Inverter**), 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/ 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.



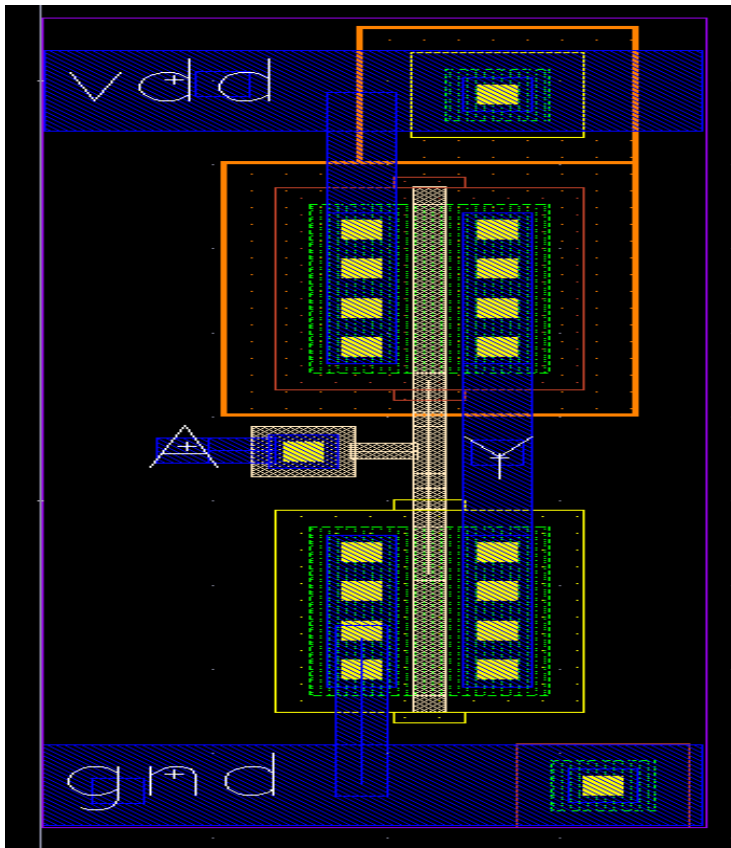
1. Execute **Create — Via** or select command to place different Contacts, as given in below table

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 — Save** or click to save the layout, and layout should appear as below.



## Physical Verification Assura DRC

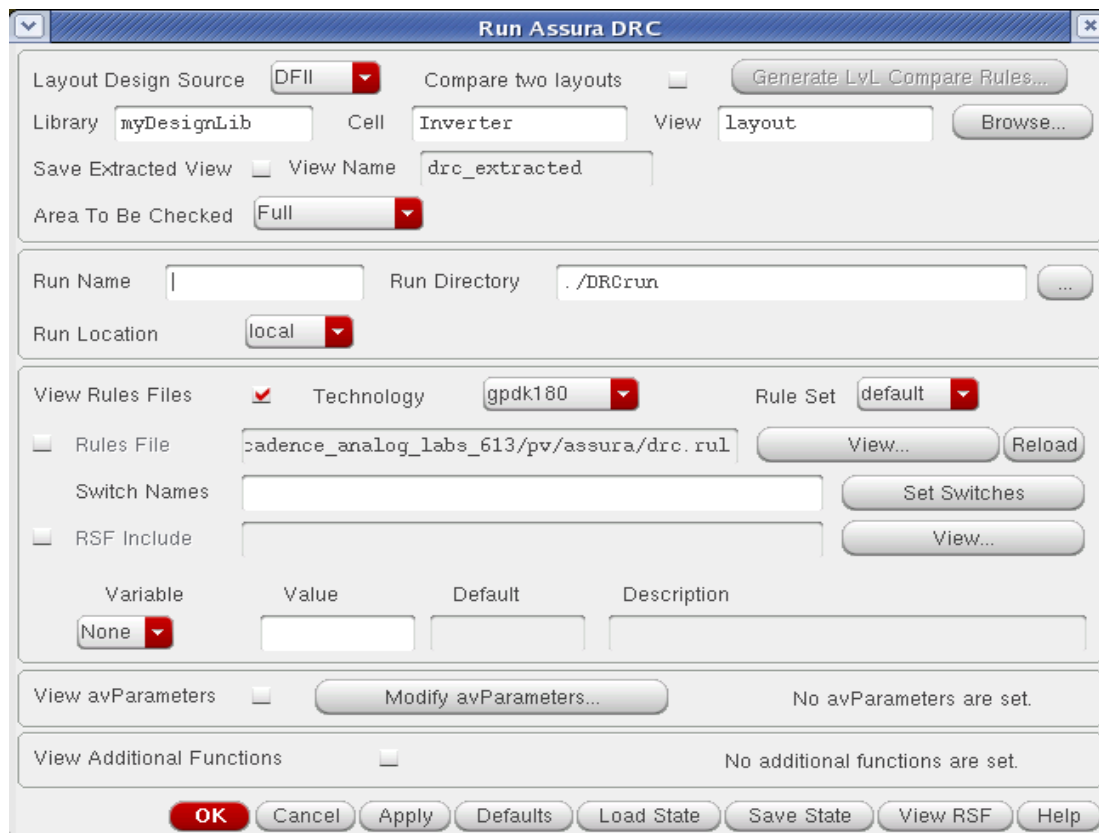
### Running a DRC

1. Open the Inverter layout from the CIW or library manager if you have closed that. Press **shift – 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 **gpd180**. This automatically loads the rule file.

Your DRC form should appear like this



3. Click **OK** to start DRC.
4. A Progress form will appear. 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 found** written 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. It will automatically load both the schematic and layout view of the cell.

2. Change the following in the form and click **OK**.

The screenshot shows the 'Run Assura LVS' dialog box. The 'Schematic Design Source' section is configured with 'DFII' as the design source, 'myDesignLib' as the library, 'Inverter' as the cell, and 'schematic' as the view. The 'Layout Design Source' section is configured with 'DFII' as the design source, 'myDesignLib' as the library, 'Inverter' as the cell, and 'layout' as the view. The 'Run Name' field is empty, and the 'Run Directory' is set to './LVS'. The 'Run Location' is set to 'local'. The 'View Rules Files' section has 'Extract Rules' checked, and the 'Compare Rules' checkbox is also checked. The 'Binding File(s)' and 'RSF Include' checkboxes are unchecked. The 'Variable' dropdown is set to 'None'. The 'Value', 'Default', and 'Description' columns are empty. The 'View avParameters' section shows '7 avParameters are set.' and the 'View avCompareRules' section shows '1 avCompare rule is set.' The 'View Additional Functions' section shows 'No additional functions are set.' The 'OK' button is highlighted in red.

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 match, 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.

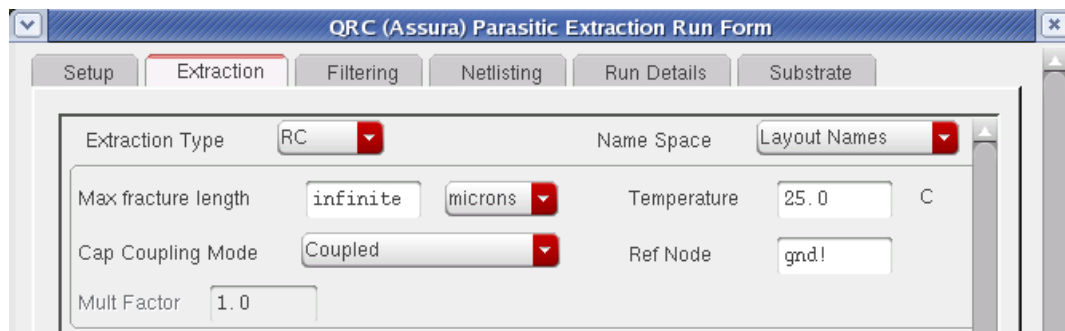
Before using RCX to extract parasitic devices for simulation, the layout should match with schematic completely to ensure that all parasites will be backannotated to the correct schematic nets.

## Running RCX

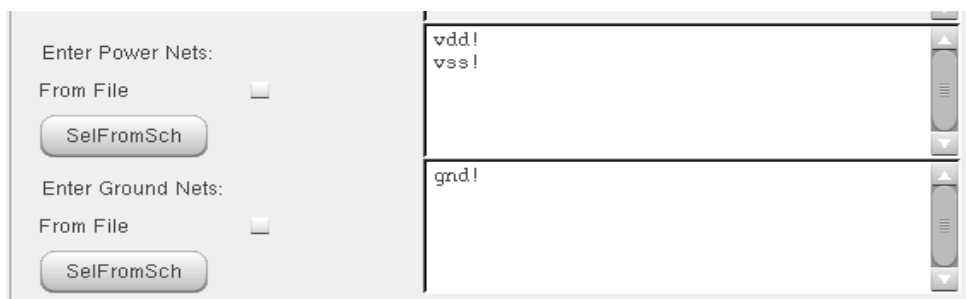
1. From the layout window execute **Assura – Run RCX**.
2. Change the following in the Assura parasitic extraction form. Select **output** type under **Setup** tab of the form.

The screenshot shows the 'QRC (Assura) Parasitic Extraction Run Form' dialog box. The 'Setup' tab is active. The 'Technology' is 'gpd180' and 'RuleSet' is 'default'. The 'Setup Dir' is '/home/darshan/cadence\_analog\_labs\_613/pv/assura/rcx'. The 'Output' is set to 'Extracted View'. The 'Lib' is 'DesignLib', 'Cell' is 'Inverter', and 'View' is 'av\_extracted'. The 'Parasitic Res Component' is 'presistor', 'Parasitic Cap Component' is 'pcapacitor', 'Parasitic Ind Component' is 'pinductor', and 'Parasitic M Component' is 'pmind'. The 'Inductance L1 Prop Id' is 'ind1' and 'Inductance L2 Prop Id' is 'ind2'. The 'Substrate Extract' checkbox is checked. The 'Extract MOS Diffusion AP' checkbox is checked. The 'Substrate Profile' is set to 'NONE'. The 'Library Prefix' and 'Library Directory' are empty. The 'OK' button is highlighted with a blue arrow.

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!** and **Enter Ground Nets** as **gnd!**

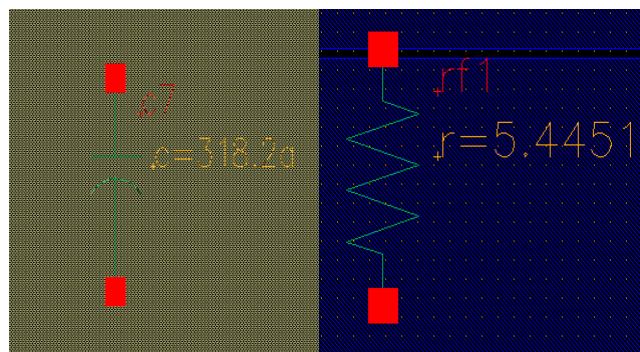


5. Click **OK** in the Assura parasitic extraction form when done.

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**.

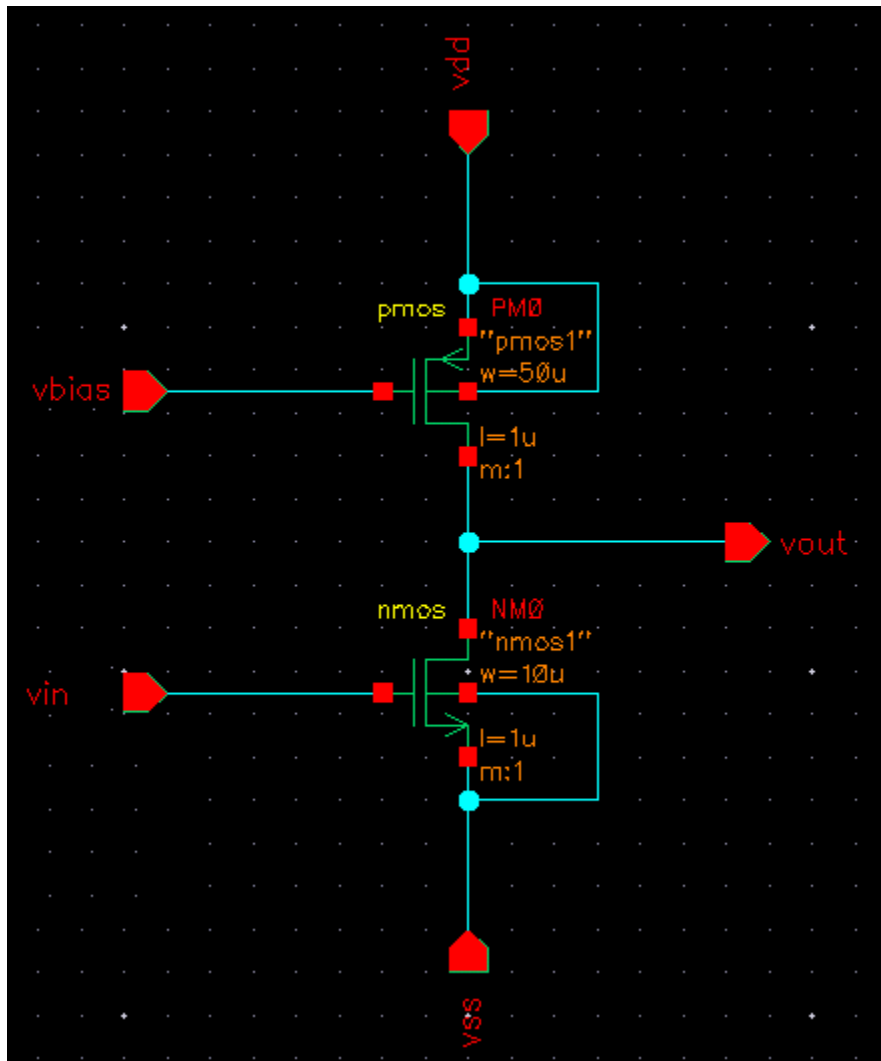
6. You can open the **av\_extracted** view from the library manager and view the parasitic.



**Results:**

**EXPERIMENT 6: DRC and LVS analysis of Common Source Amplifier Layout**

## Schematic Capture



## Schematic Entry

**Objective: To create a new cell view and build Common Source Amplifier**

---

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
gpd180	Pmos	Model Name = pmos1; W= 50u ; L= 1u
gpd180	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
vdd vss	Input

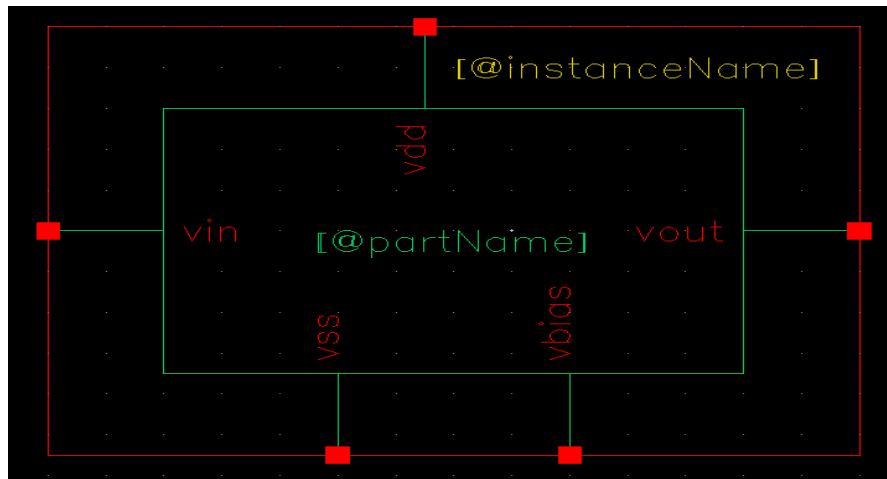
---

## Symbol Creation

**Objective:** To create a symbol for the Common Source Amplifier

---

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



## Building the Common Source Amplifier Test Design

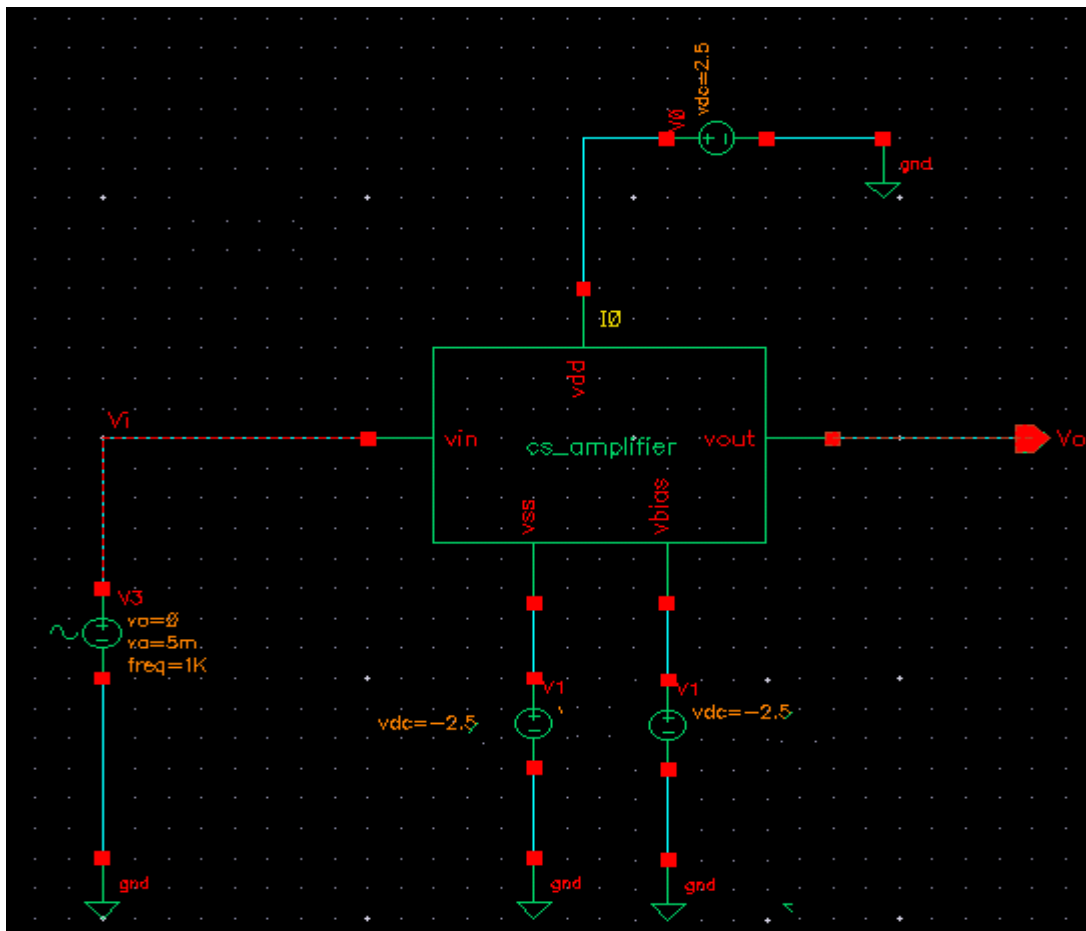
**Objective:** To build cs\_amplifier\_test circuit using your cs\_amplifier

---

Using the component list and Properties/Comments in the table, build the cs-amplifier\_test schematic as shown below.

Library	Cellview name	Properties/Comments
---------	---------------	---------------------

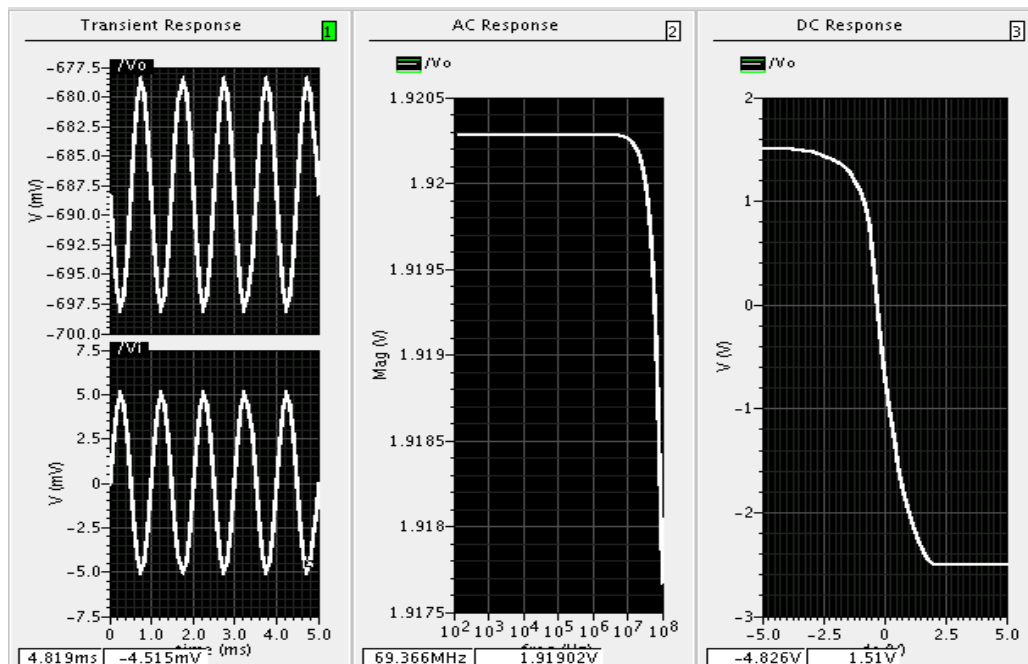
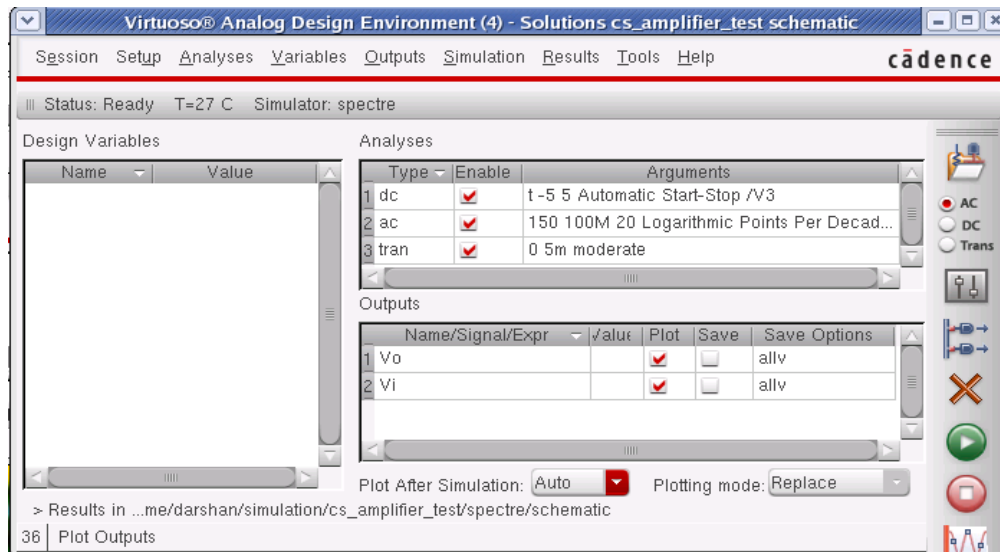
name		
myDesignLib	cs_amplifier	Symbol
analogLib	vsin	Define pulse specification as AC Magnitude= 1; DC Voltage= 0; Offset Voltage= 0; Amplitude= 5m; Frequency= 1K
analogLib	vdd,vss,gnd	vdd=2.5 ; vss= -2.5 vbias=-2.5



## Analog Simulation with Spectre

**Objective:** To set up and run simulations on the cs\_amplifier\_test design.

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

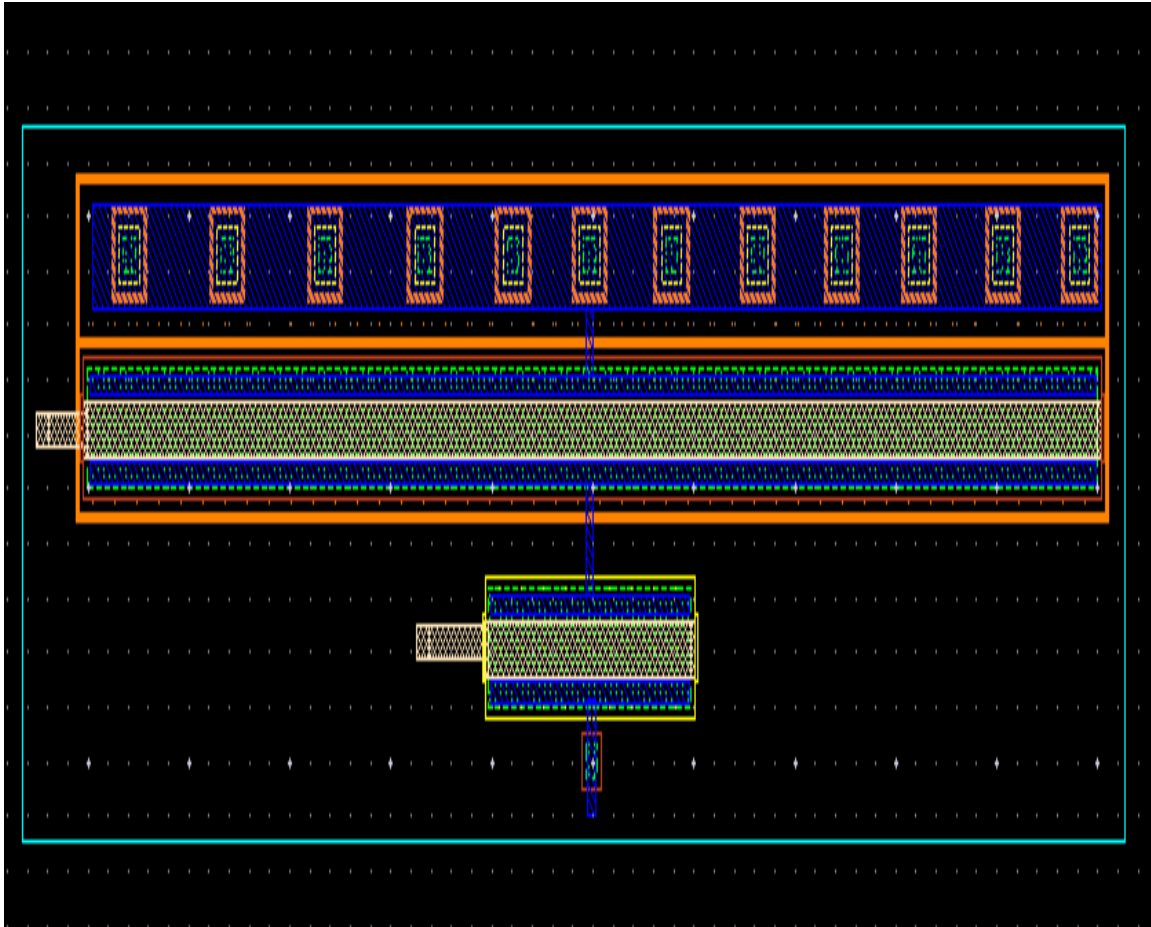


## Creating a layout view of Common Source Amplifier

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.

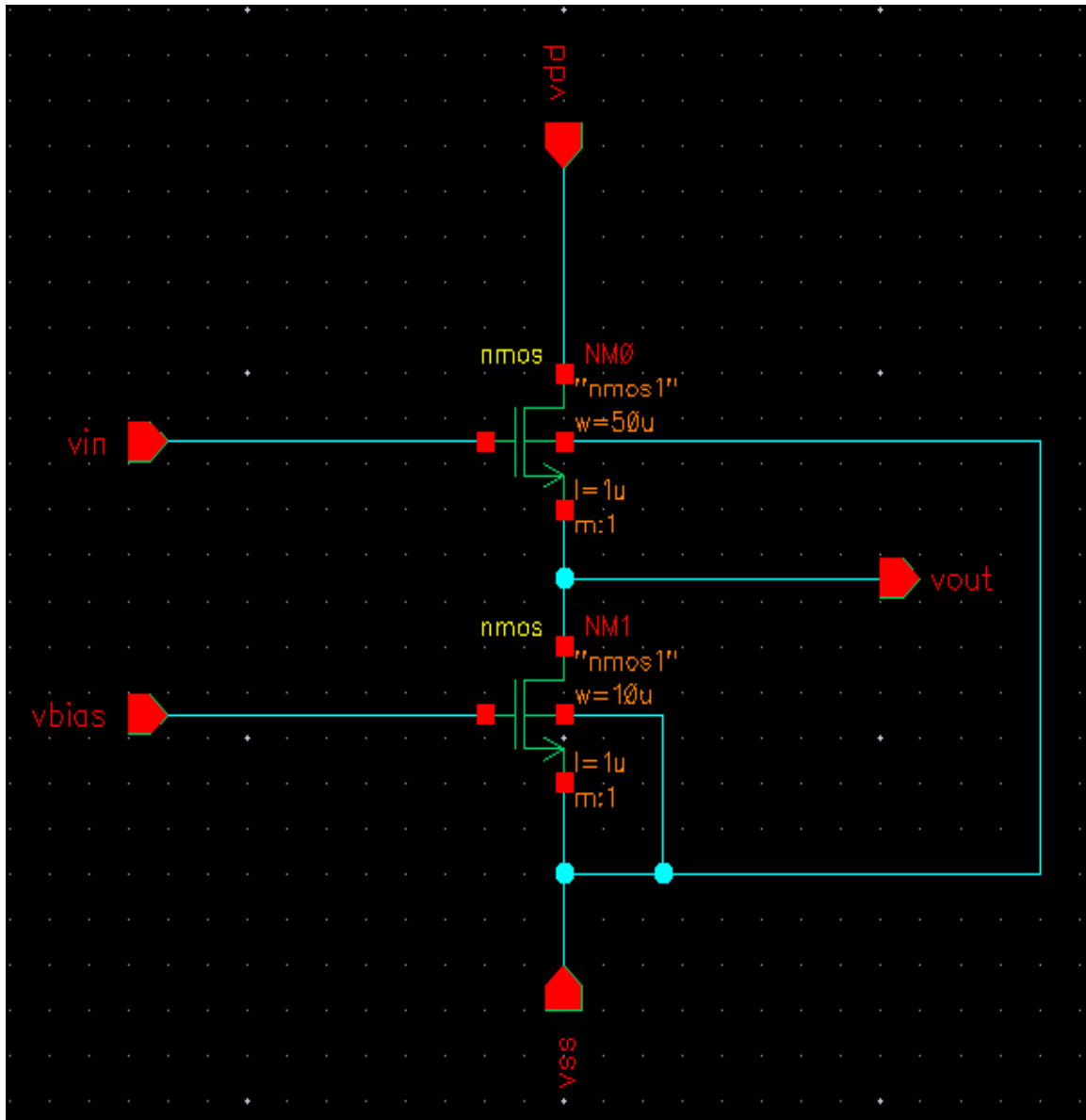


Results:



**EXPERIMENT 7: DRC and LVS analysis of Common Drain Amplifier Layout**

## Schematic Capture



## Schematic Entry

**Objective: To create a new cell view and build Common Drain Amplifier**

---

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
vdd vss	Input

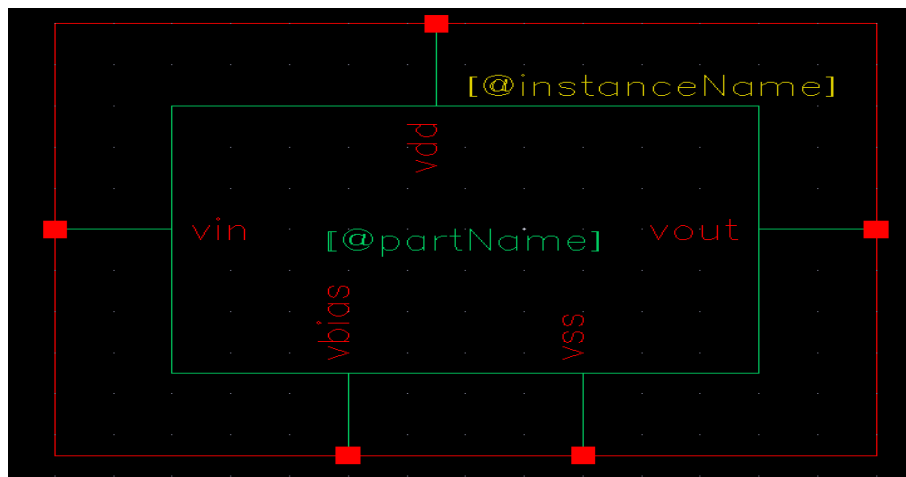
---

## Symbol Creation

**Objective: To create a symbol for the Common Drain Amplifier**

---

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



## Building the Common Drain Amplifier Test Design

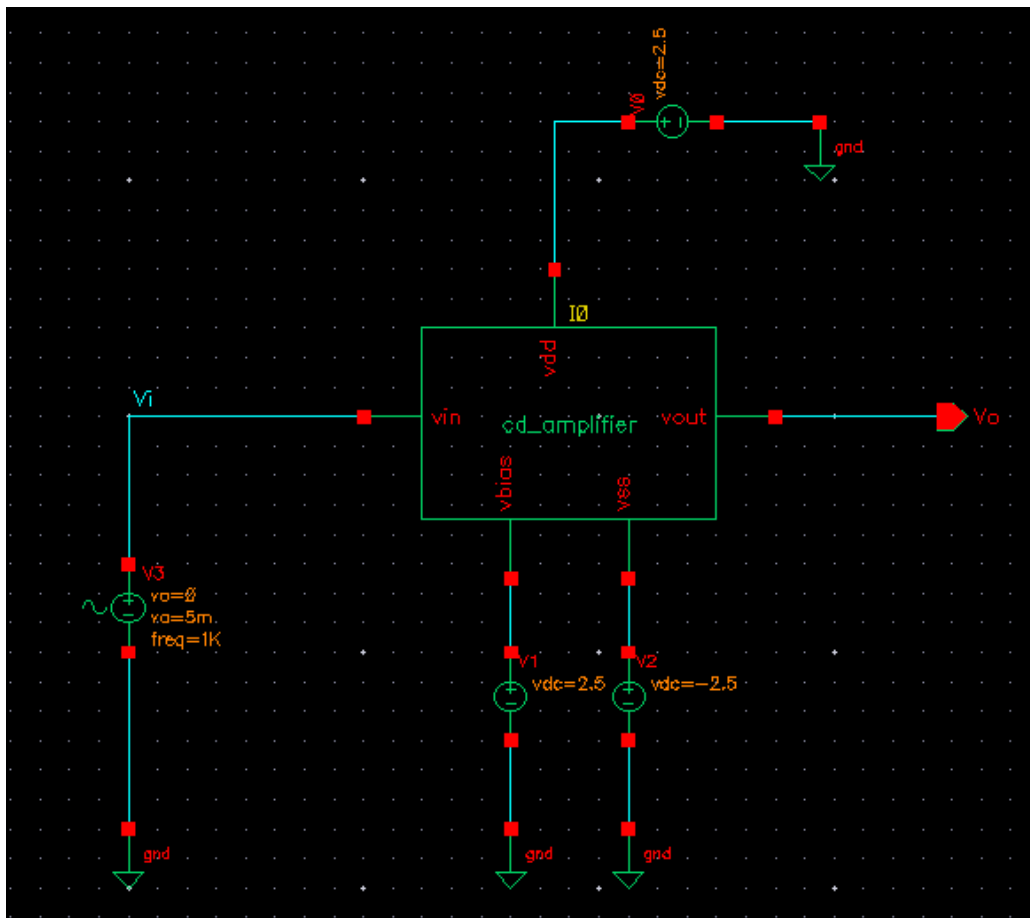
**Objective: To build cd\_amplifier\_test circuit using your cd\_amplifier**

---

Using the component list and Properties/Comments in the table,

build the cd-amplifier\_test schematic as shown below.

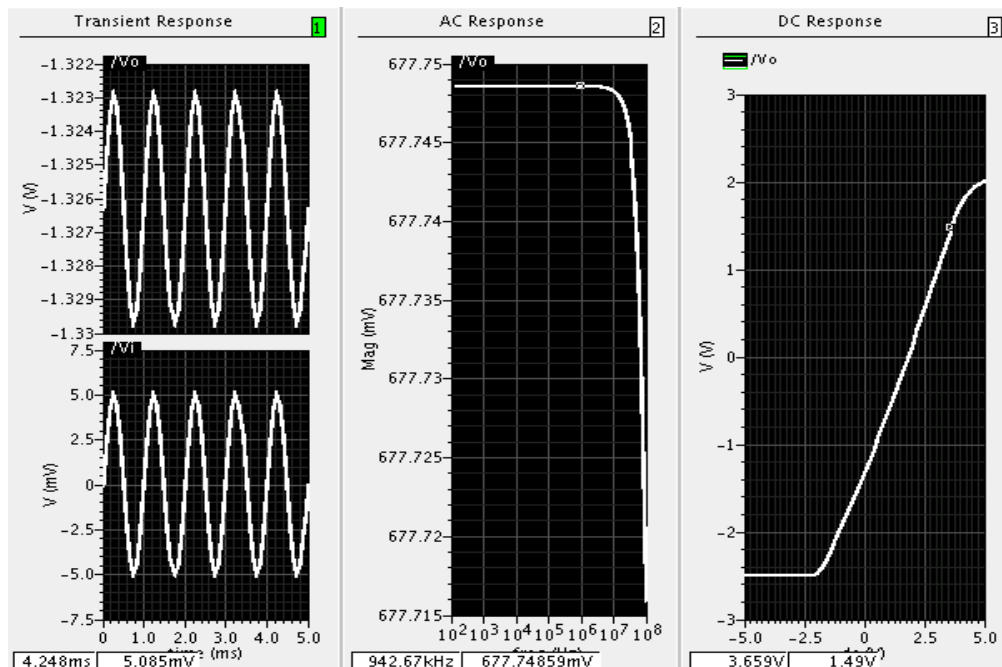
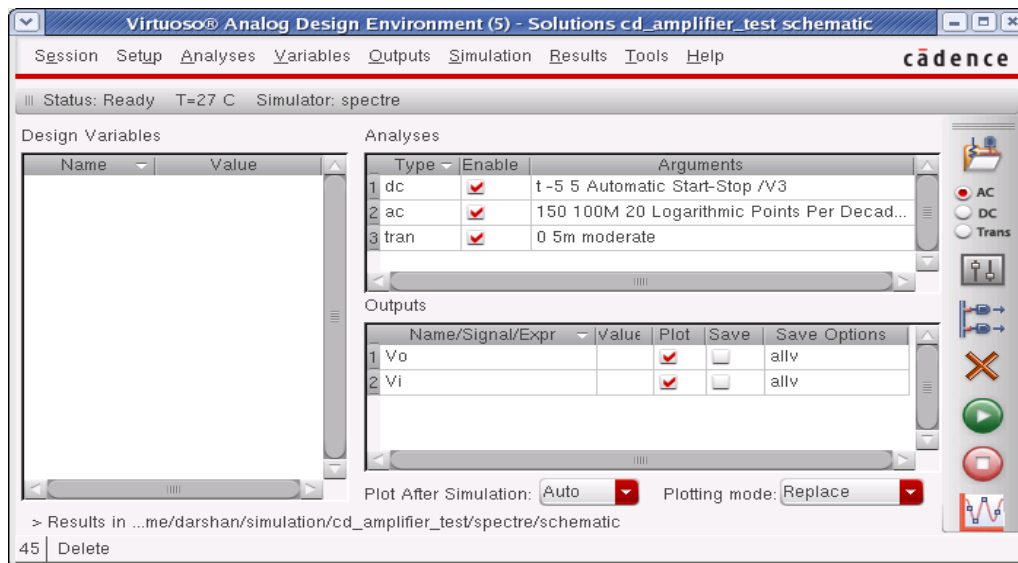
Library name	Cellview name	Properties/Comments
myDesignLib	cd_amplifier	Symbol
analogLib	vsin	Define pulse specification as AC Magnitude= 1; DC Voltage= 0; Offset Voltage= 0; Amplitude= 5m; Frequency= 1K
analogLib	vdd,vss,gnd	vdd=2.5 ; vss= -2.5



## Analog Simulation with Spectre

**Objective:** To set up and run simulations on the cd\_amplifier\_test design.

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

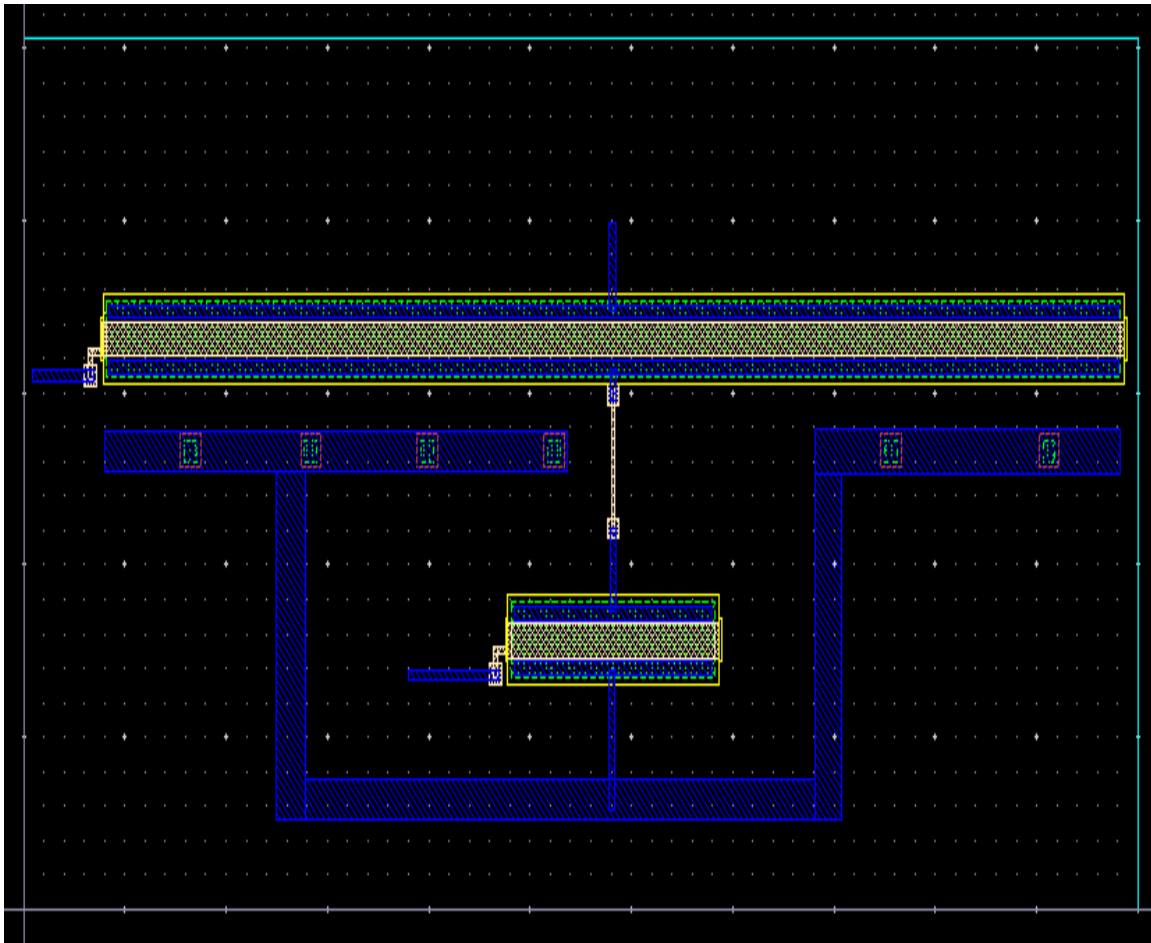


## Creating a layout view of Common Drain Amplifier

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.

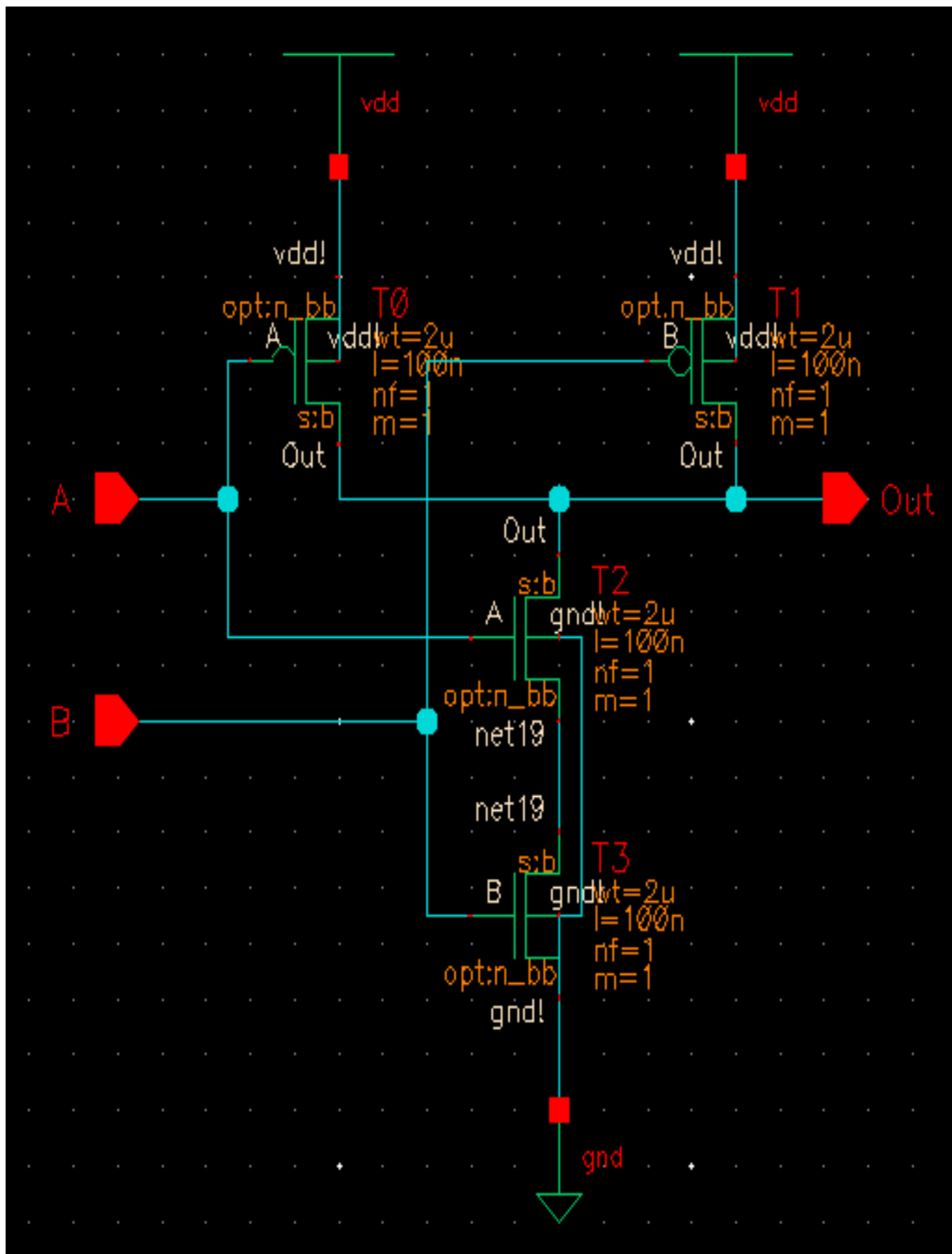
Extract RC parasites for back annotation and Re-simulation.



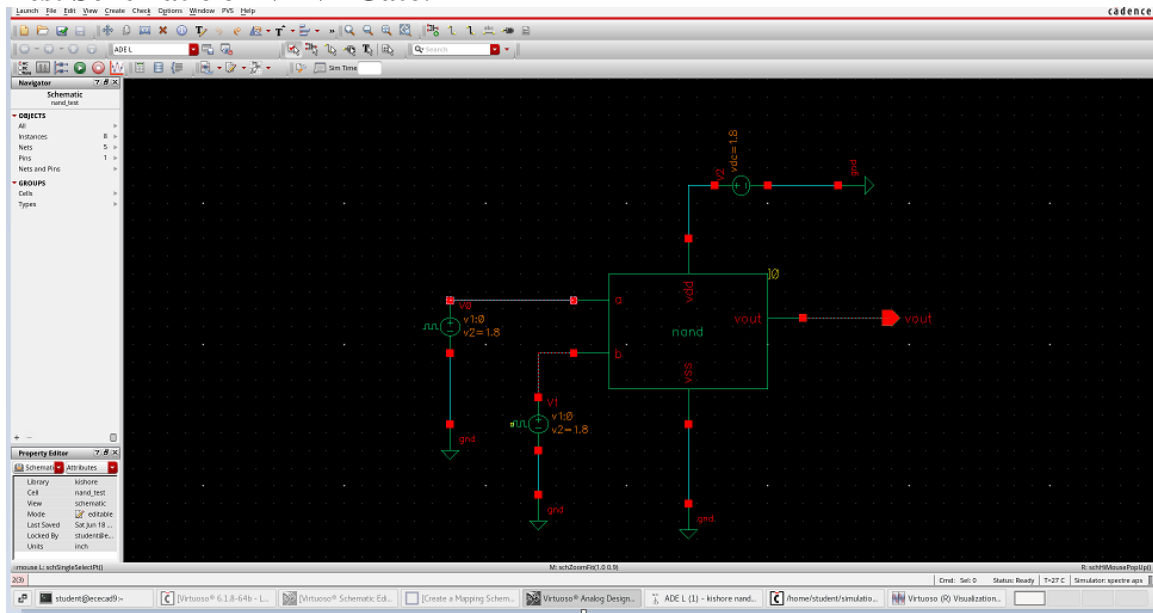
**Results:**

**EXPERIMENT 8: DC, Transient analysis of CMOS logic-Universal gates schematic**

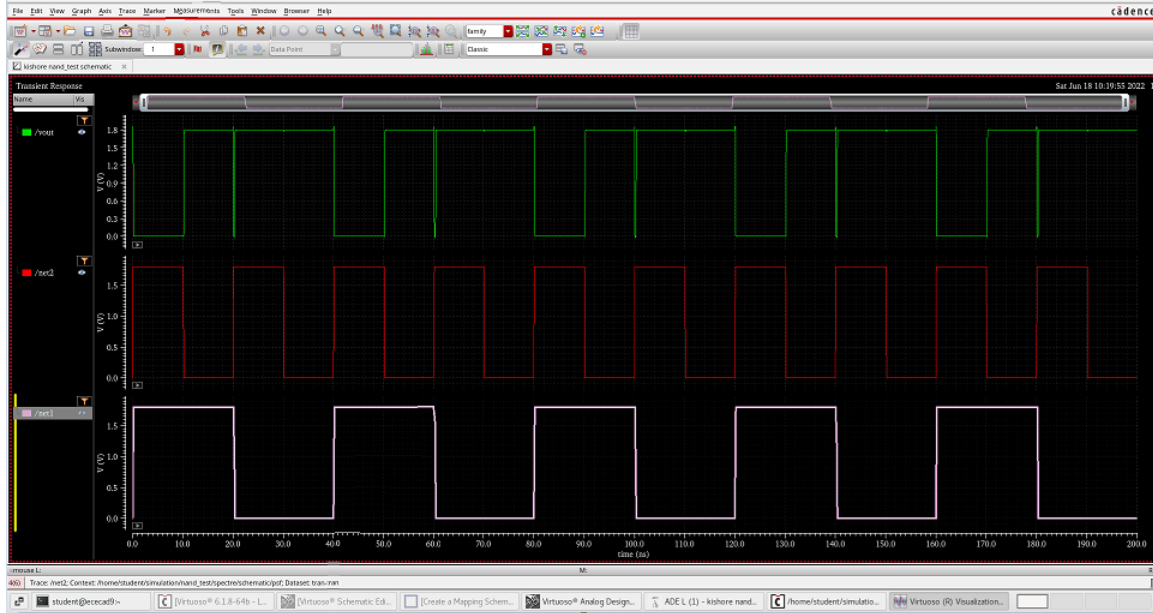
**NAND Gate Schematic**



## Test Schematic of NAND Gate:

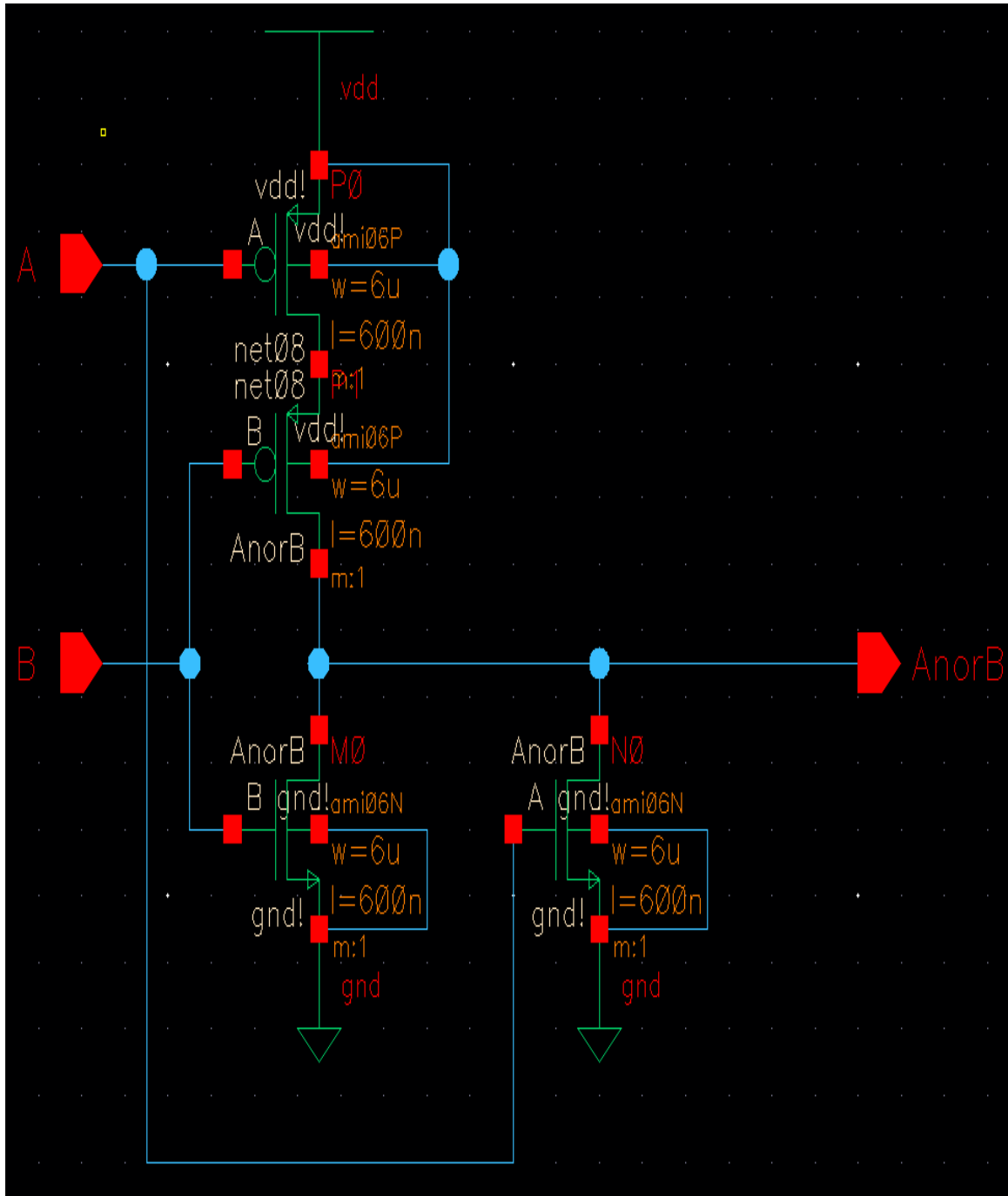


## Waveforms of NAND Gate:



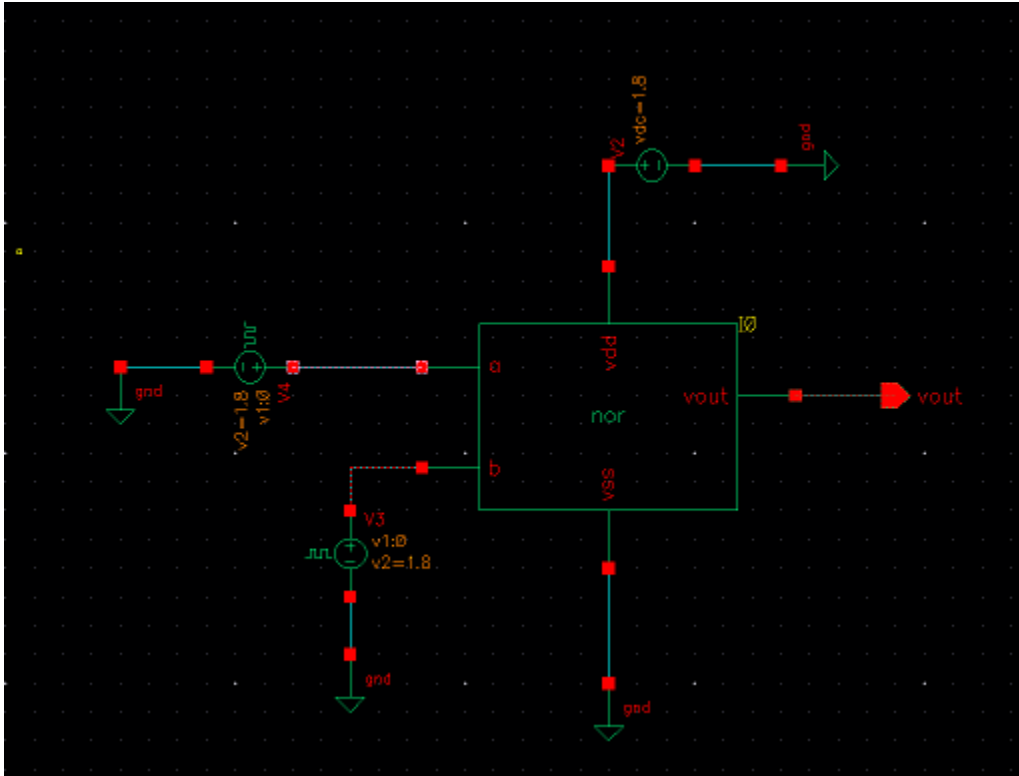
## Results:

### NOR GATE Schematic

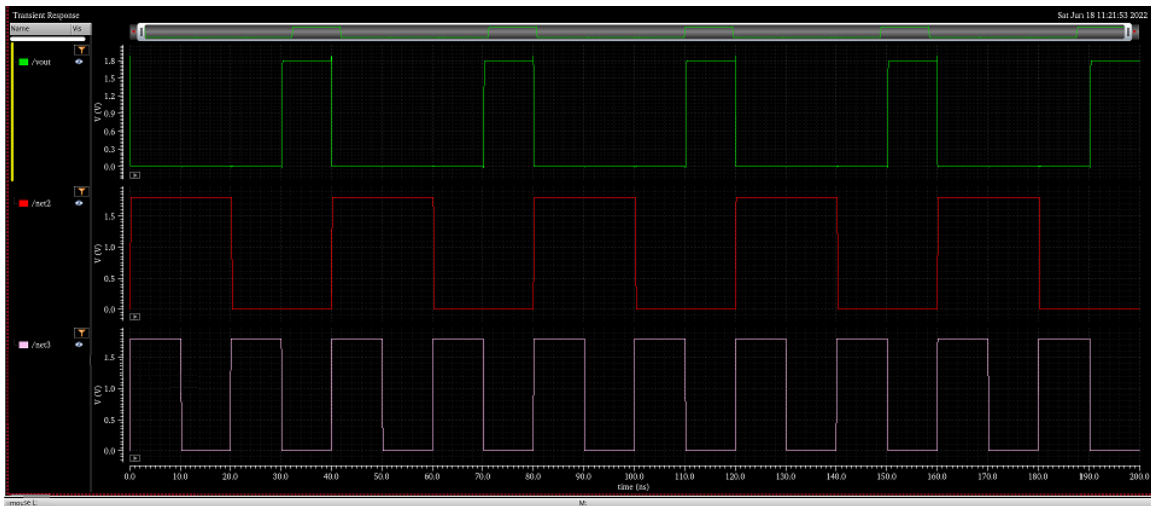




### Test Schematic NOR Gate:



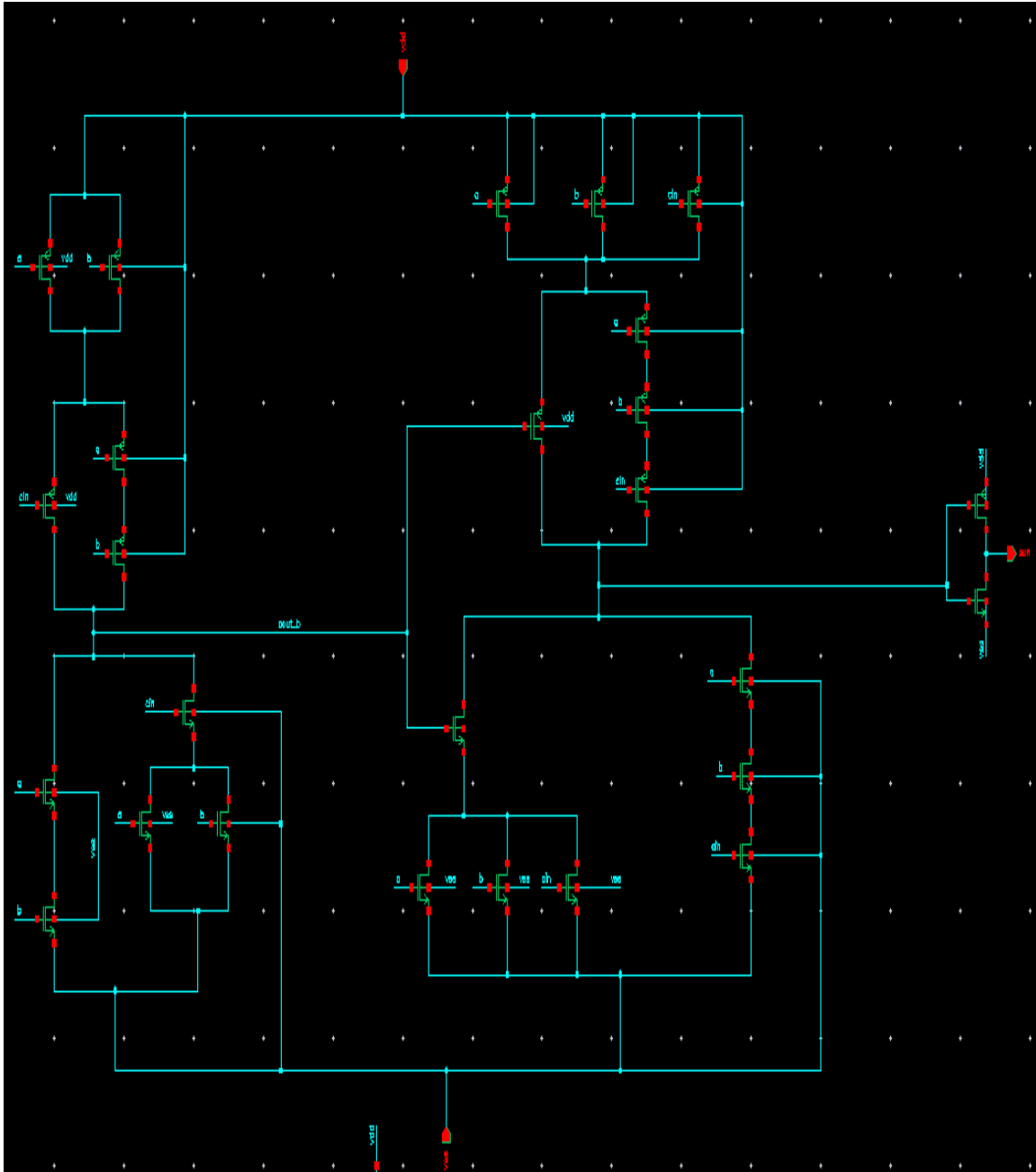
### Waveforms of NOR Gate:



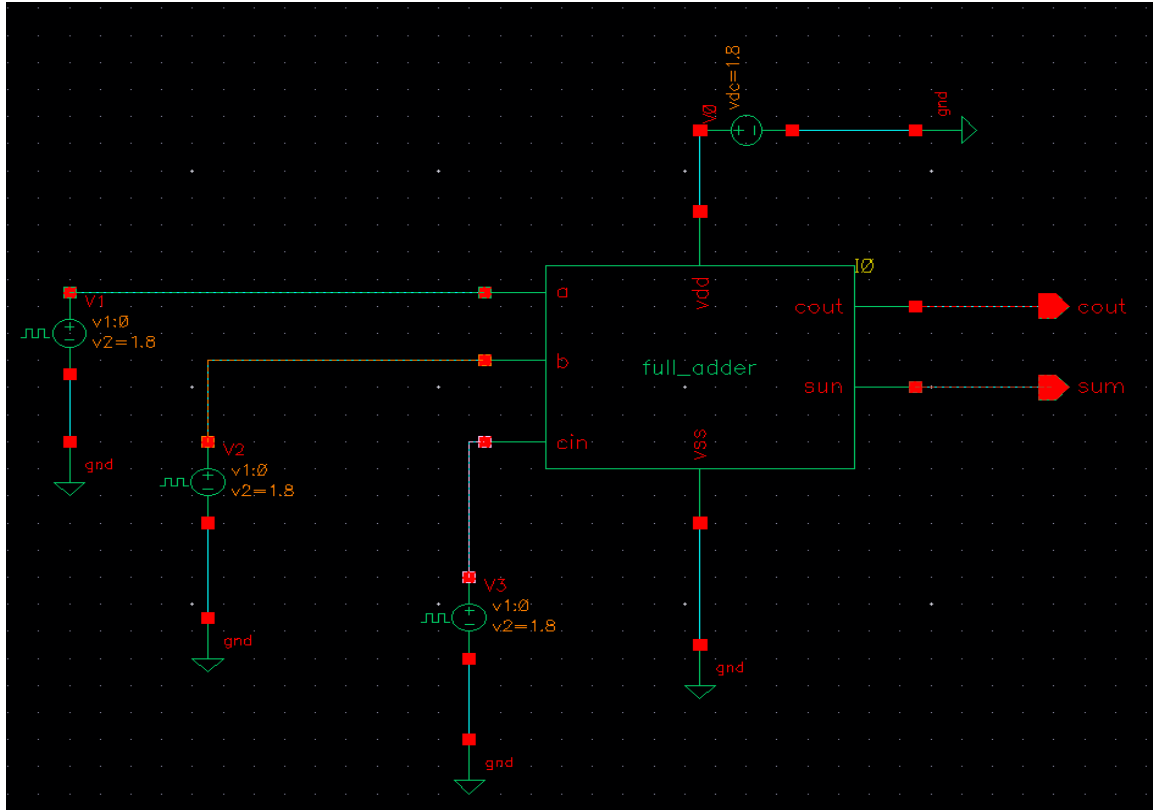
### Results:

**EXPERIMENT 9: DC, Transient analysis of CMOS full adder schematic**

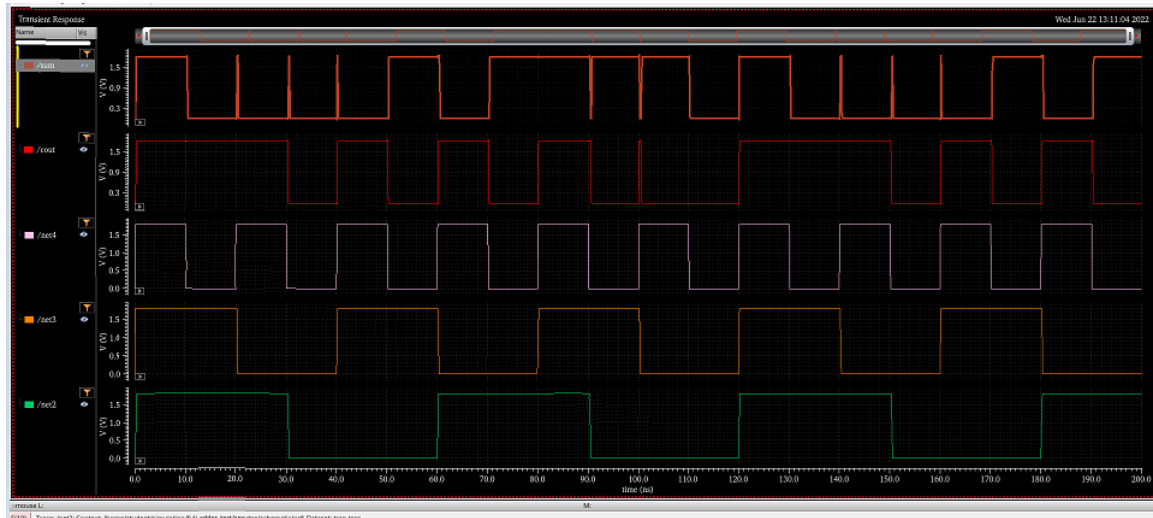
**Full Adder Schematic**



### Test Schematic Full Adder:



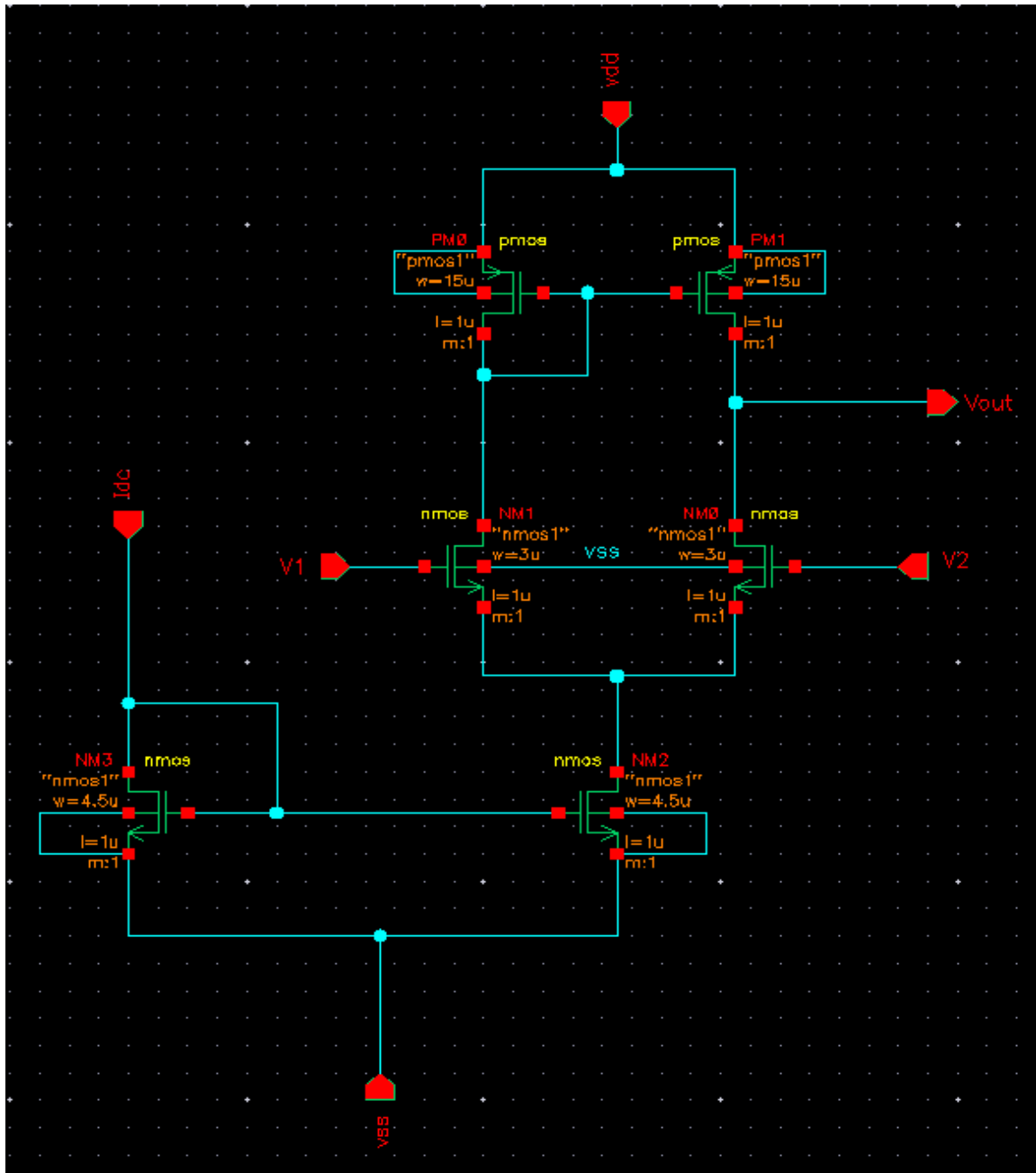
### Waveforms Full adder:



### Results:

**EXPERIMENT 12: DRC and LVS analysis of Differential Amplifier Layout**

## Schematic Capture



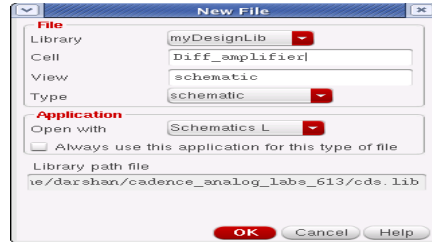
## Schematic Entry

**Objective: To create a new cell view and build Differential Amplifier**

## Creating a Schematic cellview

Open a new schematic window in the **myDesignLib** library 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.

3.	Library name	Cell Name	Properties/Comments	After
	gpd180	nmos	Model Name = nmos1 (NM0, NM1) ; W= 3u ; L= 1u	
	gpd180	nmos	Model Name =nmos1 (NM2, NM3) ; W= 4.5u ; L= 1u	
	gpd180	pmos	Model Name =pmos1 (PM0, PM1); W= 15u ; L= 1u	

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/inputoutput** when placing the **input/output/inout** pins 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

### Objective: To create a symbol for the Differential Amplifier

---

1. In the Differential Amplifier schematic window, execute **Create — Cellview— From Cellview**.

The **Cellview from Cellview** form 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.

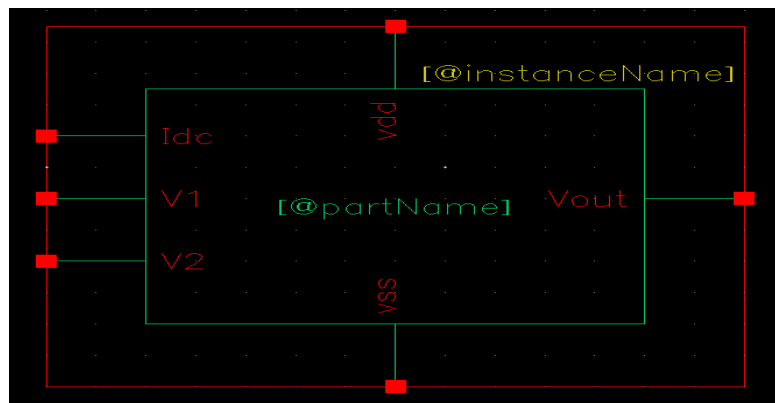
4. Modify the **Pin Specifications** as in the below symbol.

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.



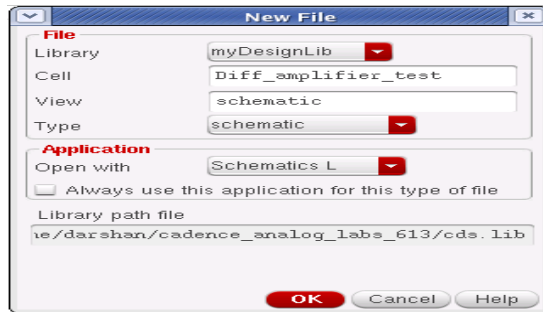
9. After creating symbol, click on the **save** icon in the symbol editor window to save the symbol. In the symbol editor, execute **File— Close** to close the symbol view window.

## Building the Diff\_amplifier\_test Design

**Objective: To build Differential Amplifier Test circuit using your Differential Amplifier**

## 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\_test Circuit

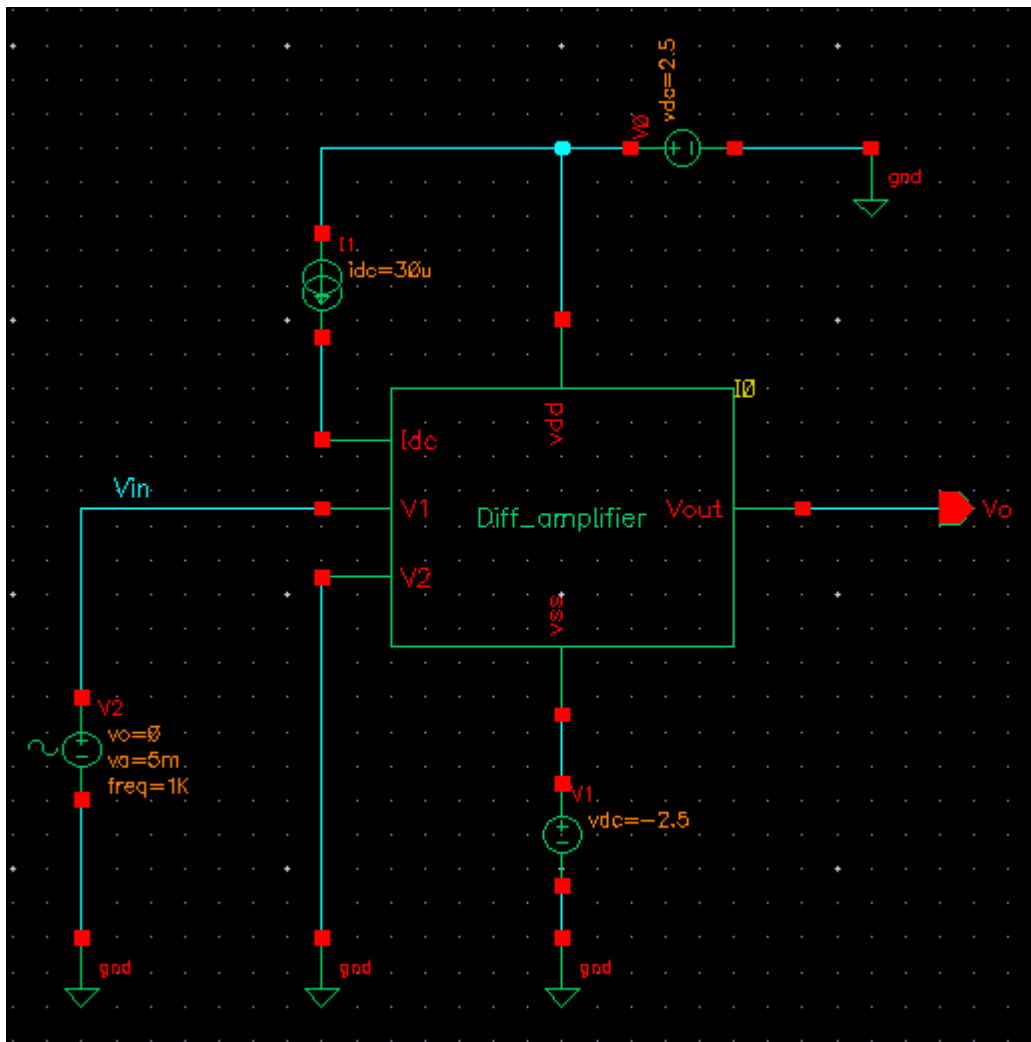
1. Using the component list and Properties/Comments in this table, build the Diff\_amplifier\_test schematic.

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	ldc	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

**Objective:** To set up and run simulations on the Differential Amplifier Test design.

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**.

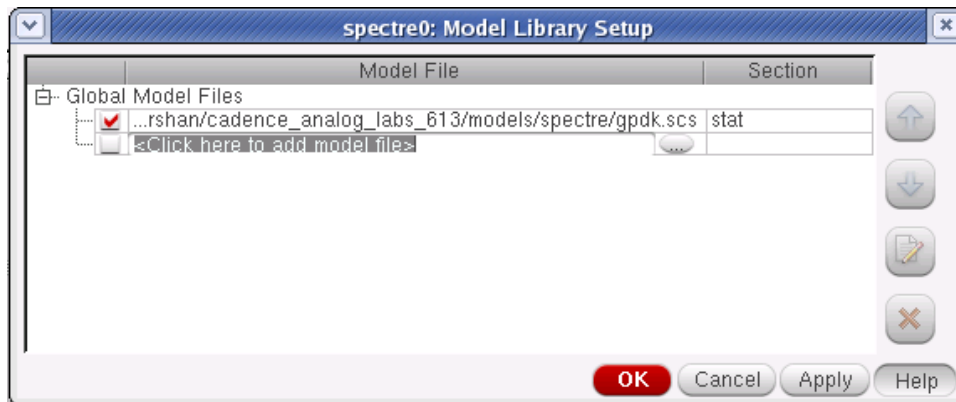
## Setting the Model Libraries

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 **gpd180.scs** file in the **./models/spectre** directory.

Select **stat** in Section field, click **Add** and click **OK**.



## Choosing Analyses

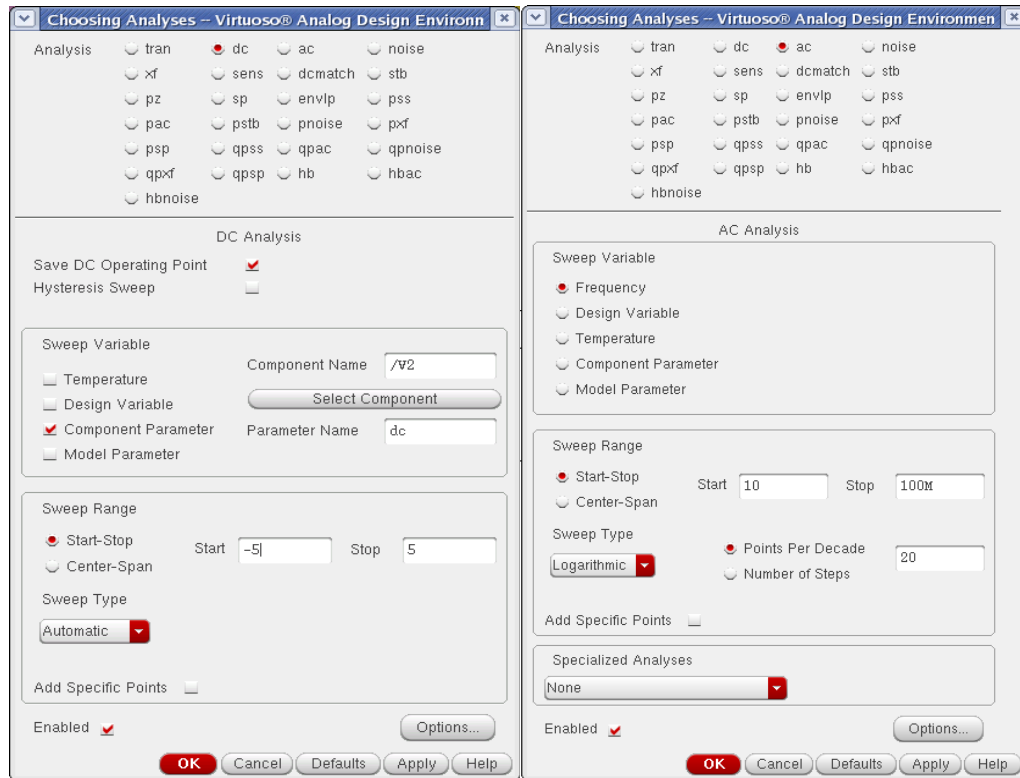
1. In the Simulation window, click the **Choose - Analyses** icon. You can also execute **Analyses - Choose**.

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 **Enabled** button at the bottom, and then click **Apply**.

## 3. To set up for DC Analyses:

- In the Analyses section, select **dc**.
- In the DC Analyses section, turn **on** Save DC Operating Point.
- Turn on the **Component Parameter**
- Double click the **Select Component**, Which takes you to the schematic window.
- Select input signal **Vsin** for dc analysis.
- In the analysis form, select **start** and **stop** voltages as **-5** to **5** respectively.
- Check the enable button and then click **Apply**.



## 4. To set up for AC Analyses form is shown in the previous page.

- In the Analyses section, select **ac**.
- In the AC Analyses section, turn on **Frequency**.
- In the Sweep Range section select **start** and **stop** frequencies as **150** to **100M**
- Select Points per Decade as **20**.
- Check the enable button and then click **Apply**.

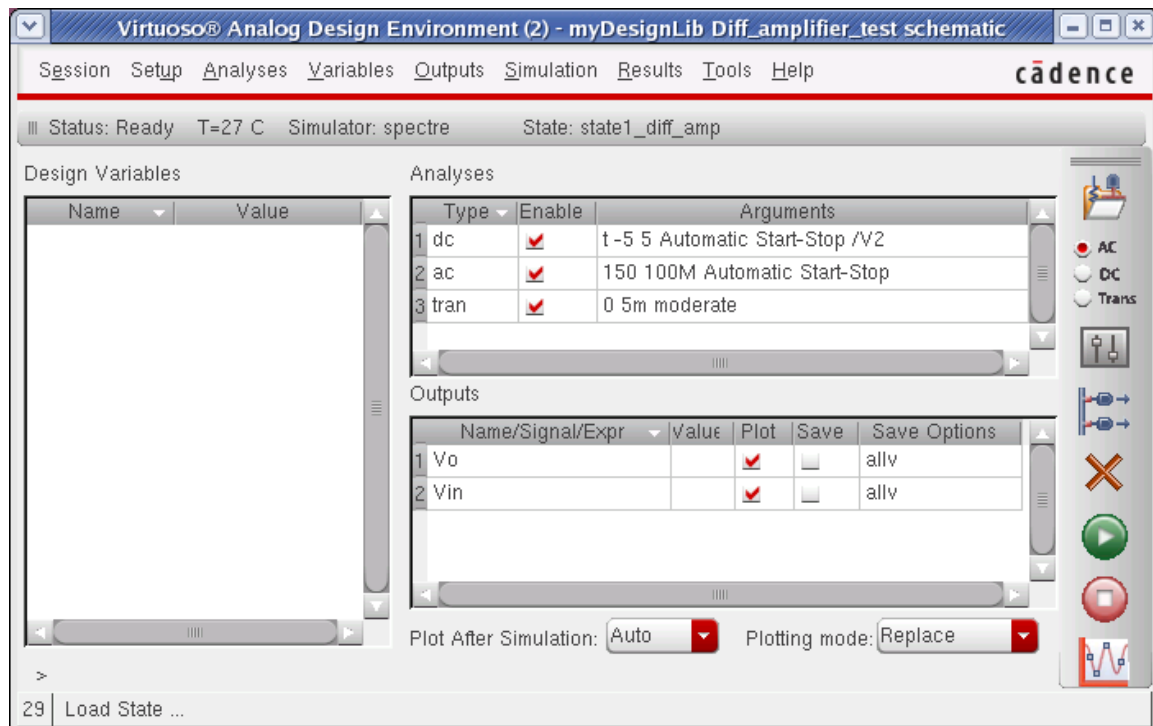
5. Click **OK** in the Choosing Analyses Form.

## Selecting Outputs for Plotting

Select the nodes to plot when simulation is finished.

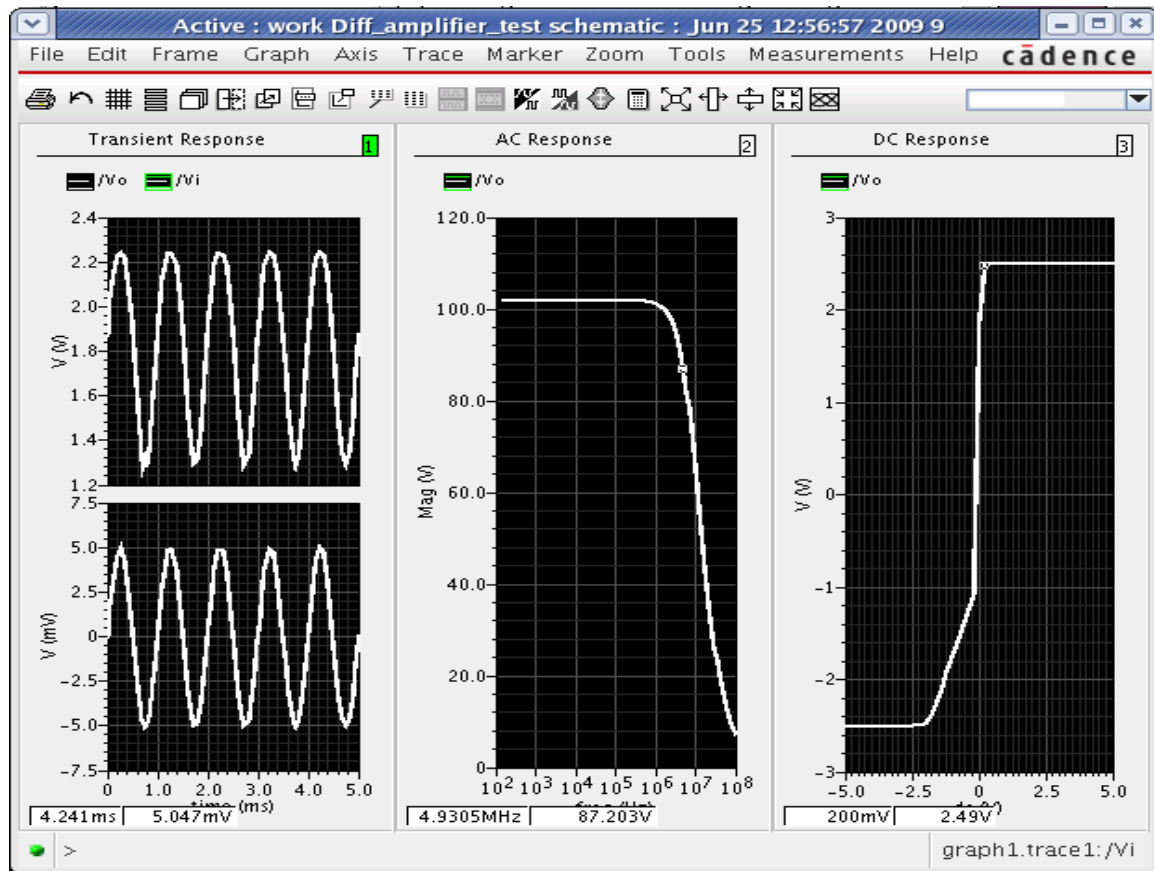
1. Execute **Outputs – To be plotted – Select on Schematic** in the simulation window.
2. Follow the prompt at the bottom of the schematic window, Click on output net **Vo**, input net **Vin** of the Diff\_amplifier. Press **ESC** with the cursor in the schematic after selecting node.

Does the simulation window look like this?



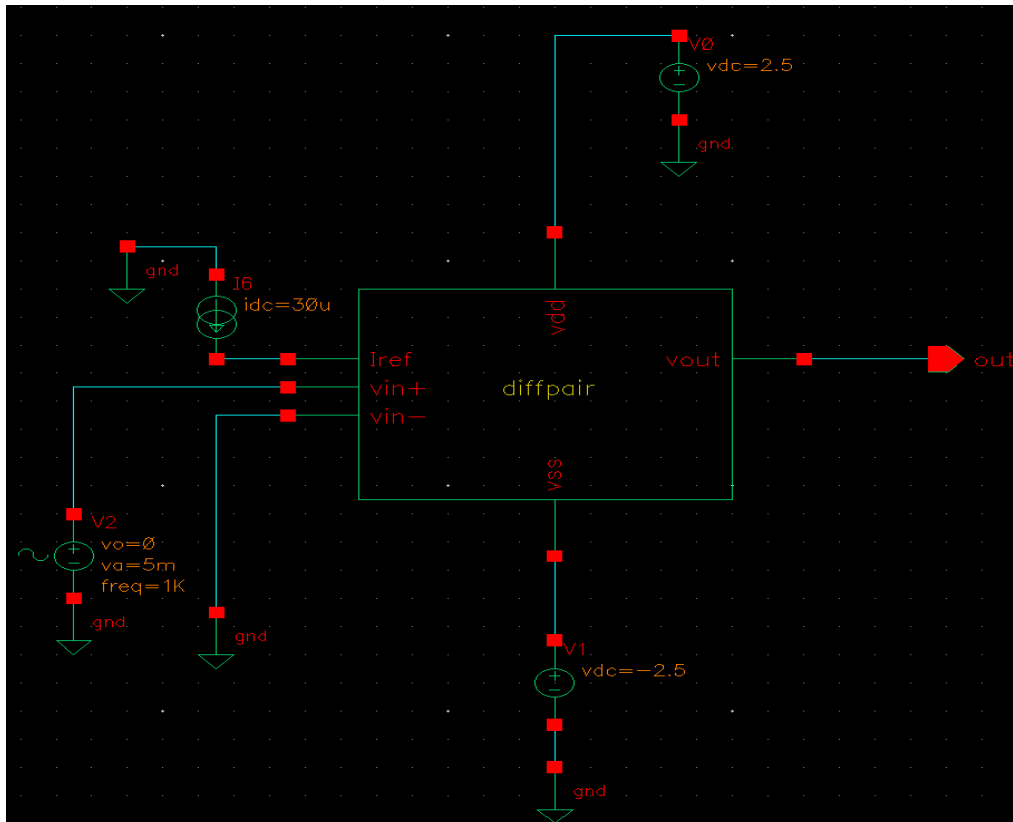
## Running the Simulation

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.

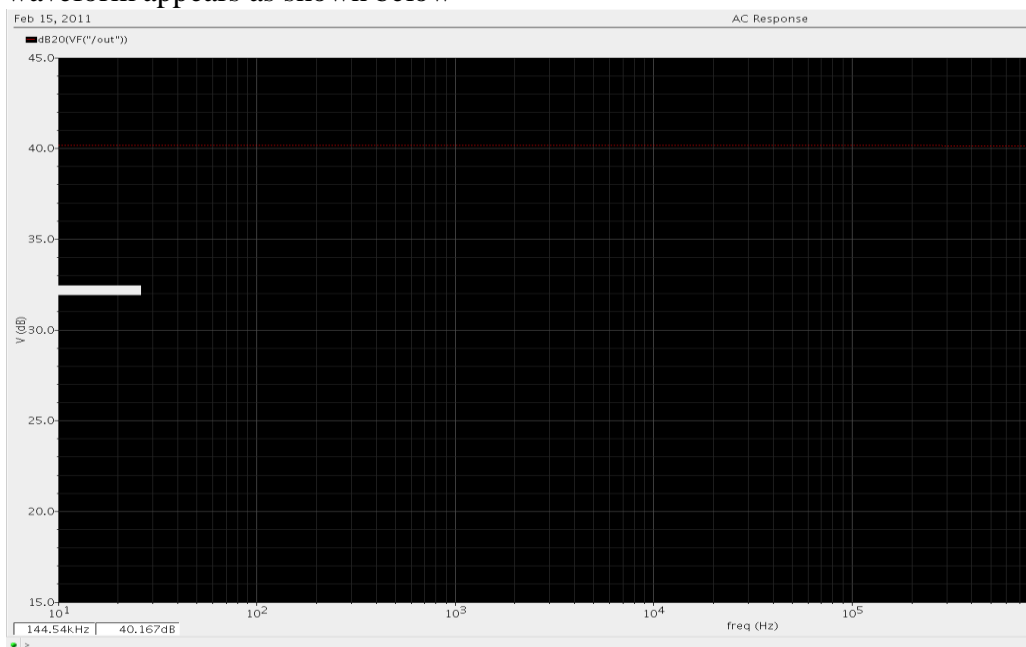


### To Calculate the gain of Differential pair:

Configure the Differential pair schematic as shown below –

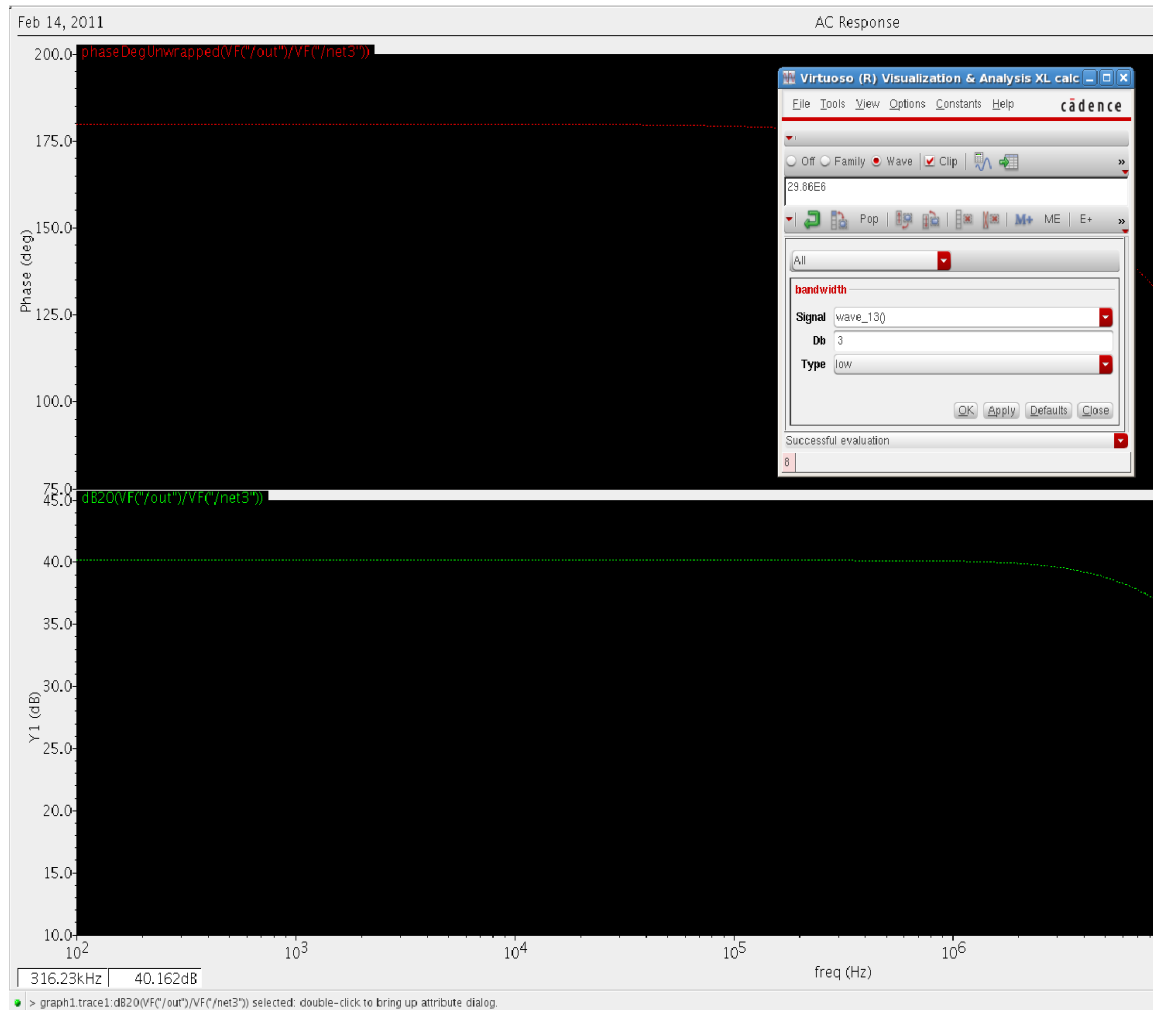


Now, open the ADE L, from **LAUNCH** → **ADE L**, choose the analysis set the ac response and run the simulation, from **Simulation** → **Run**. Next go to **Results** → **Direct plot** select AC dB20 and output from the schematic and press escape. The following waveform appears as shown below –



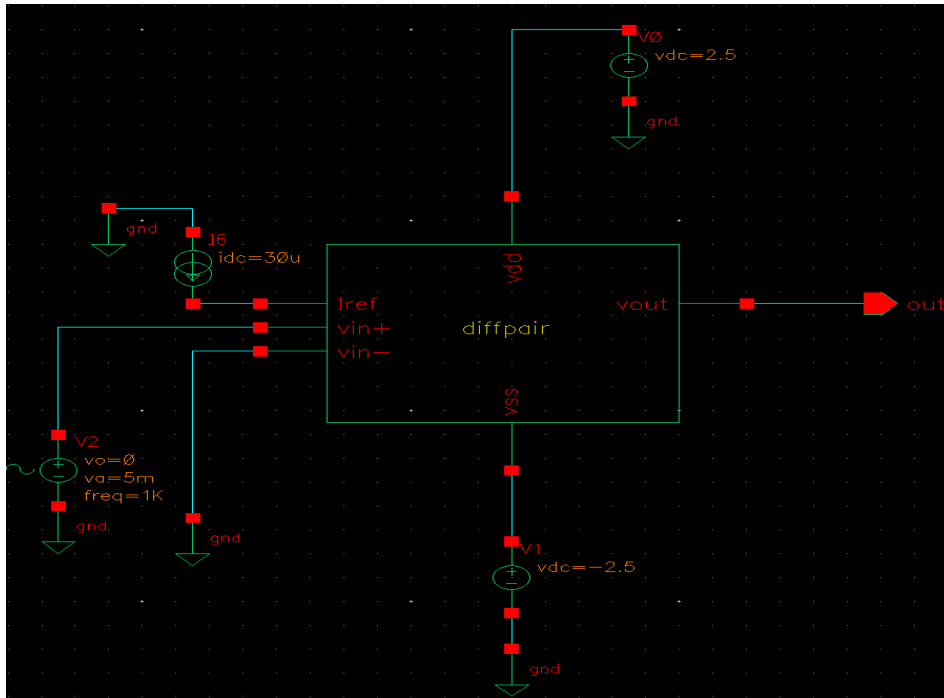
## To Calculate the BW of the Differential pair :

Open the calculator and select the bandwidth option, select the waveform of the gain in dB and press Evaluate the buffer -

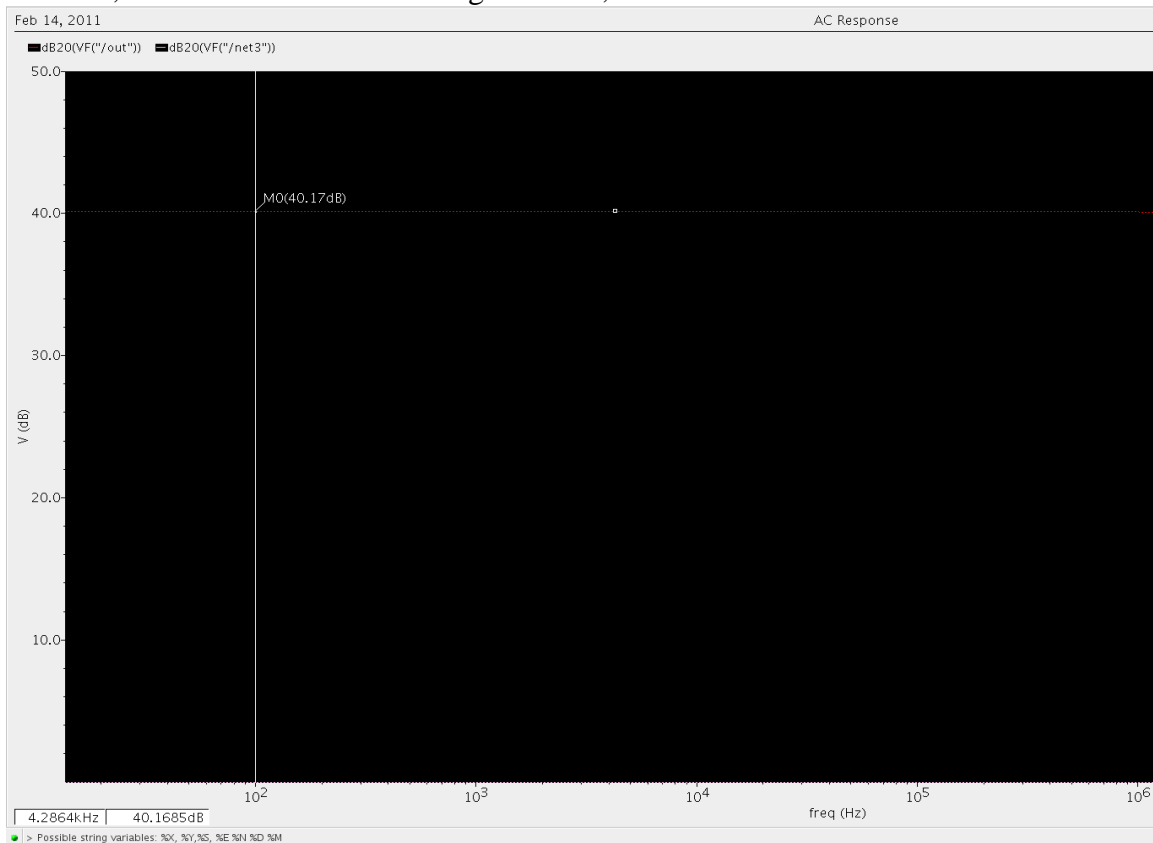


## To Calculate the CMRR of the Differential pair :

Configure the Differential pair schematic to calculate the differential gain as shown below –

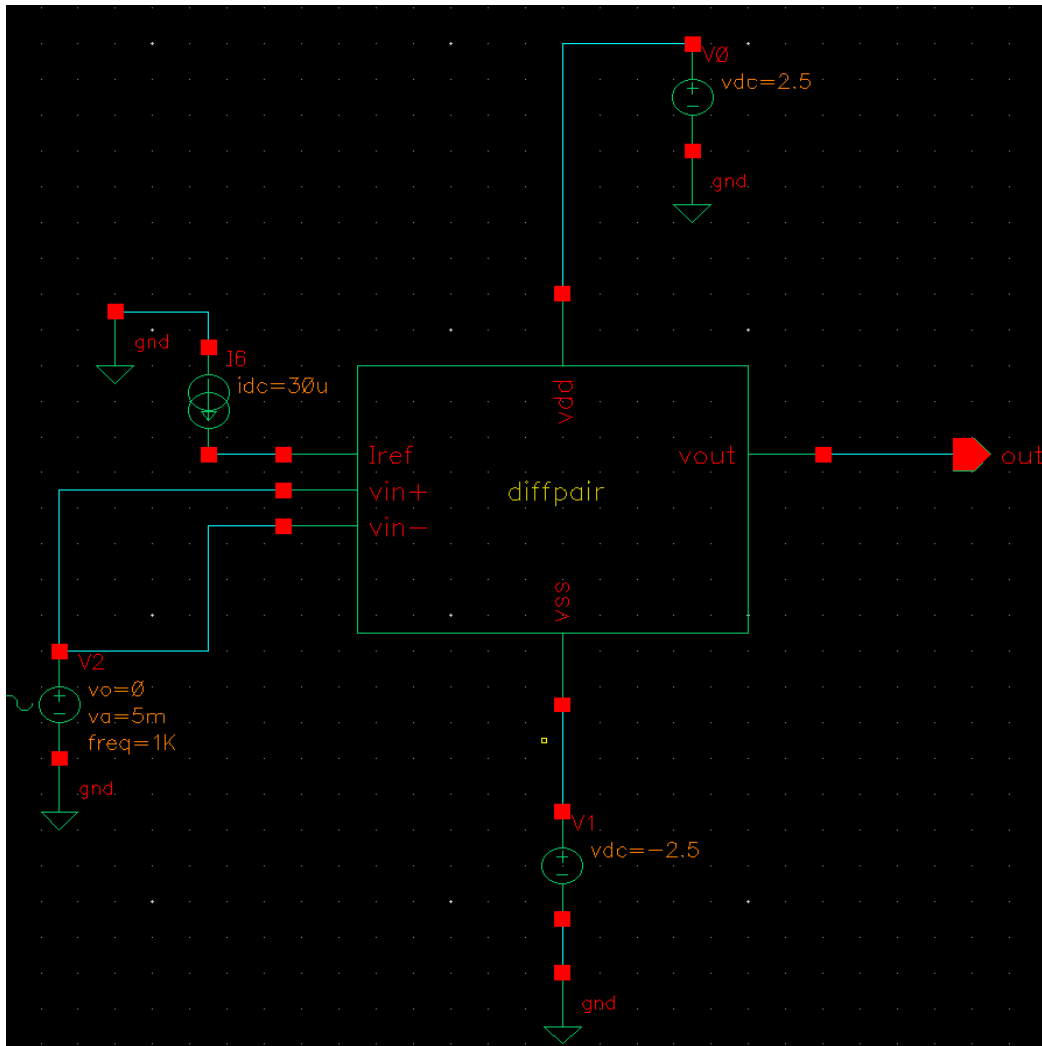


In the ADE L, plot the ac response with gain in dB. Measure the gain at 100hz and at 100Mhz, note down the value of the gain in dB, as shown below –

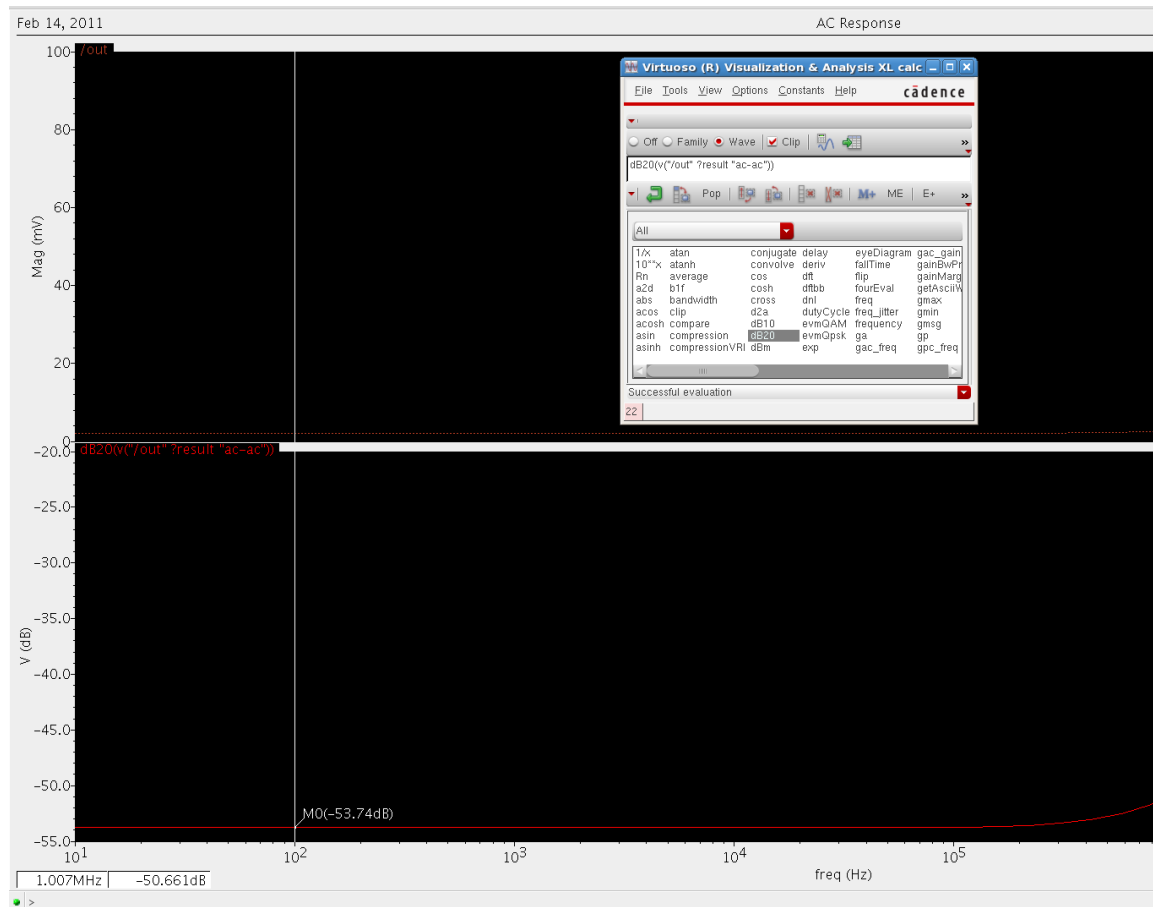




Configure the Differential pair schematic to calculate the common-mode gain as shown below –



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 =  $\left| \frac{A_d}{A_c} \right|$ , add the gains in dB i.e.,  $A_d - (-A_c)$ . For the output impedance note down the output resistance of the pmos and nmos transistors at the output side, and use the necessary equation like  $r_{01} \parallel r_{02}$ .

## 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\_amplicifier 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

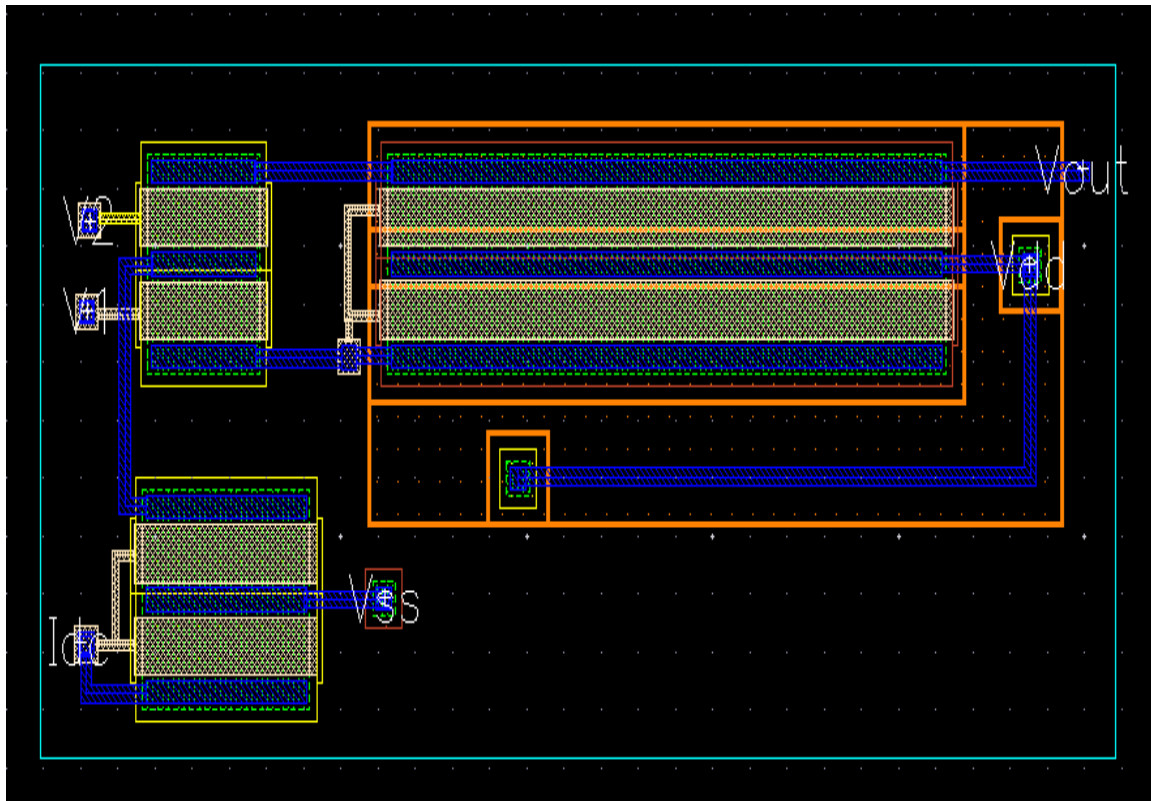
You will use the contacts or vias to make connections between two different layers.

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

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

## Saving the design

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



## Physical Verification Assura DRC

### Running a DRC

1. Open the Differential\_Amplifier layout from the CIW or library manger if you have closed that. Press **shift – 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 **gpd180**. This automatically loads the rule file.

3. Click **OK** to start DRC.

4. A Progress form will appear. 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.

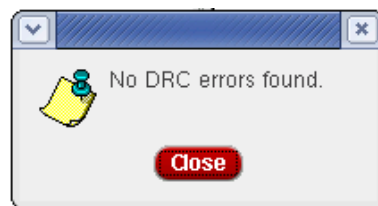


6. If there any DRC error exists in the design **View Layer Window (VLW)** and **Error Layer Window (ELW)** appears. Also the errors highlight in the design itself.

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 found** written 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.

**Run Assura LVS**

Schematic Design Source: **DFII** Use Existing Netlist ☐ Netlisting Options...

Library: **myDesignLib** Cell: **Diff\_amplifier** View: **schematic** Browse...

Layout Design Source: **DFII** Use Existing Extracted Netlist ☐

Library: **myDesignLib** Cell: **Diff\_amplifier** View: **layout** Browse...

Run Name: Run Directory: **./LVS** ...

Run Location: **local**

View Rules Files ☒ Technology: **gpd180** Rule Set: **default**

☐ Extract Rules: **.../cadence\_analog\_labs\_613/pv/assura/extract.rul** View... Reload

☐ Compare Rules: **.../cadence\_analog\_labs\_613/pv/assura/compare.rul** View...

Switch Names: Set Switches

☐ Binding File(s): View...

☐ RSF Include: View...

Variable	Value	Default	Description
<b>None</b>			

View avParameters ☐ Modify avParameters... 7 avParameters are set.

View avCompareRules ☐ Modify avCompareRules... 1 avCompare rule is set.

View Additional Functions ☐ No additional functions are set.

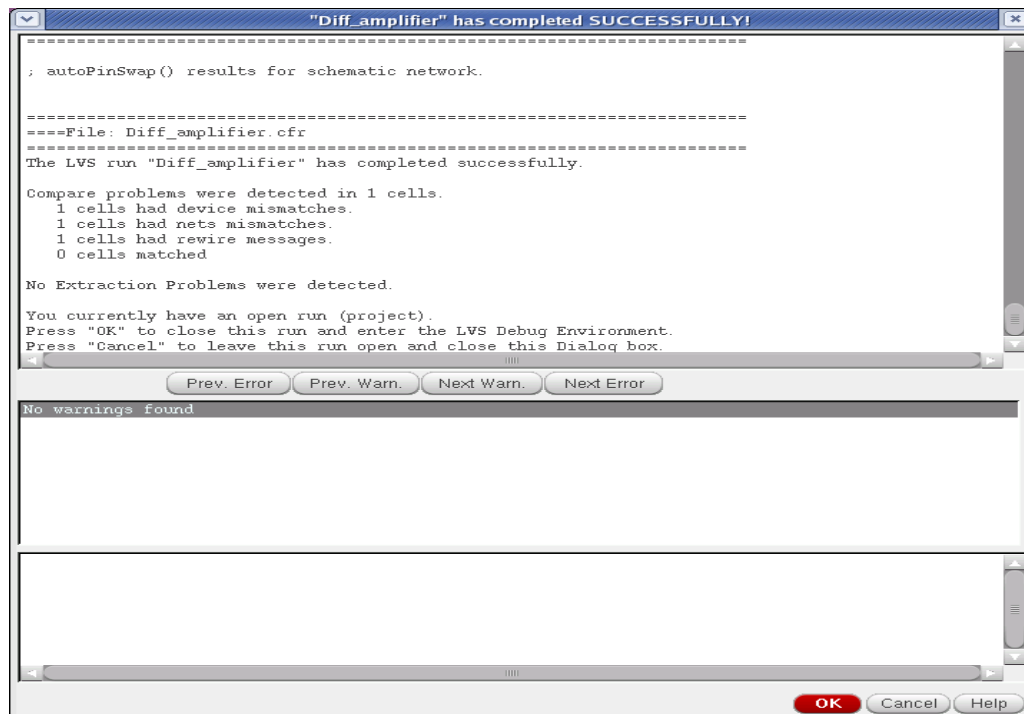
**OK** Cancel Apply Defaults Load State Save State View RSF Help

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 backannotated 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.

The screenshot shows the 'QRC (Assura) Parasitic Extraction Run Form' with the 'Setup' tab selected. The form contains various configuration options for parasitic extraction.

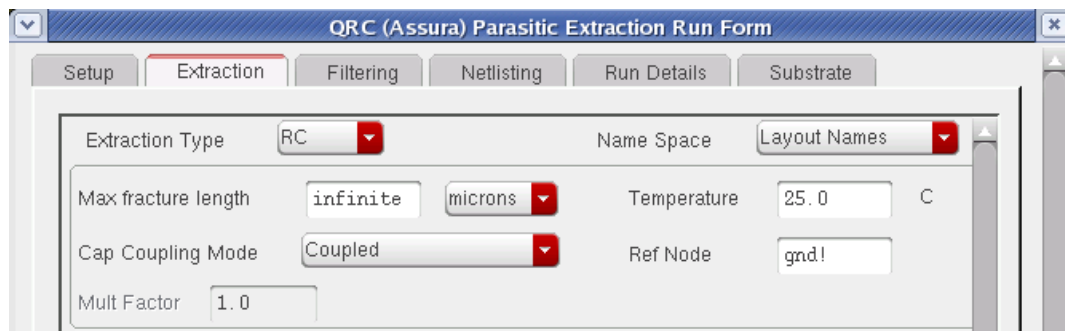
**Technology:** gpd180  
**RuleSet:** default  
**p2lvsSet:** NONE  
**UseMultRuleSets:** ☐  
**Setup Dir:** port/home/darshan/cadence\_analog\_labs\_613/pv/assura,  
**RSF Include:**  **View** **Edit**  
**Rule RSF Include:**  **View** **Edit**  
**Tech Cmd File:** Default  **View** **Edit**

**Output:** Extracted View  
**Lib:** DesignLib **Cell:** amplifier **View:** av\_extracted  
**Enable CellView Check:** ☐  
**Parasitic Res Component:** presistor **Prop Id:** r  
**Parasitic Cap Component:** pcapacitor **Prop Id:** c  
**Parasitic Ind Component:** pinductor **Prop Id:** l  
**Parasitic M Component:** pmind **Prop Id:** k  
**Inductance L1 Prop Id:** ind1 **Inductance L2 Prop Id:** ind2  
**Call Procedure:**   
**Substrate Extract:** ☐ **Extract MOS Diffusion Res:** ☒  
**Extract MOS Diffusion AP:** ☒ **Extract MOS Diffusion High:** NONE  
**Substrate Profile:** NONE  
**Library Prefix:**   
**Library Directory:**

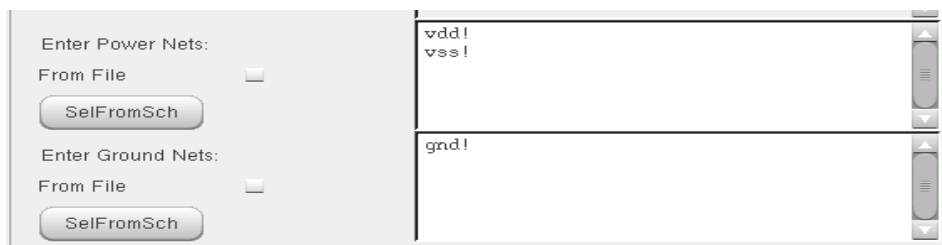
**Buttons:** OK, Cancel, Defaults, Apply, Load State, Save State, ViewRSF, Help

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!** and **Enter 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\_extracted** view from the library manager and view the parasitic.

**Results:**



## VIVA QUESTIONS:

1. What is VLSI

### **VERY LARGE SCALE INTEGRATED CIRCUIT**

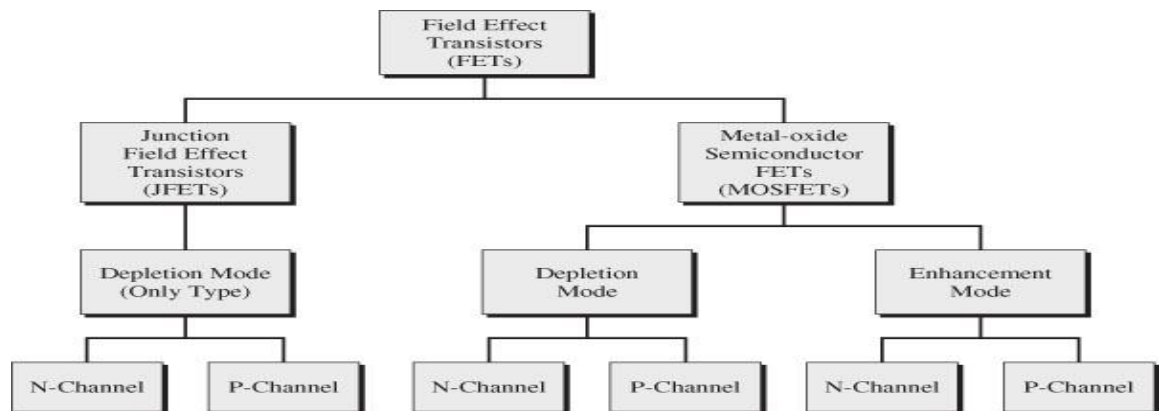
Technique where many circuit components and the wiring that connects them are manufactured simultaneously into a compact, reliable and inexpensive chip.

2. Mention the advantages of VLSI

Advantages:

- Fast, cheap, low power transistors
- High integration, low cost.

3. How are FETS classified?



4. Differentiate between BJT and FET

#### **BJT**

- Current controlled devices.
- Current flows due to both majority & minority charge carriers & hence called Bipolar junction transistors.
- Types: NPN, PNP.
- I/P resistance is less.
- Bigger in size.
- Less thermal stability.
- Relation b/w i/p & o/p is linear.

#### **FET**

- Voltage controlled devices.
- Current flows due to majority charge carriers & hence called unipolar transistors.

- Types: N-channel, P-channel.
- Compared to BJT I/P resistance is high.
- Smaller in construction than BJT thus useful in IC's.
- More thermal stability.
- Relation b/w i/p & o/p is non-linear.

## 5. Differentiate depletion and enhancement transistors.

### Depletion

- The channel Pre exists.
- It can be operated in depletion mode As well as enhancement mode.
- Drain current flows on application of Drain to source voltage at  $V_{gs} = 0V$ .
- Ex. JFET, MOSFET.

### Enhancement

- Channel is physically absent & is induced by applying gate voltage above the threshold voltage.
- Can be operated in only enhancement mode.
- Practically no current flows on applying  $V_{ds}$  at  $V_{gs} = 0V$ . Current flows only when  $V_{gs}$  is above the threshold level.
- Ex. MOSFET.

## 6. Explain CMOS

One of the most popular MOSFET technologies available today is the complementary MOS, or CMOS, technology. This technology makes use of both P and N channel devices in the same substrate material. Such devices are extremely useful, since the *same* signal which turns *on* a transistor of one type is used to turn *off* a transistor of the other type. This allows the design of logic devices using *only* simple switches, without the need for a pull-up resistor.

## 7. Explain BiCmos

BiCMOS technology is a combination of **B**ipolar and **C**MOS technology. CMOS technology offers less power dissipation, smaller noise margins, and higher packing density. Bipolar technology, on the other hand, ensures high switching and I/O speed and good noise performance. It follows that BiCMOS technology accomplishes both - improved speed over CMOS and lower power dissipation than bipolar technology. The main drawback of BiCMOS technology is the higher costs due to the added process complexity. Impurity profiles have to be optimized to both NPN and CMOS issues.

## 8. Mention different types of CMOS fabrication

The different types of fabrication are N well, P well and Twin Well.

9. Mention the advantages of BiCmos

1. BiCMOS is inherently robust with respect to temperature and process variations, resulting in less variability in final electrical parameters, resulting in higher yield, an important economic consideration.
2. Large circuits can impose severe performance penalties due to simultaneously switching noise, internal clock skews and high nodal capacitances in critical paths - BiCMOS has demonstrated superiority over CMOS in all of these factors.
3. BiCMOS can take advantage of any advances in CMOS and/or bipolar technology, greatly accelerating the learning curve normally associated with new technologies.

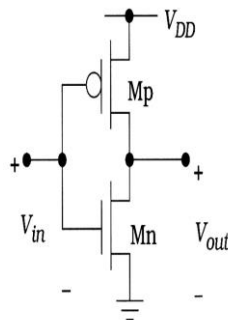
10. Mention the disadvantages of BiCmos

Greater process complexity compared to CMOS

11. What is threshold voltage

Threshold voltage is defined as the minimum voltage that is required for making the transistor ON. Transistor may be either NMOS or PMOS. For NMOS the value of threshold voltage is positive value and for PMOS the value of threshold voltage is negative value.

12. Draw the circuit of CMOS inverter



13. Differentiate between transmission gate and pass transistor

Pass transistors produce degraded outputs

Transmission gates pass both 0 and 1 well

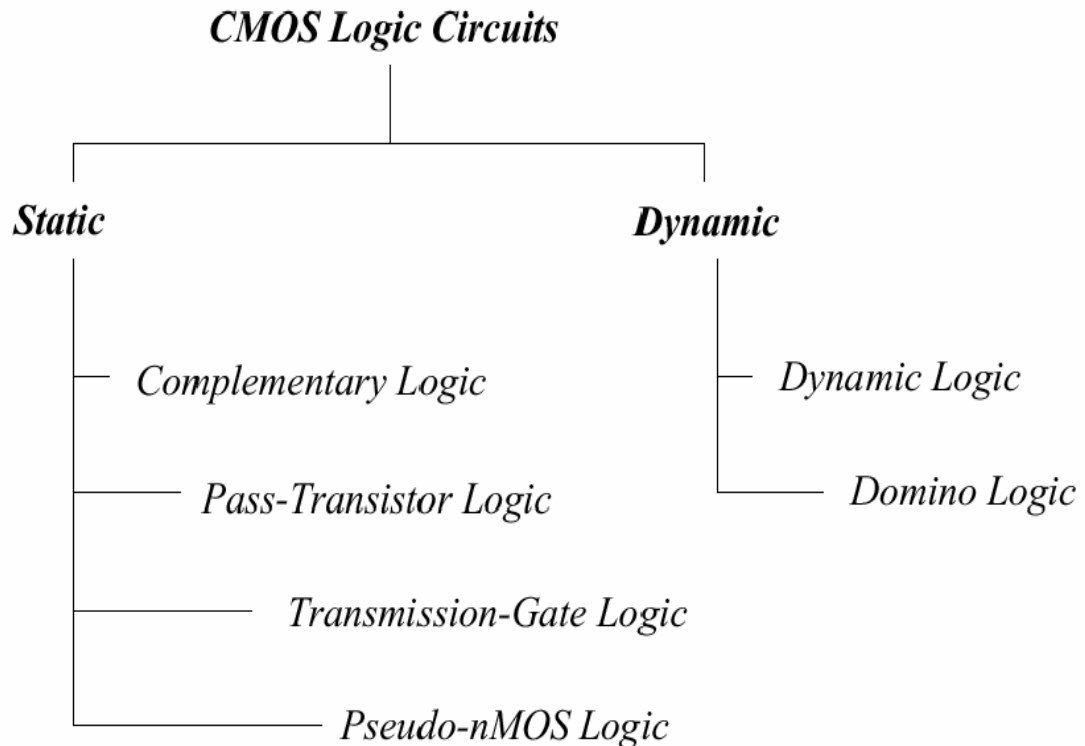
14. Differentiate between tristate inverter and buffer

Tristate buffer produces Z when not enabled. Tristate inverter produces restored output and Violates conduction complement rule because we want a Z output.

15. What is lambda based design rule?

One lambda = one half of the "minimum" mask dimension, typically the length of a transistor channel. Usually all edges must be "on grid", e.g., in the MOSIS scalable rules, all edges must be on a lambda grid.

16. How is Cmos logic structure classified



Different structures of CMOS logic circuits.

17. Mention the advantages of usage of pmos and nmos transistors in CMOS.

Pmos passes full logic 1 and nmos passes full logic 0

18. Mention the advantages and disadvantages of cmos.

Advantages: passes full logic 1 and 0

Disadvantages: requires more number of transistors.

19. Explain pseudo nmos

Consists of grounded pmos and PDN.

20. Mention the advantages of pseudo nmos.

Transistor requirement reduces compared to cmos.

21. What are buried and butting contacts.

Buried contact: The contact cut is made down each layer to be joined.

Butting contact: The layers are butted together in such a way the two contact cuts become contiguous.

22. What is pwell cmos?

It has a nsubstrate and a pwell is etched in it inside the well nmos is formed and in the substrate pmos is formed.

23. What is nwell cmos?

It has a psubstrate and a nwell is etched in it inside the well pmos is formed and in the substrate nmos is formed.

24. What is twin tub cmos?

Both nwell and pwell is formed in the substrate

25. What is Latch up?

Condition of direct current flow from VDD to GND due to parasitic transistors formation.

26. What are the colours used for polysilicon, metal, contact cut in stick diagram encoding?

Red, blue, black.

27. Mention the advantages of using HDL.

Need not rig up huge logic circuits.

28. Differentiate between VHDL and Verilog.

<b>VHDL</b>	<b>VERILOG</b>
Case insensitive	Case sensitive
Libraries are to included	Libraries are included when synthesized.
Entity architecture pair is used	Module is used
Has no built in constructs	Has built in constructs

29. What are the different styles of VERILOG modeling.

Data flow, structural, behavioral, switch level

30.What is the advantage of switch level description.

Pmos and nmos switches can be used.

31.Differentiate between always and initial statements.

Always is used along with sensitivity list. Initial is used to initialize a value.

32. Differentiate between always and process statements.

Always is used in verilog and process is used in VHDL.

33.Differentiate between mixed type and mixed language descriptions.

Mixed type uses different styles of description in a single program. Mixed language uses both Verilog and VHDL codes in a single program.