Gokaraju Rangaraju
Institute of Engineering and Technology

ECAD and VLSI

Laboratory

Manual & WorkBook

Name:			
Reg. No.:			
Branch:			
Class:		**Section:**	

GRIET - ECE
Focused Engineering

CERTIFICATE

This is to certify that it is a bonafide record of practical work done by Mr. /Ms._____, ***Reg. No._____in the*** **ECAD & VLSI LABORATORY** ***in_____semester of ____ year during 20____-20____.***

LAB Incharge

Table of Contents

1 Introduction

The objective of this lab is to learn and get acquainted with HDL designing using Xilinx Design Suite, FPGA programming using Altera Kits, spice simulations and layout using the Cadence Design Suite

1.1 LIST OF EXPERTIMENTS

1.1.1 ECAD Programs

PART1: The programs are designed and verified in Xilinx.

1. HDL code to realize all logic gates
2. Design of 2-to-4 decoder
3. Design of 8-to-3 encoder (without and with parity)
4. Design of 8-to-1 multiplexer
5. Design of 4bit binary to gray converter
6. Design of De-multiplexer, comparator
7. Design of full adder using 3 modeling styles
8. Design of flipflops: SR, D, JK, T
9. Design of 4 bit Binary BCD counters(synchronous, asynchronous reset) or any sequence counter
10. Finite state machine design

PART2: Programs are to be downloaded on the FPGA boards Cyclone2, Cyclone3 using Quartus II. The performance testing may be done using pattern generator and logic analyzer.

All the programs in the (PART1) should be programmed and configured on the FPGA board

1.1.2 VLSI Programs

PART1: Introduction to spice simulation

1. V-I curves of a NMOS and PMOS transistor.
2. Transfer characteristics of an CMOS Inverter
3. Design and simulation of a Latch circuit
4. Design and simulation of D-flipflop
5. Design and simulation of NOR/NAND gates
6. Design and simulation of CMOS 1-bit adder
7. Analog circuit simulation (AC analysis) – CS & CD amplifier

PART2: Introduction to Layout & design rules. Verification (DRC, LVS, ANTDRC) of the designs

1. Layout of CMOS inverter
2. Layout of any combinational circuit (complex cmos logic gate, NAND,NOR)
3. Layout of Latch
4. Layout of Adder

5. Layout of MUX

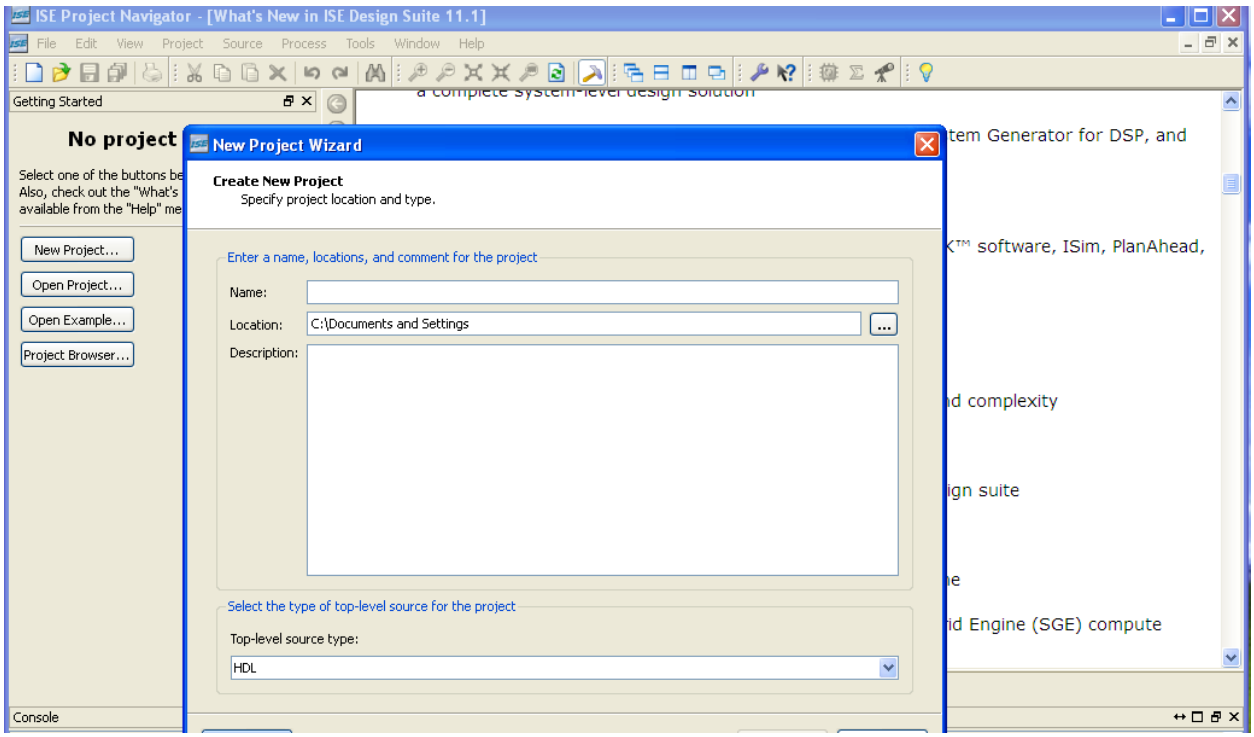
2 Xilinx DSI Manual

Xilinx ISE is a software tool produced by Xilinx for synthesis and analysis of HDL designs, which enables the developer to synthesize ("compile") their designs, perform timing analysis, examine RTL diagrams, simulate a design's reaction to different stimuli, and configure the target device with the programmer.

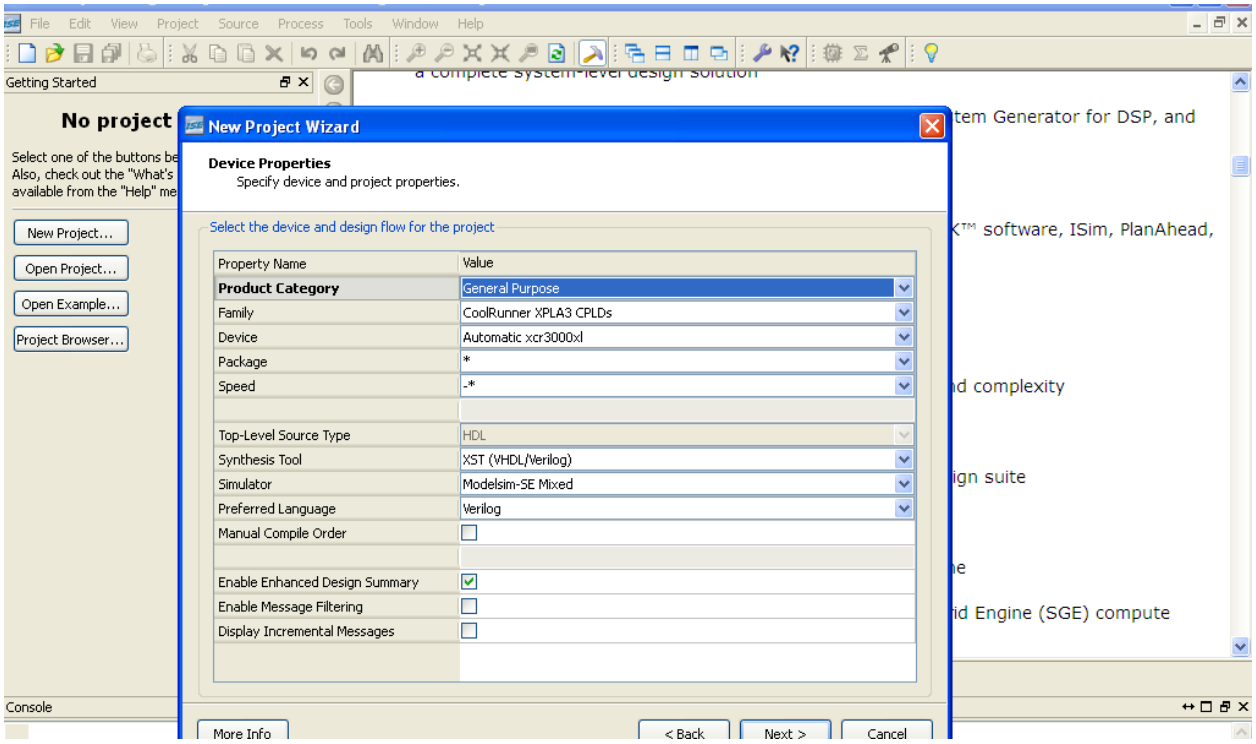
In our Lab, the scope is limited to design and analyze the design using test benches & simulation.
The following is the step by step procedure to design in the Xilinx ISE:

1. ***New Project Creation***

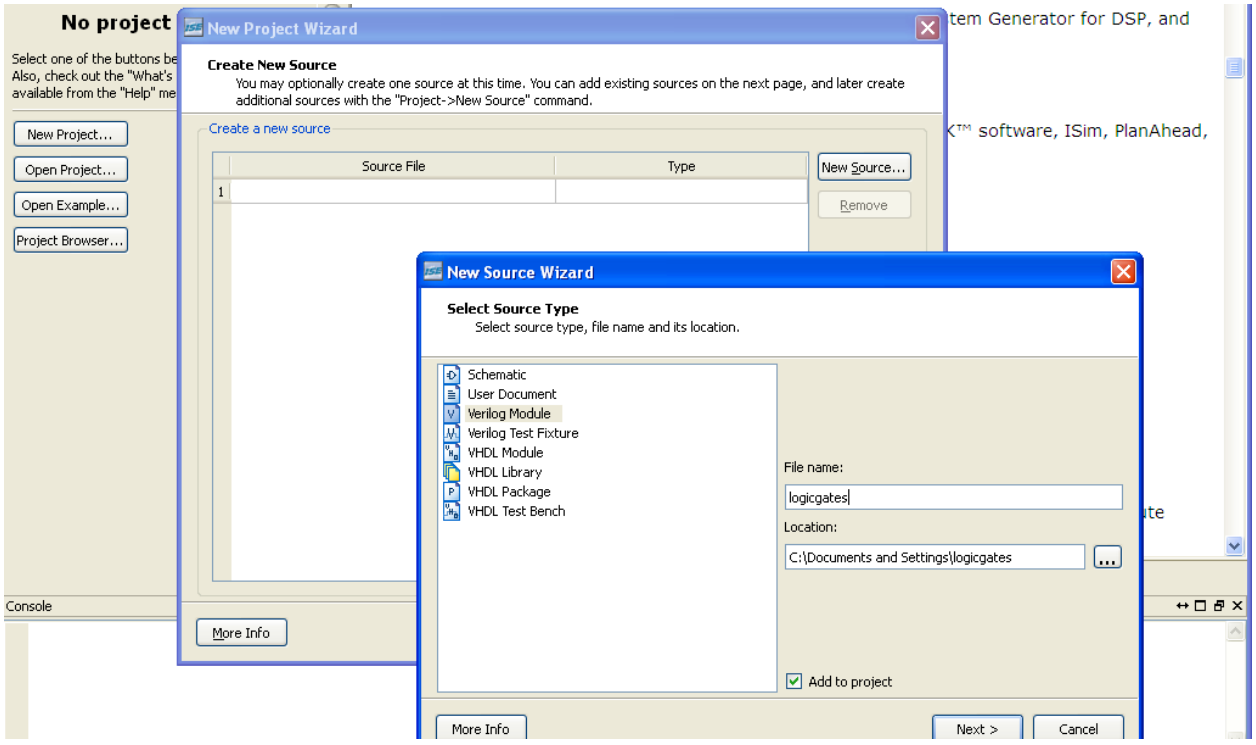
   Once the Xilinx ISE Design suite is started, open a new project & enter your design name and the location path. By default 'HDL' is selected as the top-level source type. (If not, please select Top-level source type as 'HDL')



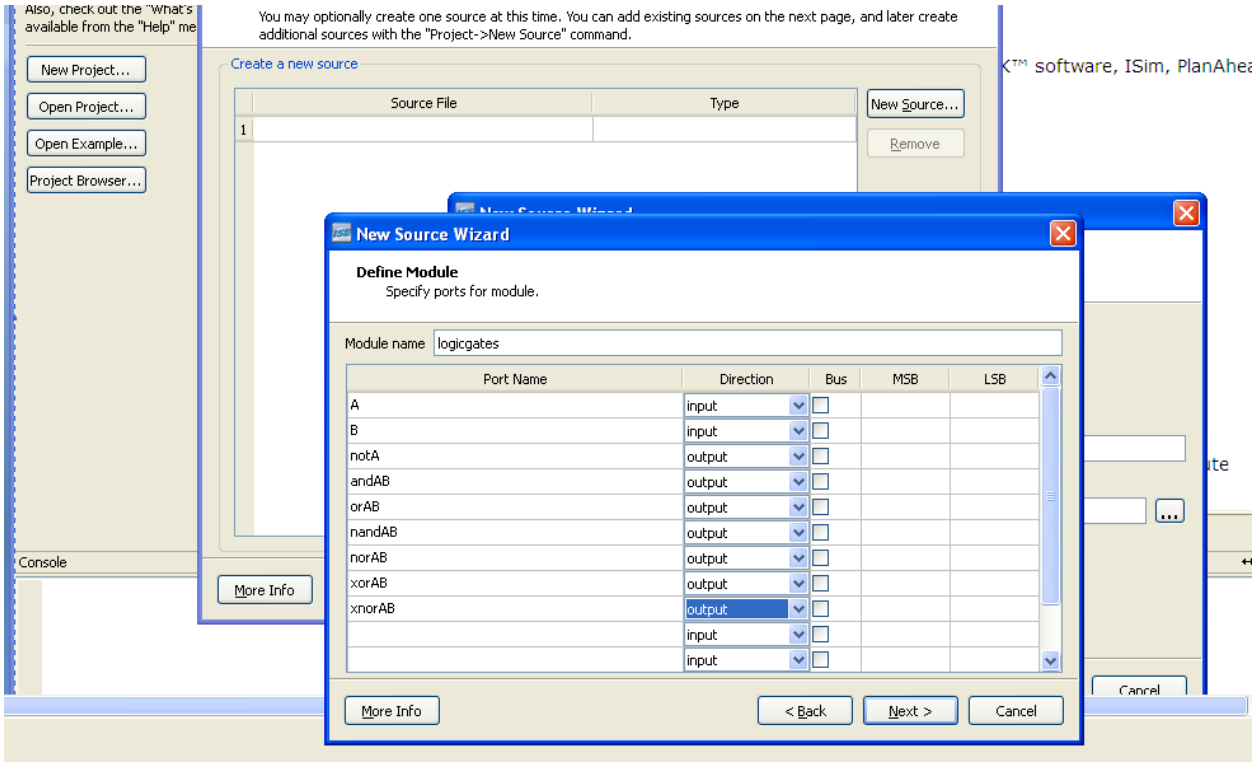
2. Continue to the next window and check if the Preferred Language is selected as ‘Verilog’



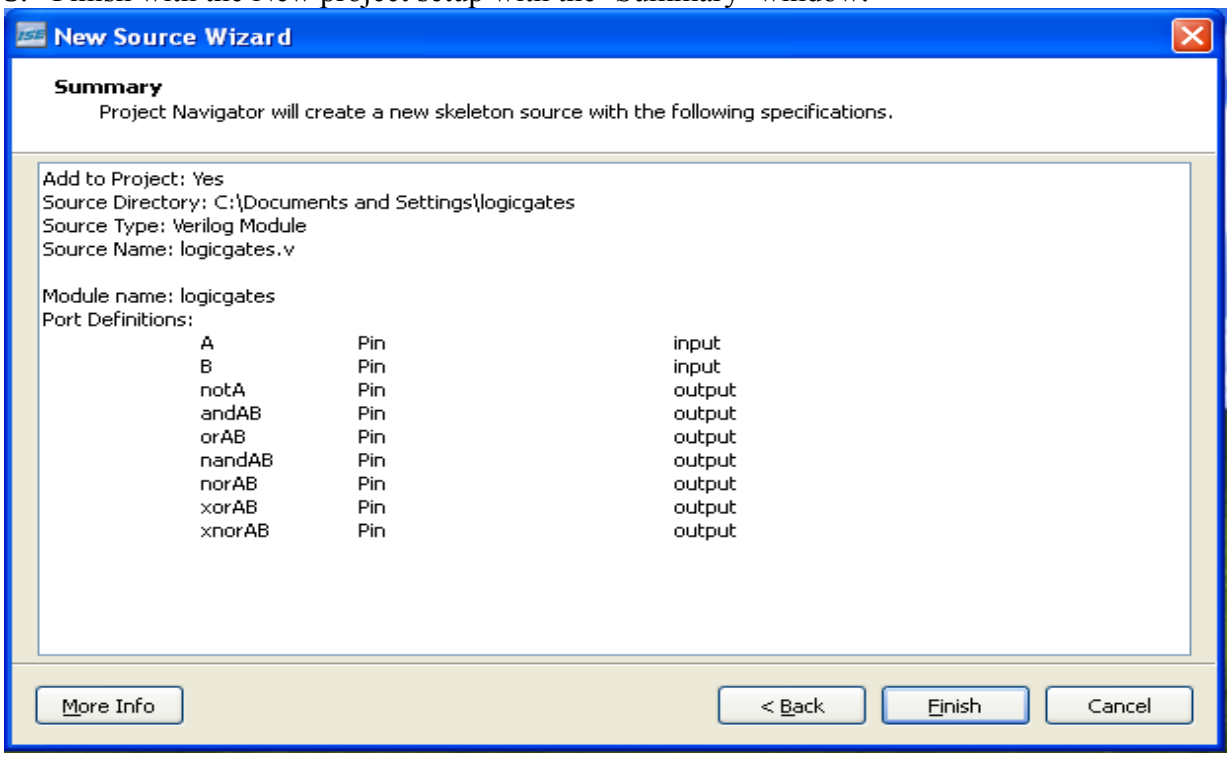
3. Proceed by clicking ‘Next’ and create a ‘New Source’ using the ‘Create New Source’ Window



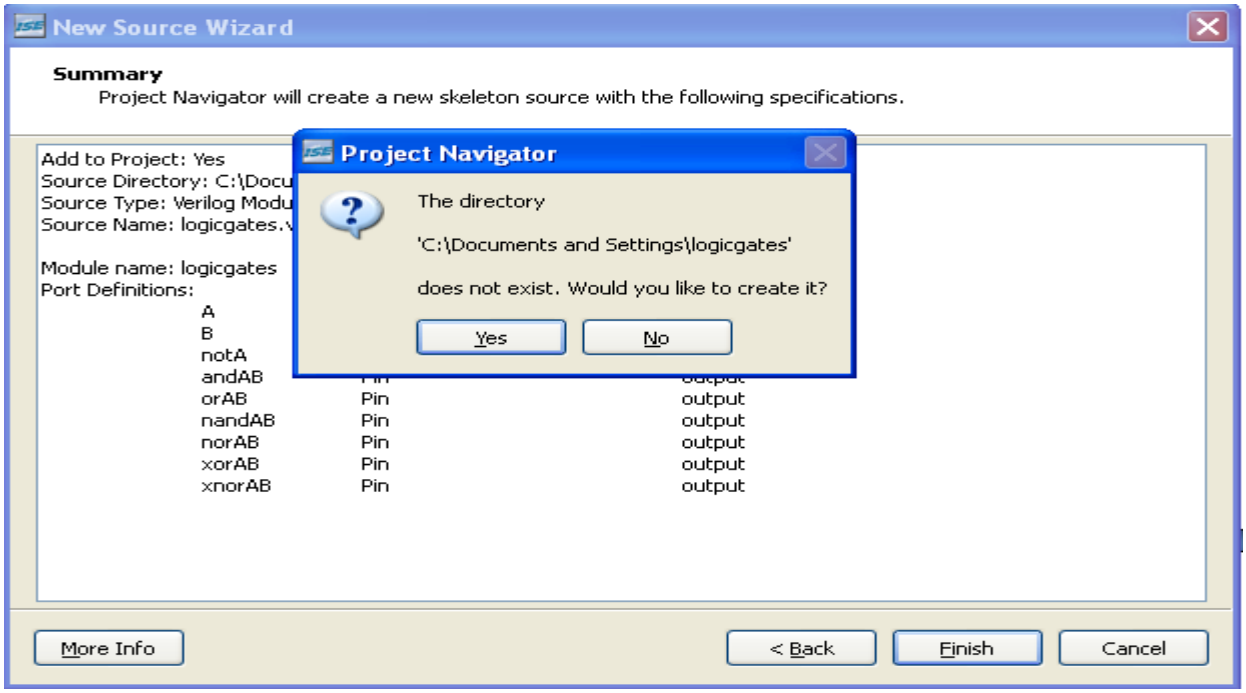
4. Select the source type as ‘Verilog Module’ and input a filename and proceed to ‘Next’. In the next window ‘Define Module’ enter the ports.



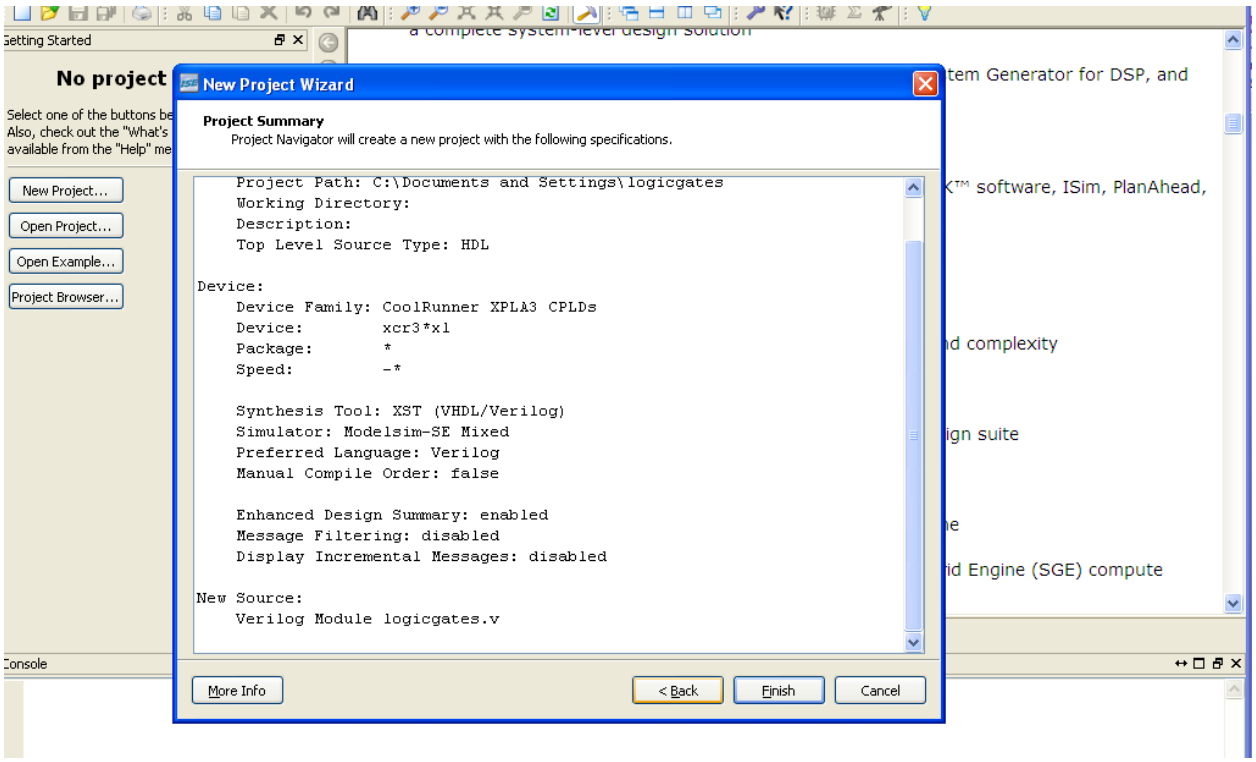
5. Finish with the New project setup with the ‘Summary’ window.



6. Once ‘Finish’ is selected a pop-up appears to create the directory. Select ‘yes’

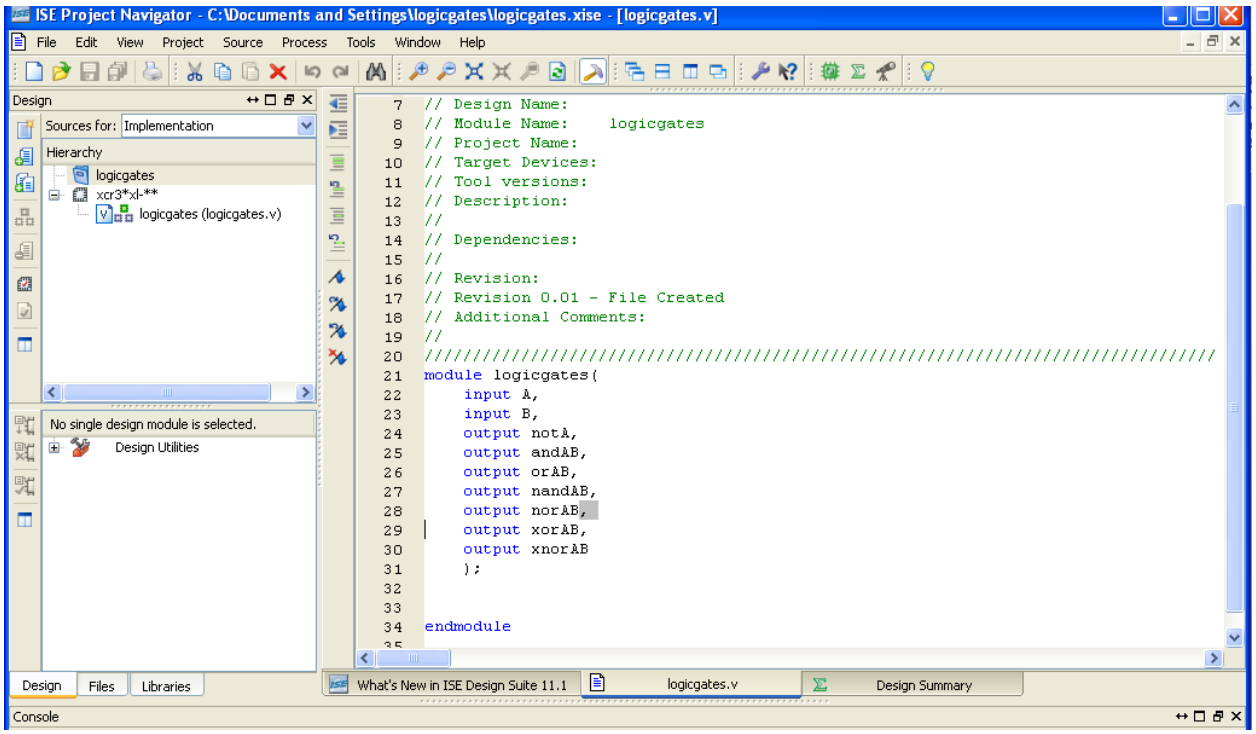


7. Then proceed to ‘Next’ in the “New Project Wizard’ to ‘Add Existing Sources’. ‘Add source’ if an existing source is available, If not proceed to ‘Next’ and finish with the ‘Project Summary’ window

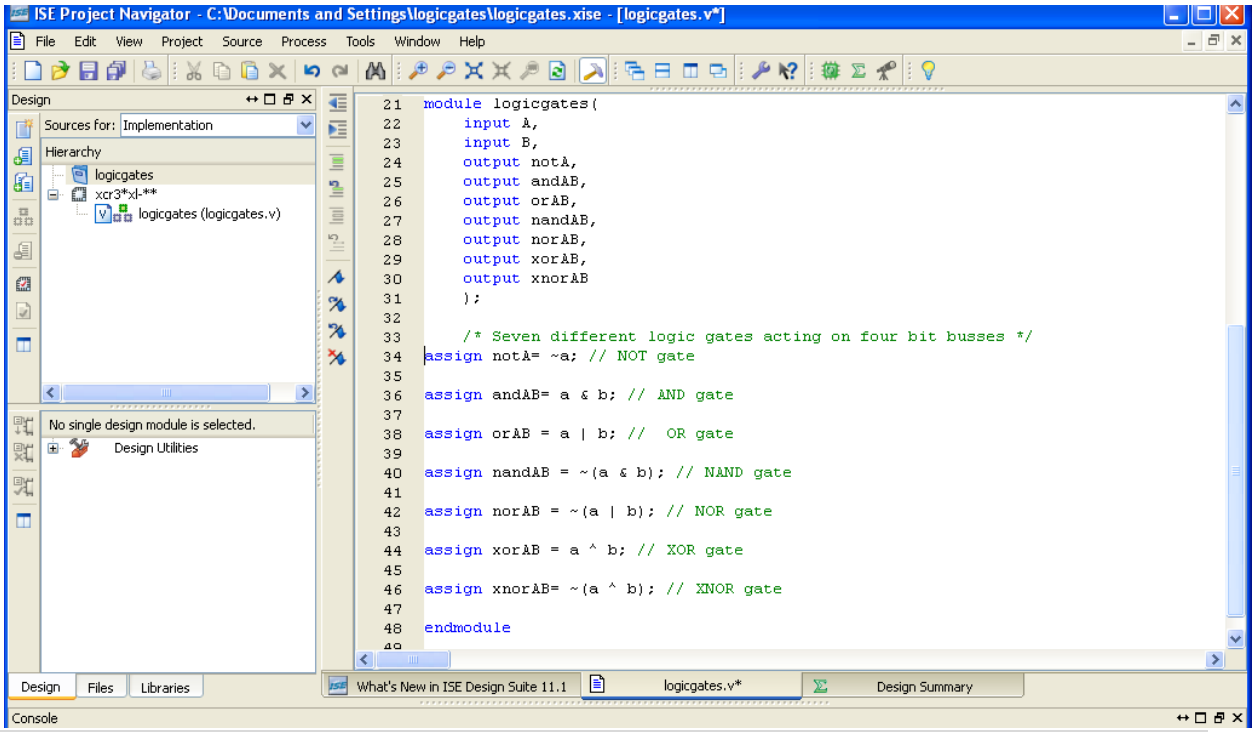


8. Design Entry and Syntax Check

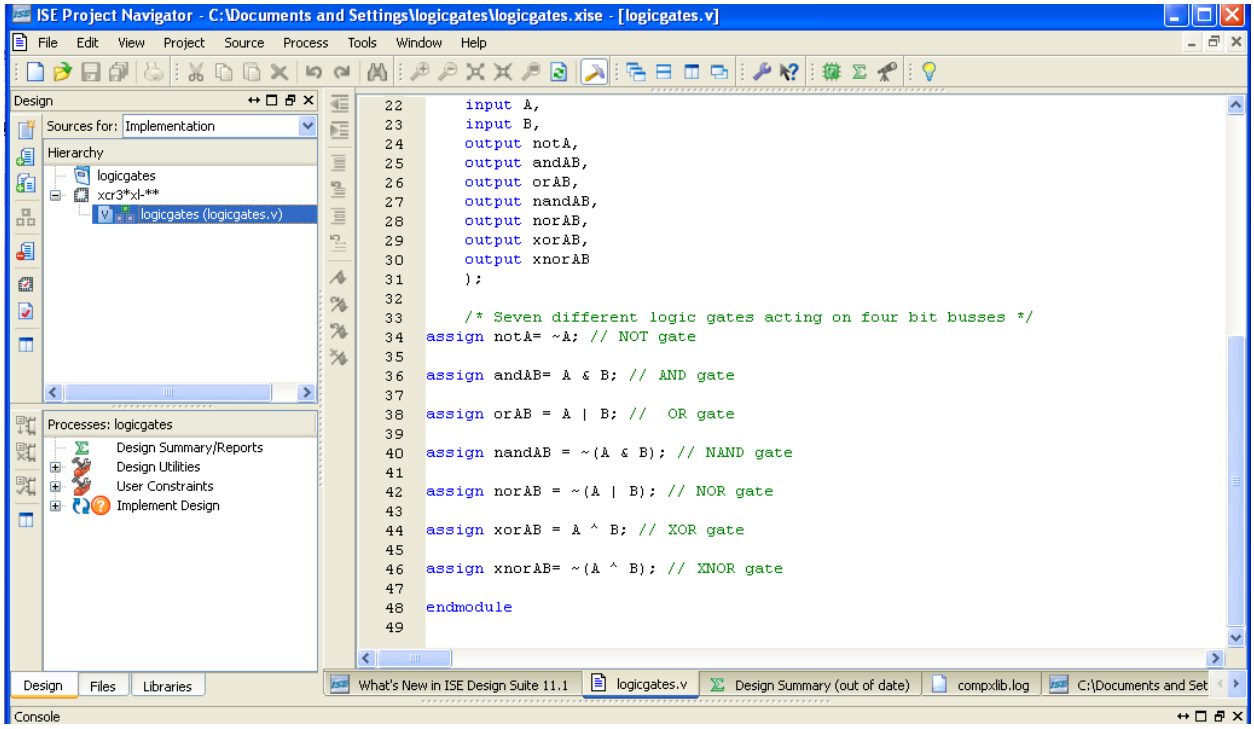
The ports defined during the 'Project Creation' are defined as a module in the 'filename.v' file



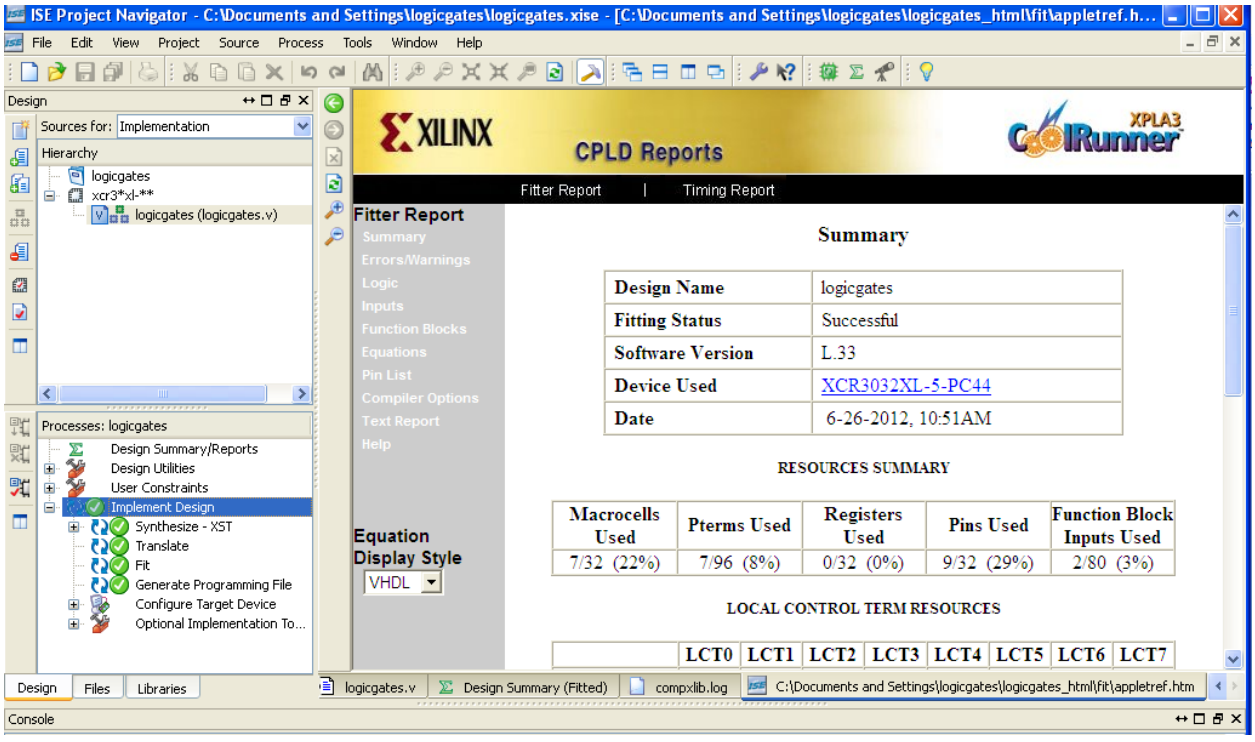
9. Input your design (verilog code) within the module definition



10. Select the design from the ‘Hierarchy‘window. In the below window of Processes ‘Implement Design ‘would be orange (in color) ready for implementation

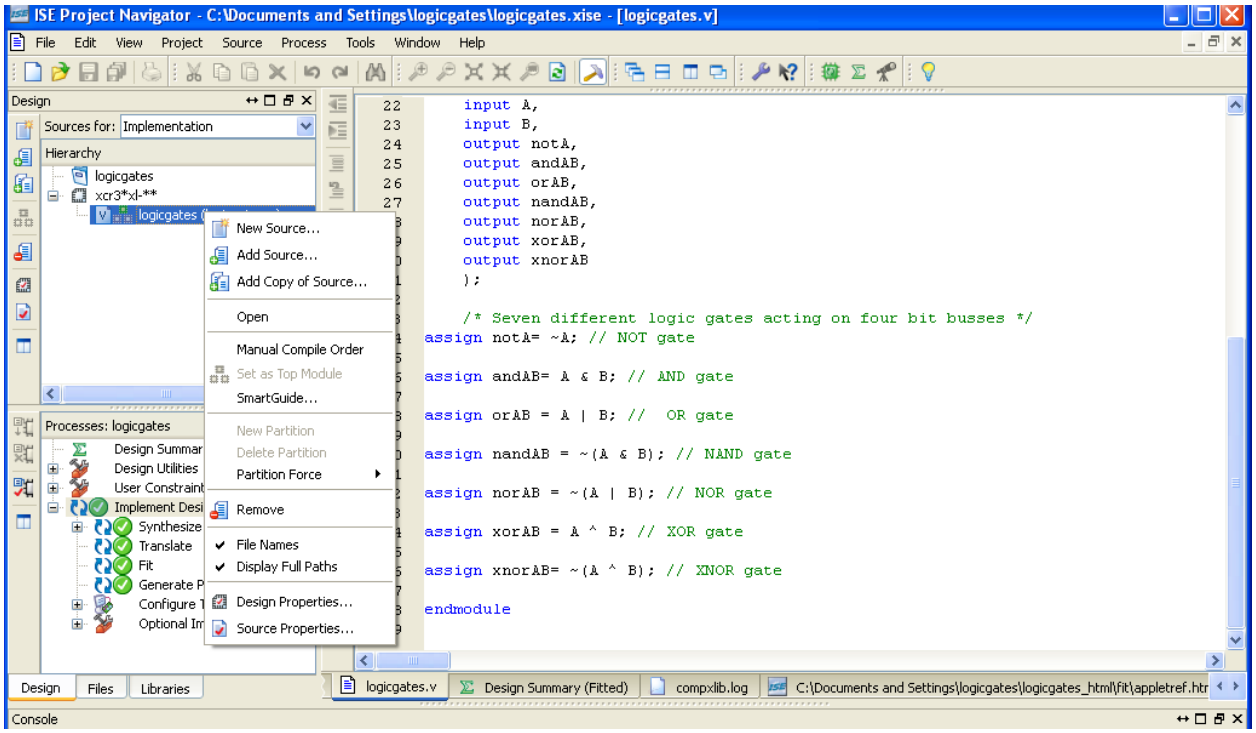


11. Double click on implement design, it turns green (in color) once the design is implemented successfully and the Summary report is displayed.

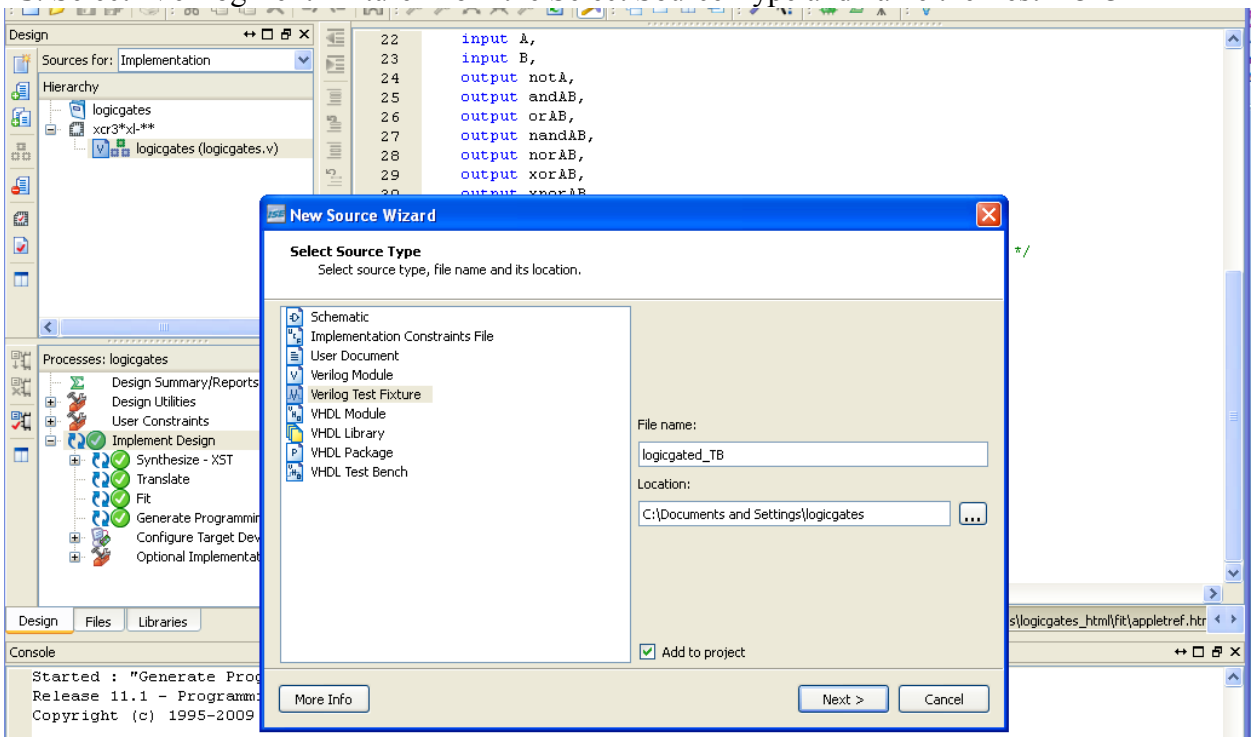


12. Test-Bench creation, Simulation & Verification

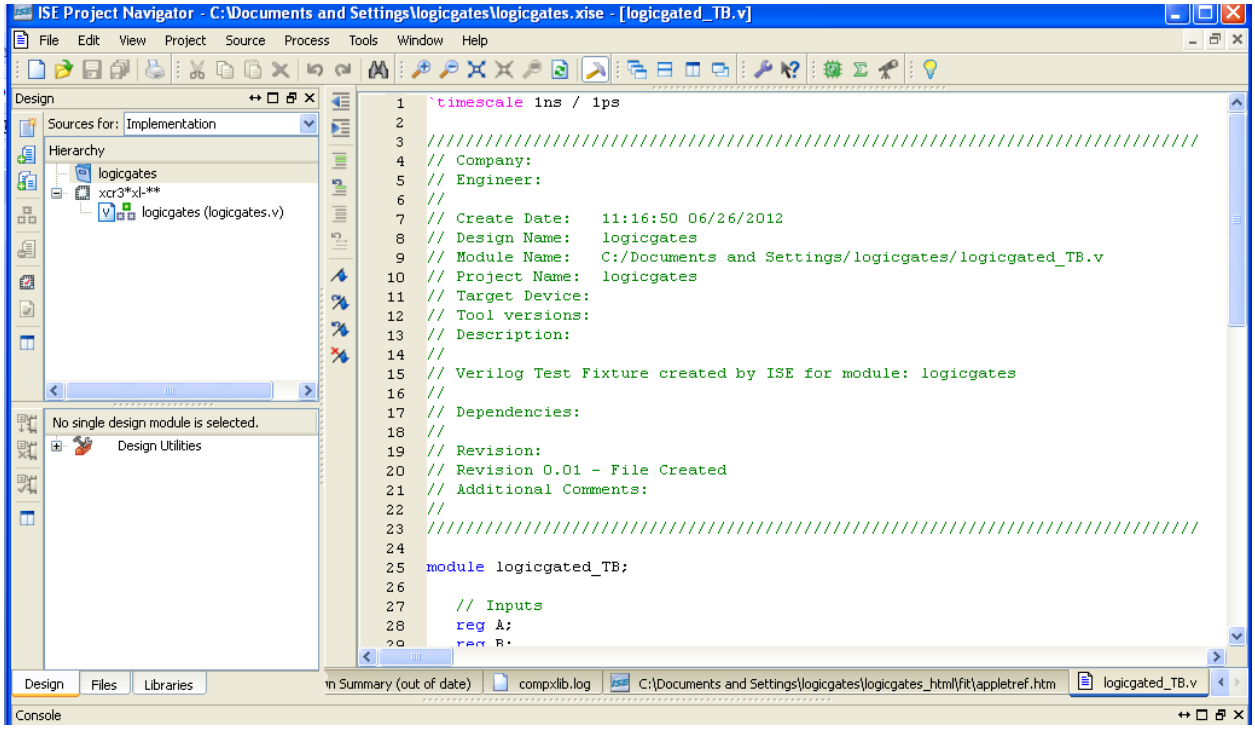
To add a test-bench to the existing design, right click on the '.v' file from the Hierarchy window and select 'New Source'



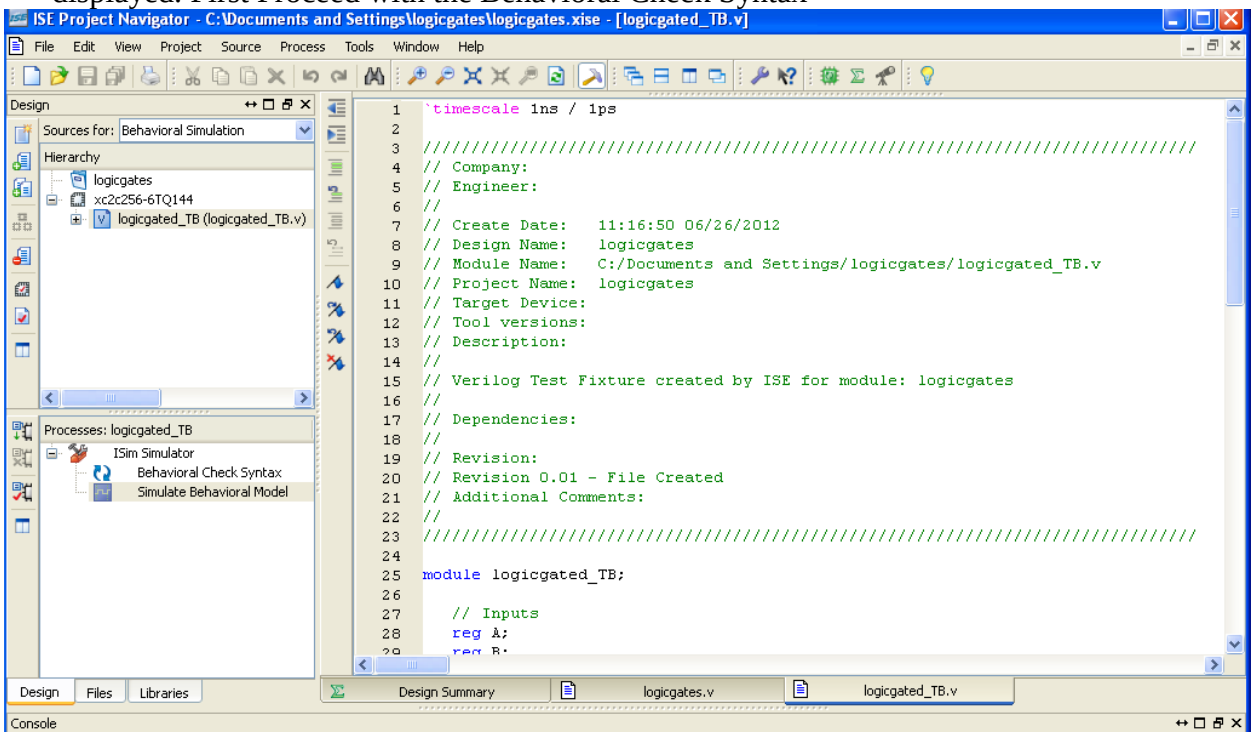
13. Select 'Verilog Text Fixture' from the Select Source Type and name the Test-Bench



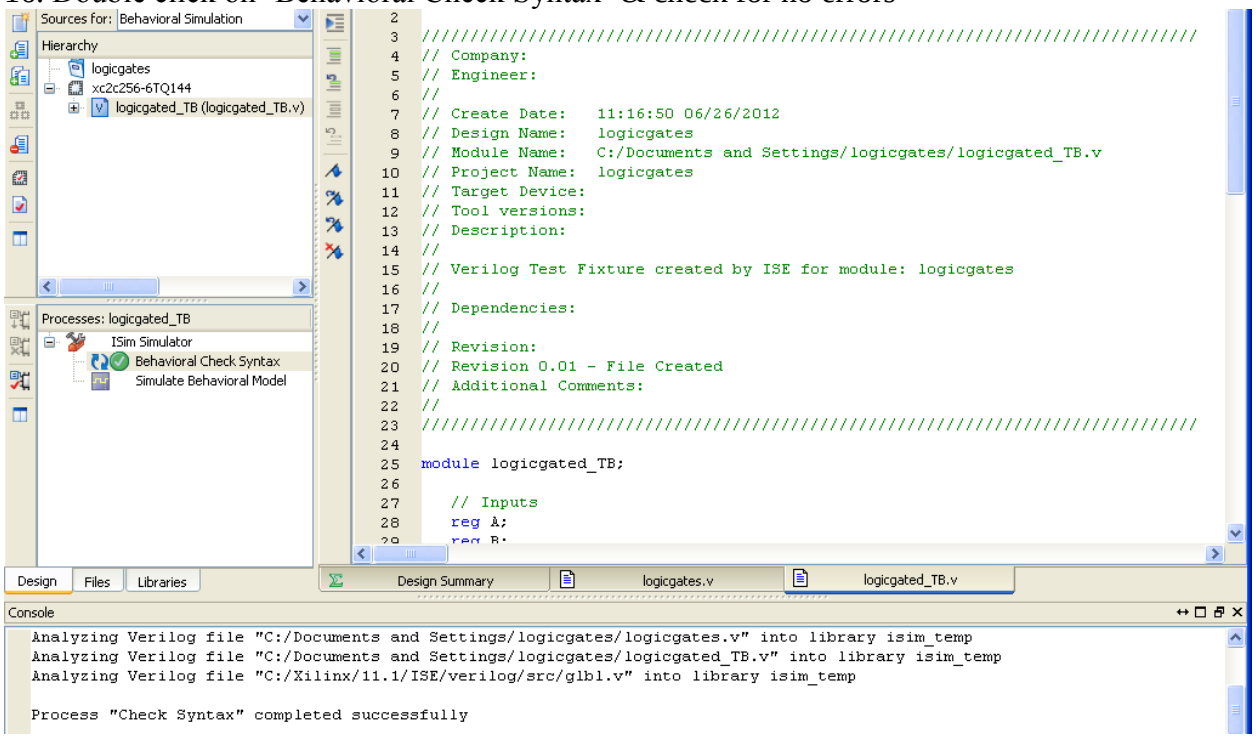
14. Continue to ‘Finish’ and a test bench is added in the project area



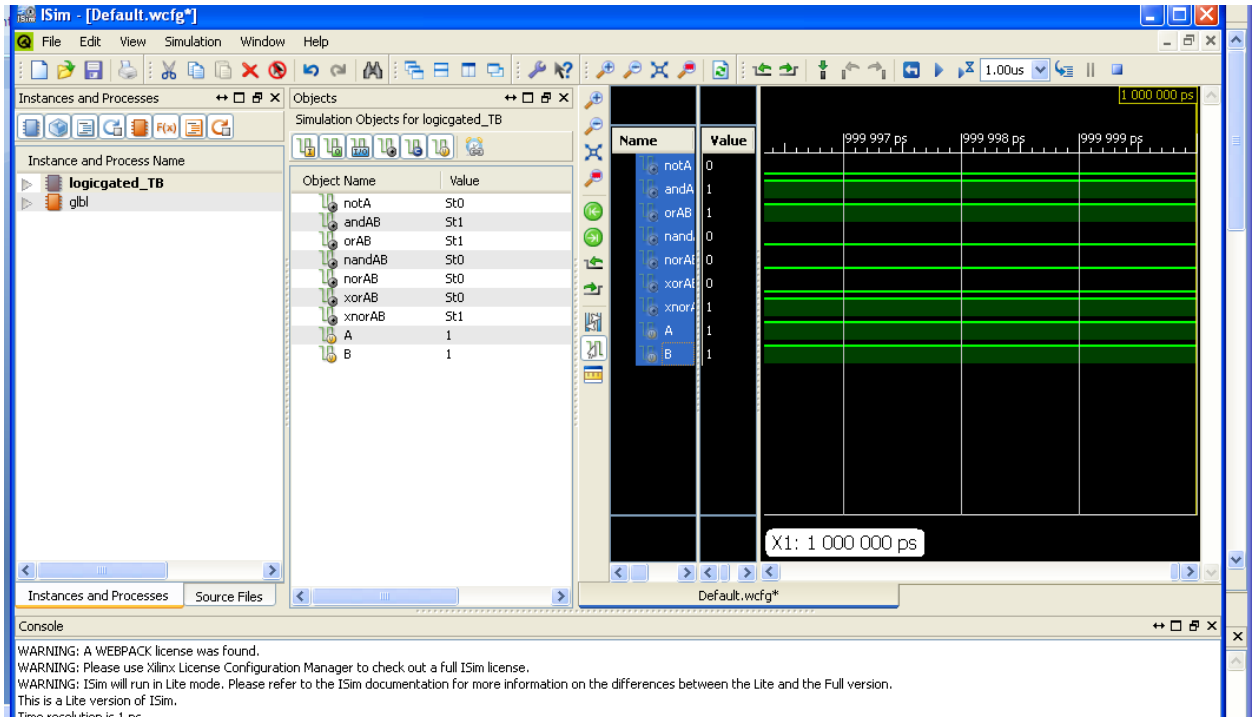
15. Edit the test bench as per your simulation requirements and select ‘Behavioral Simulation’ in the ‘Design Window’. In the Processes window Isim Simulator would be displayed. First Proceed with the Behavioral Check Syntax



16. Double click on ‘Behavioral Check Syntax’ & check for no errors



17. Then double click on ‘Simulate Behavioral Model’ and the ISIM simulator window would open. Check for the outputs



3 Altera Quartus II Programming Manual

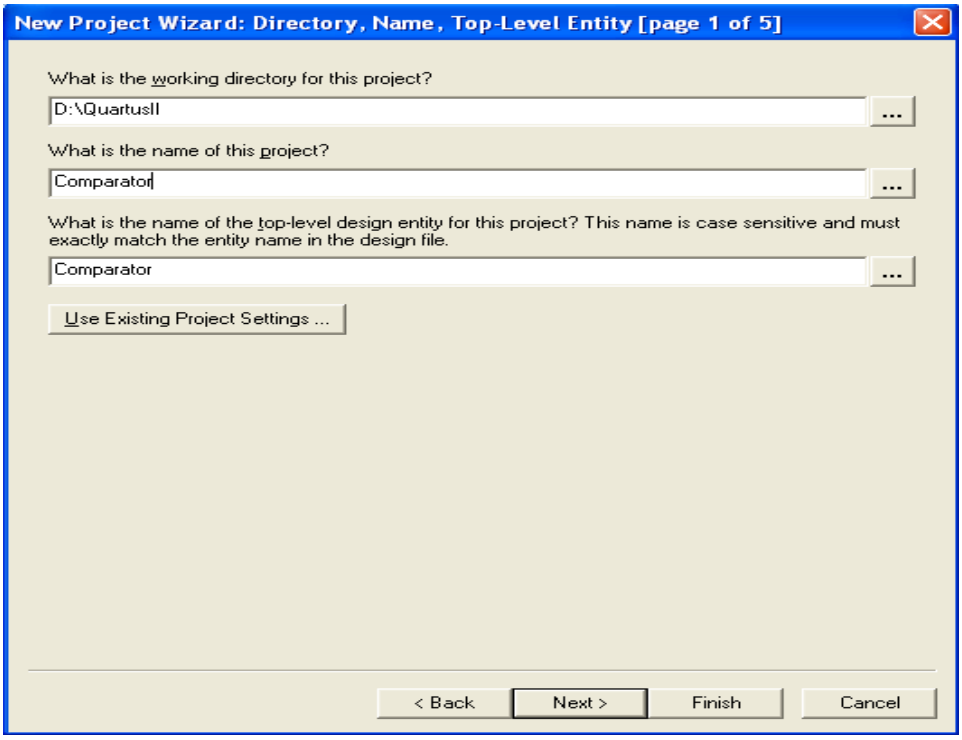
The Quartus II system includes full support for all of the popular methods of entering a description of the desired circuit into a CAD system. Typical FPGA CAD flow is shown below. The CAD flow involves the following steps:

- **Design Entry –** the desired circuit is specified either by means of a schematic diagram, or by using a hardware description language, such as Verilog or VHDL
- **Synthesis** – the entered design is synthesized into a circuit that consists of the logic elements (LEs) provided in the FPGA chip
- **Functional Simulation** – the synthesized circuit is tested to verify its functional correctness; this simulation does not take into account any timing issues
- **Fitting** – the CAD Fitter tool determines the placement of the LEs defined in the netlist into the LEs in an actual FPGA chip; it also chooses routing wires in the chip to make the required connections between specific LEs
- **Timing Analysis** – propagation delays along the various paths in the fitted circuit are analyzed to provide an indication of the expected performance of the circuit
- **Timing Simulation** – the fitted circuit is tested to verify both its functional correctness and timing
- **Programming and Configuration** – the designed circuit is implemented in a physical FPGA chip by programming the configuration switches that configure the LEs and establish the required wiring connections

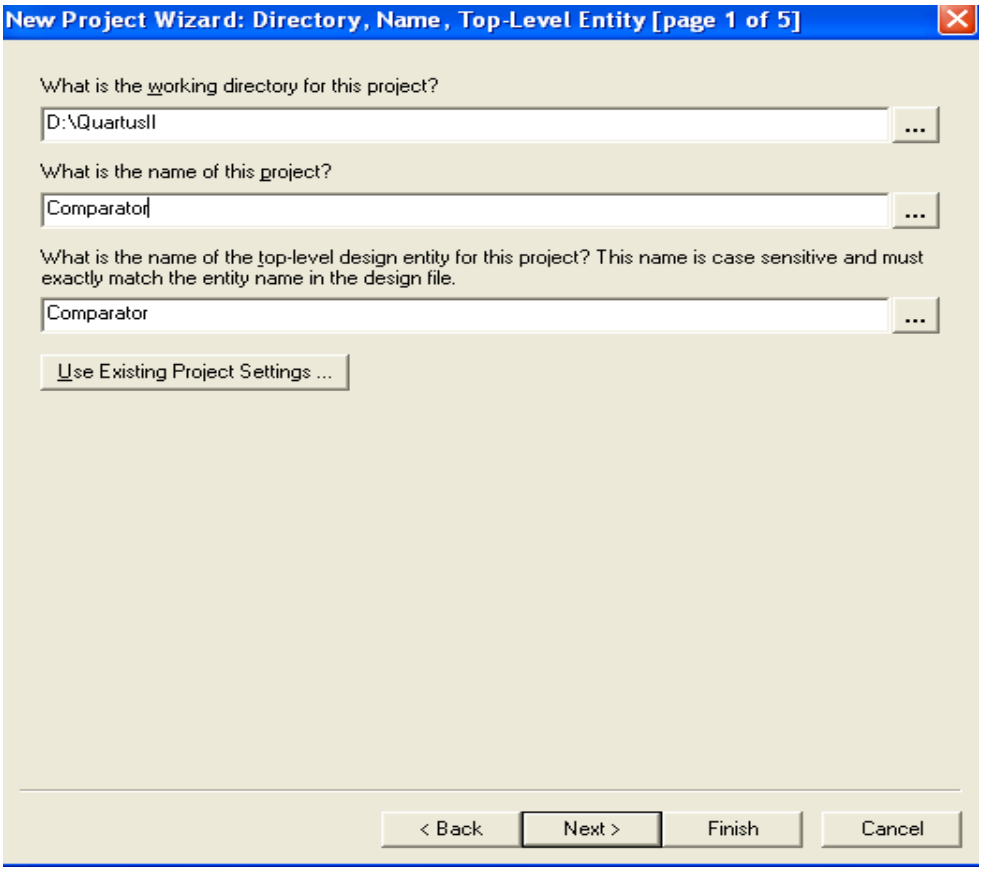
For the Lab, since the designs are already verified using Xilinx, we use Quartus II mainly for the Programming and the configuration of the FPGA
The Steps to from the start of the new project in Quartus II to programming the design in the DE2 Kit are illustrated below:

1. Invoke the Quartus II window and create a new project

2. Enter the working directory, project name and the top-level-design



3. Add your verilog file to the project



4. The “Add Files” window will appear. Enter the design file to be programmed. If there is no design file created, leave this section blank. Click ‘Next’

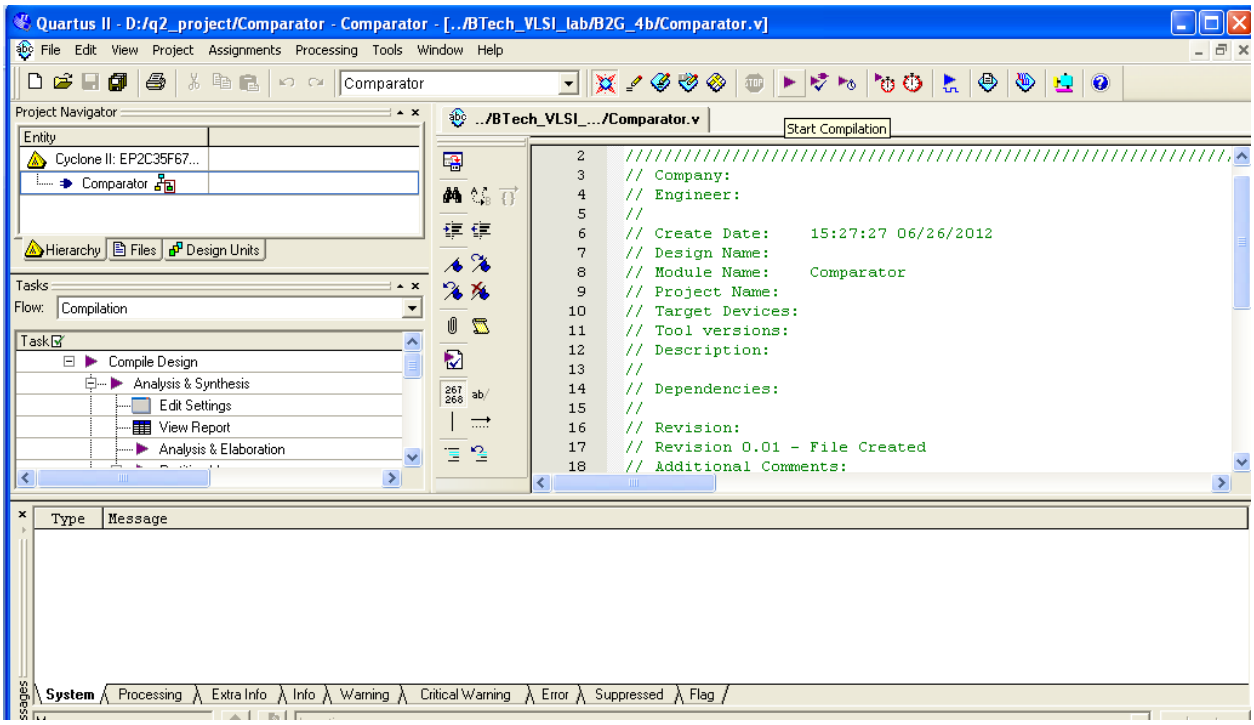
5. The “Family and Device Settings” will appear. Enter the settings for the Family, Package, Pin Count and Speed Grade

   The settings would vary depending on the board

   For CYCLONE II,
   a. Family > Cyclone II
   b. Name > EP2C35F672C6 FPGA

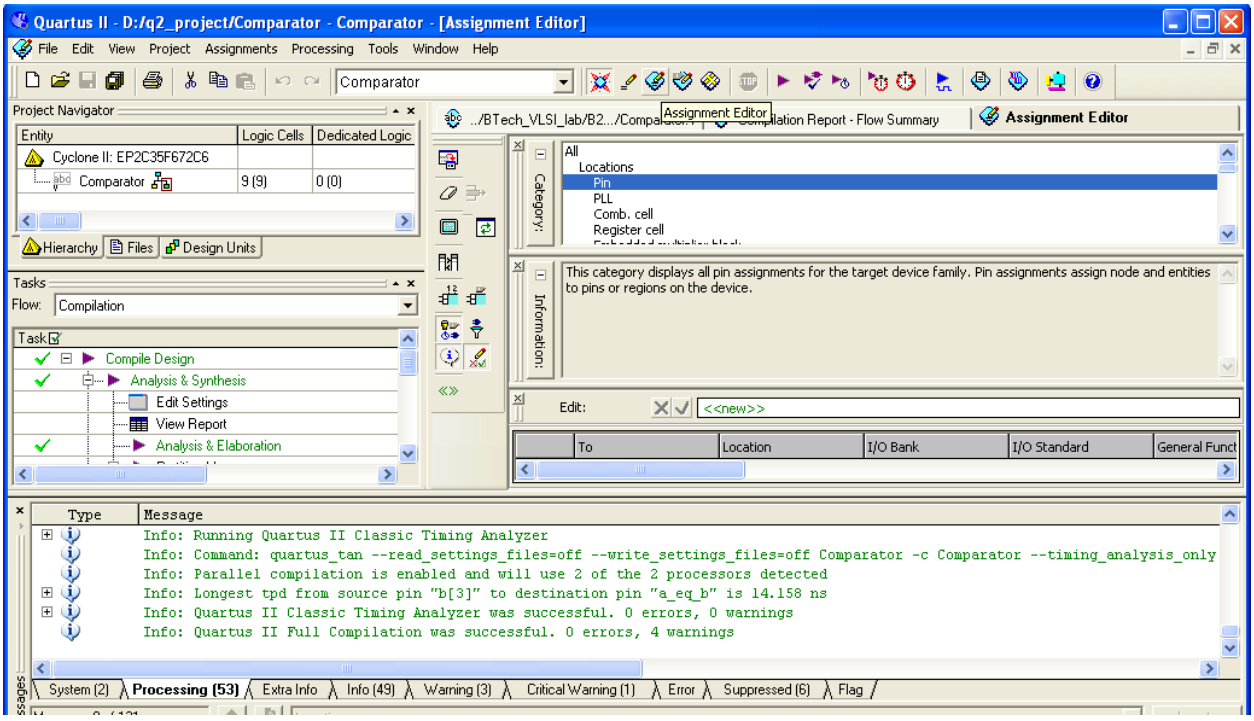
   For CYCLONE III,
   a. Family > Cyclone III
   b. Package > FBGA
   c. Pin Count > 324
   d. Speed Grade > Fastest
   e. Name > EP3C25F324C6

6. The “EDA Tool Settings” window will appear. Select the desired tools and proceed.
7. The “Summary” window will appear. Verify that the settings are correct. Click ‘Finish’.
8. The project window with the added verilog file is opened and then continue to compile.

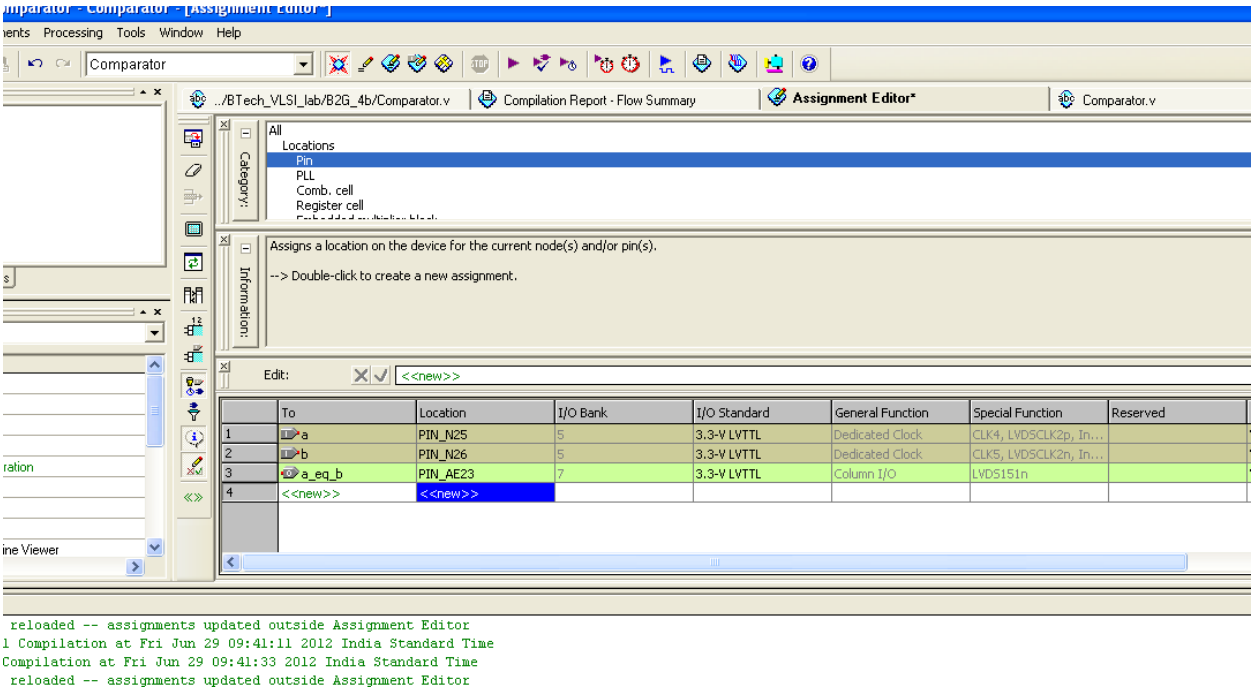


9. Once the compilation is done assign the pins using the Assignment editor

- For the CYCLONE II KIT

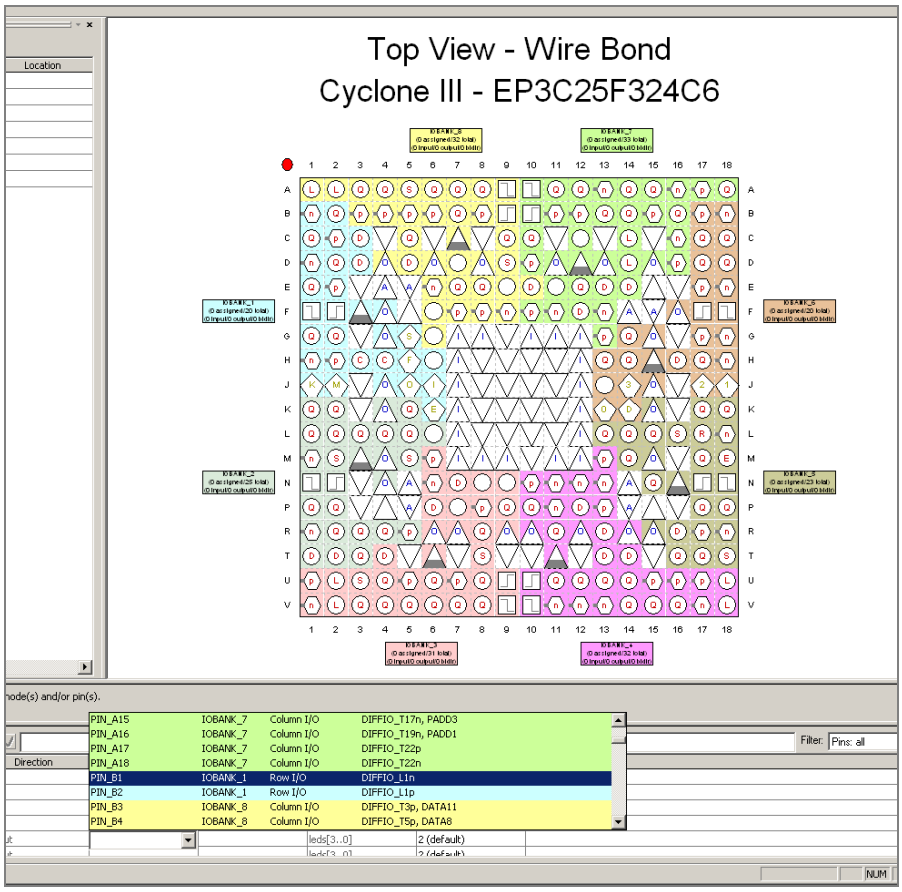


Assign the pins as per the CYCLONE II pin assignment (Section5)

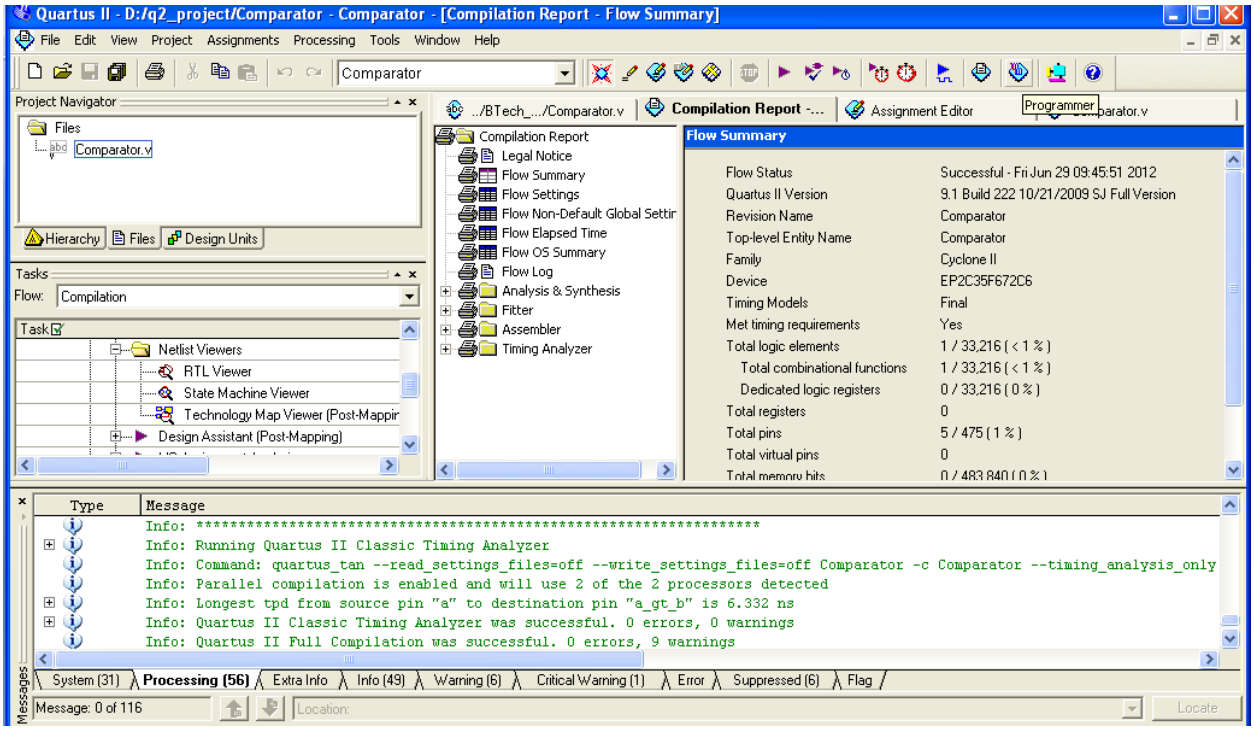


- For the CYCLONE III KIT

Select the PIN planner from the menu and assign the pins as per the CYCLONE III pin assignments (Section6)

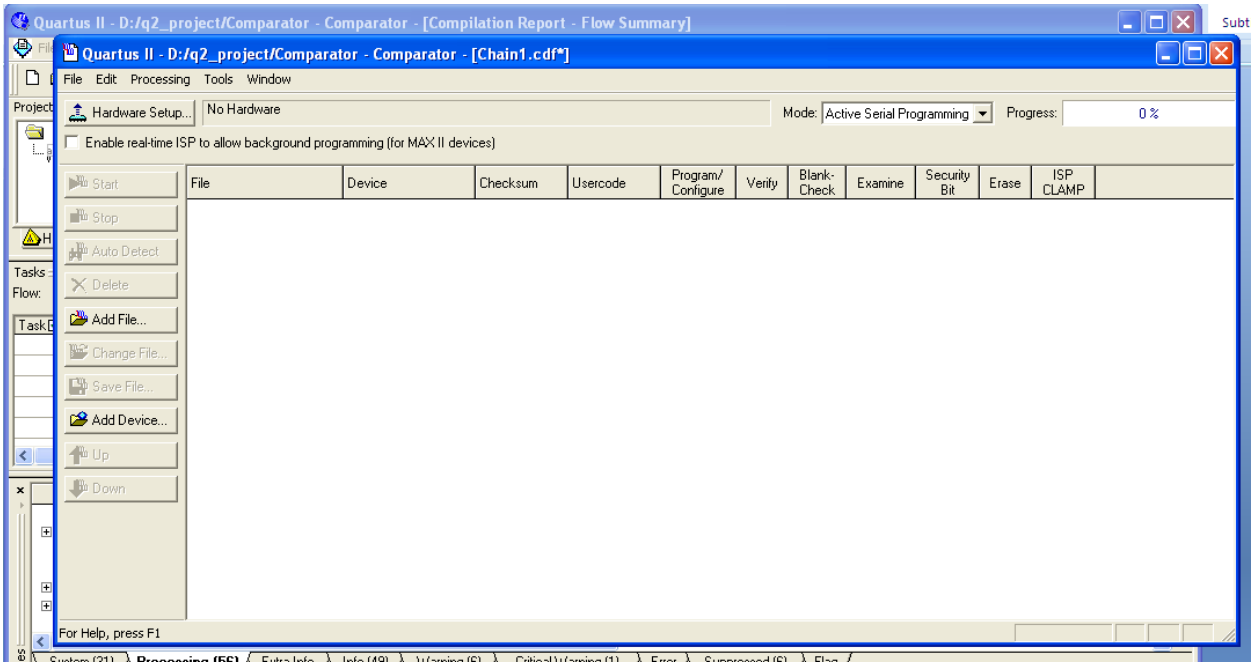


10. Recompile after the pin assignment. Once the compilation is successful, open the programmer window from the menu



- In the programmer window, select the mode as 'Active Serial programming' and choose the 'USB blaster in the Hardware setup. (USB blaster cable to be connected from the DE2/DE3 kit.

Select the .pof file and start the programming if the KIT is for CYCLONE II FPGA
Select the .sof file and start the programming if the KIT is for CYCLONE II FPGA


Quartus II - D:/q2_project/Comparator - Comparator - [Compilation Report - Flow Summary]
Quartus II - D:/q2_project/Comparator - Comparator - [Chain1.cdf*]
File Edit Processing Tools Window
Hardware Setup...
No Hardware
Mode: Active Serial Programming
Progress:
0 %
Enable real-time ISP to allow background programming (for MAX II devices)
Start
Stop
Auto Detect
Delete
Add File...
Change File...
Save File...
Add Device...
Up
Down
File
Device
Checksum
Usercode
Program/ Configure
Verify
Blank-Check
Examine
Security Bit
Erase
ISP CLAMP
For Help, press F1

4 Cadence User Manual (IC613)

This User Manual is provided by the Cadence Design Systems Support Team and has been integrated into our lab manual.
Objective of this lab is to learn the Virtuoso tool as well learn the flow of the Full Custom IC design cycle. You will finish the lab by running DRC, LVS and Parasitic Extraction on the various designs. In the process you will create various components like inverter, differential amplifier, operational amplifier etc.
Start the lab by creating a library called "**myDesignLib**" and attach the library to a technology library called "**gpdk180**". Attaching a technology library will ensure that you can do front to back design.
Create a new cell called "**Inverter**" with schematic view and hence build the inverter schematic by instantiating various components. Once inverter schematic is done, symbol for "**Inverter**" is generated. Now you will create a new cell view called "**Inverter_Test**", where you will instantiate "**Inverter**" symbol. This circuit is verified by doing various simulations using spectre. In the process, you will learn to use spectre, waveform window options, waveform calculator, etc...
You will learn the Layout Editor basics by concentrating on designing an "**Inverter**" through automatic layout generation. Then you will go ahead with completing the other layouts. After that, you will run DRC, LVS checks on the layout, Extract parasitics and back-annotate them to the simulation environment.
After completing the parasitic back- annotation flow, design is ready for generating GDSII.

4.1 General Notes

There are a number of things to consider before beginning these lab exercises. Please read through this section completely, and perform any needed steps in order to ensure a successful workshop. These labs were designed for use with Incisive Unified Simulator82, IC613 and Assura32.
Before running any of these labs, ensure that you've set up IUS82, IC613, MMSIM71 and Assura32 correctly:
%> setenv CDSHOME <*IC613-installation-home*>
%> setenv MMSIMHOME <MMSIM71-installation-home>
%> setenv PVHOME <Assura32-*installation-home*>
%> setenv AMSHOME <*IUS82-installation-home*>

You will also need to ensure that the IUS82 is setup correctly for lab 5.
To setup the lab environment, please perform the following steps:

1. Ensure the software mentioned above is correctly setup.
2. Source the C-Shell related commands file i.e. (cshrc file).

These labs were designed to be run using Cadence Virtuoso tool and Assura tool.

Lab Getting Started

1. Log in to your workstation using the username and password. The home directory has a **cshrc** file with paths to the Cadence installation.
2. In a terminal window, type **csh** at the command prompt to invoke the C shell.
>csh
>source cshrc
3. To verify that the path to the software is properly set in the **cshrc** file, type the below command in the terminal window and enter:
>which virtuoso
It gives the complete path of IC613 tool Installation.
>**which spectre**
It gives the complete path of MMSIM71 tool Installation.
>**which assura**
It gives the complete path of Assura32 tool Installation.

>**which ncsim**
It gives the complete path of IUS82 tool Installation.

Starting the Cadence Software

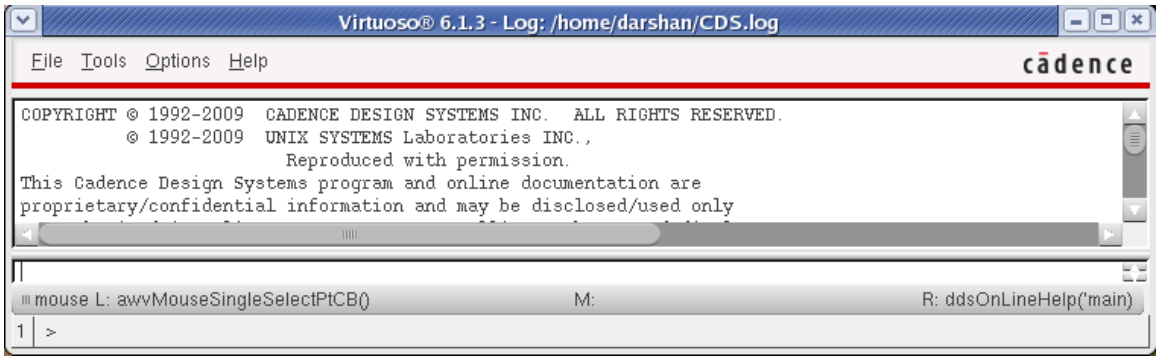
Use the installed database to do your work and the steps are as follows:

1. Change to the course directory by entering this command:
> **cd ~/Database/cadence_analog_labs_613**
You will start the Cadence Design Framework II environment from this directory because it contains cds.lib, which is the local initialization file. The library search paths are defined in this file.
The **Cadence_Analog_labs_613** directory contains Solutions folder and also Work folder. Inside Work folder you can create new cell / modifications of the cell locally without affecting your Source cell present inside Solutions directory.

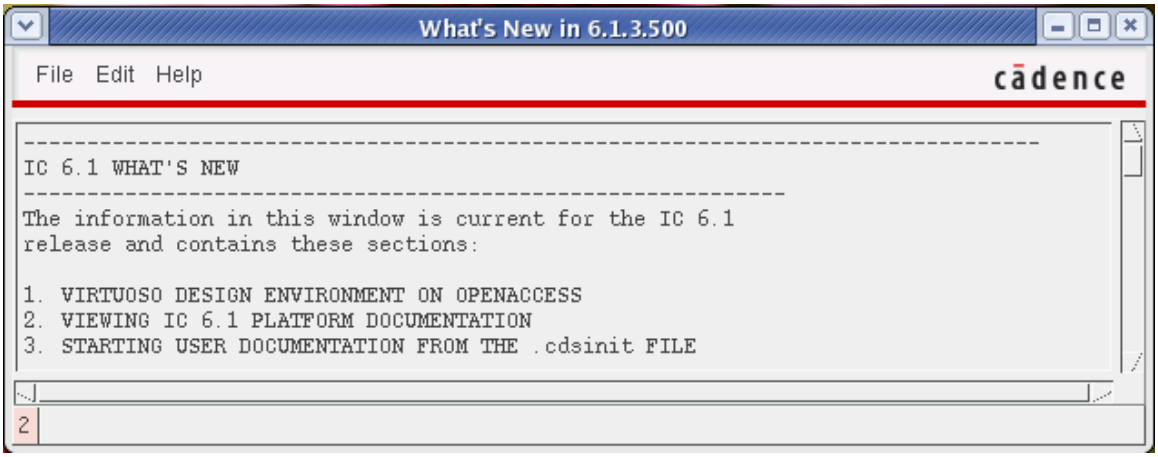
Lab directory details:

. /Solution	Contains a local copy of all the lab experiments including test circuit for simulation.
. /libs.cdb	Contains a technology library for the design (gpdk180nm).
. /models	Contains spectre models of components for simulation in gpdk180nm technology.
. /stream	Contains layer map file for GDSII format
. /pv	Containing the Assura and Diva verification files
. /techfiles	Contains ASCII versions of the oa22 techfiles
. /dig_source	Contains verilog codes for SAR register and clock
. /cds.lib	File containing pointer to the Cadence OA22 initialization file**.**
. /hdl.var	File defines the work library for AMS simulation
. /docs	Reference manual and user manual for gpdk180nm technology

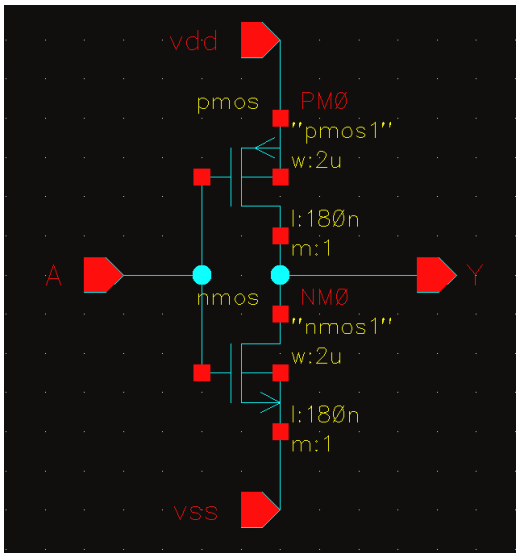
2. In the same terminal window, enter:
> **virtuoso &**
The virtuoso or Command Interpreter Window (CIW) appears at the bottom of the screen.



3. If the "What's New ..." window appears, close it with the **File— Close** command.
4. 4. Keep opened CIW window for the labs.



4.2 Tutorial1: Inverter



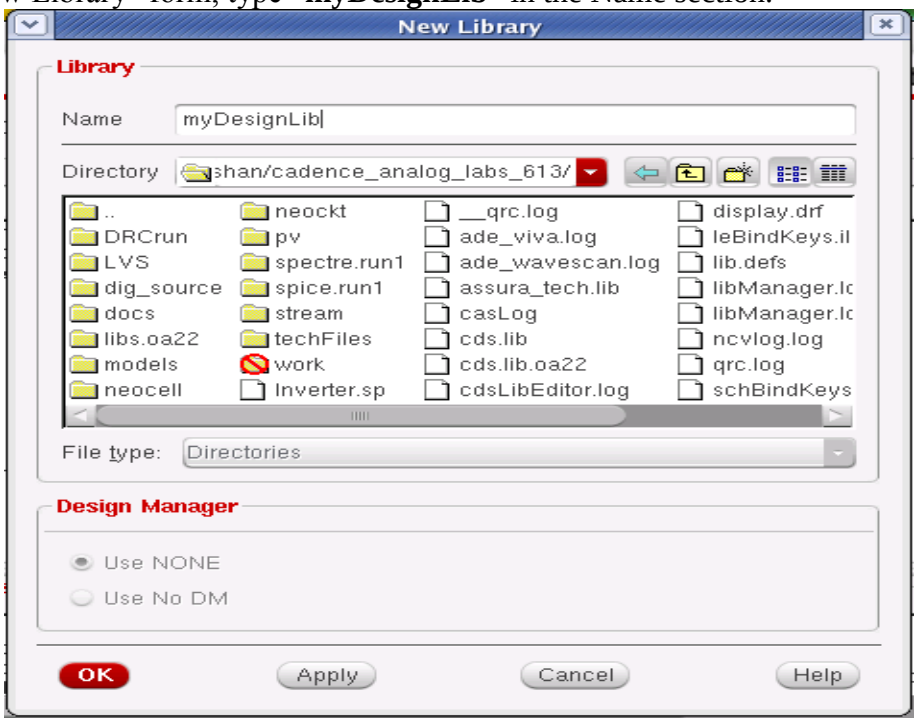
Schematic Capture

4.2.1 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.

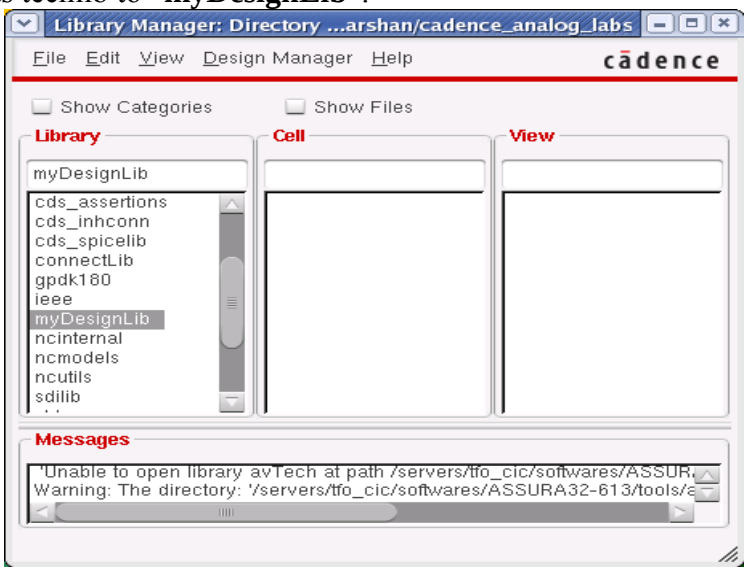


3. In the field of Directory section, verify that the path to the library is set to **~/Database/cadence_analog_labs_613** and click **OK**.
Note: A technology file is not required if you are not interested to do the layouts for the design

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

5. In the "**Attach Design Library to Technology File**" form, select **gpdk180** from the cyclic field and click **OK**.

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**".



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**
gpdk180	pmos	For M0: Model name = pmos1, **W= wp**, L=180n
gpdk180	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.

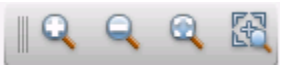
Adding pins to Schematic

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.

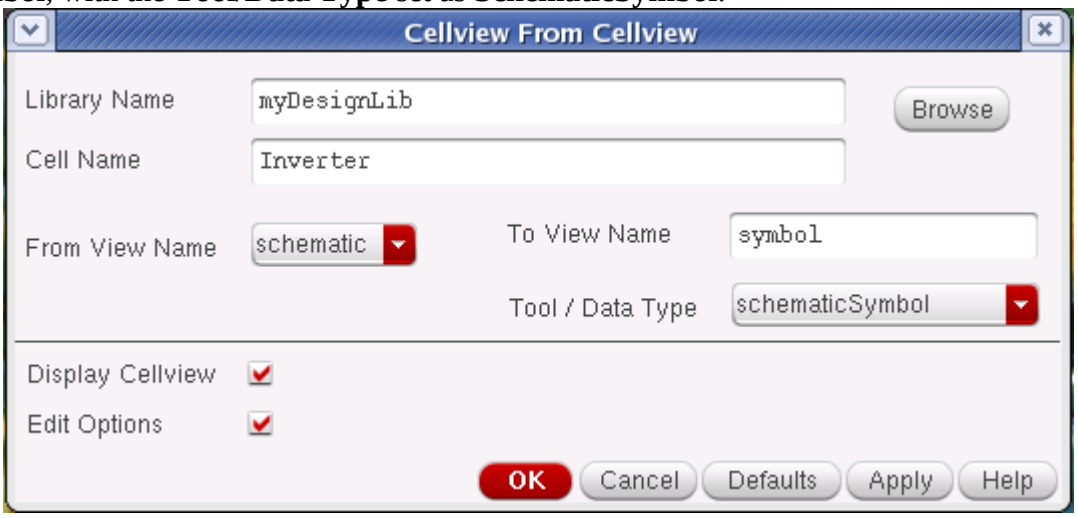
4.2.2 Symbol Creation

Objective: To create a symbol for the Inverter

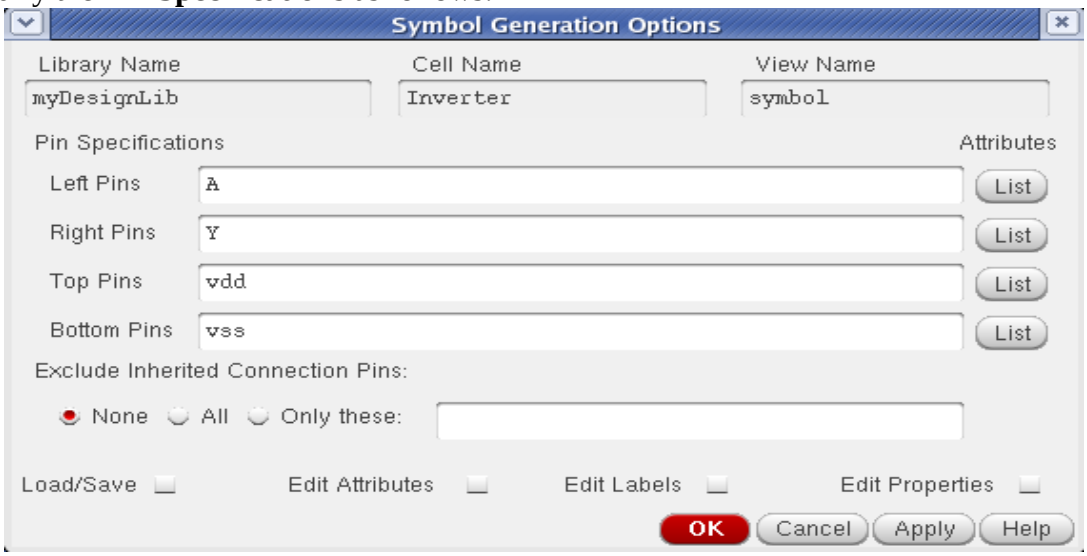
In this section, you will create a symbol for your inverter design so can place it in a test circuit for simulation. A symbol view is extremely important step in the design process. The symbol view must exist for the schematic to be used in a hierarchy. In addition, the symbol has attached properties (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.

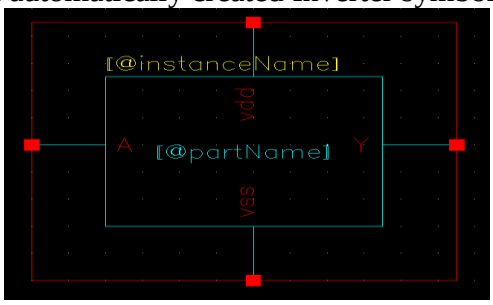
2. Verify that the **From View Name** field is set to **schematic**, and the **To View Name** field is set to **symbol**, with the **Tool/Data Type** set as **SchematicSymbol**.



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



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



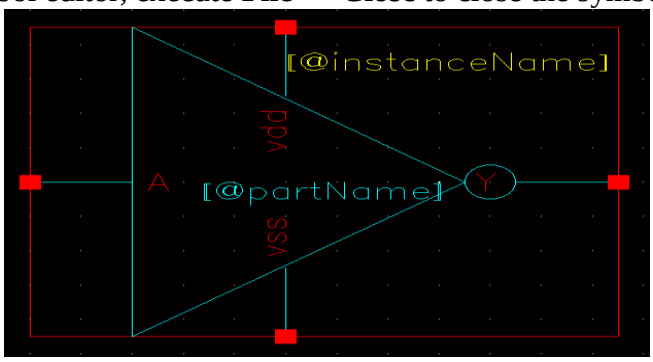
Editing a Symbol

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.



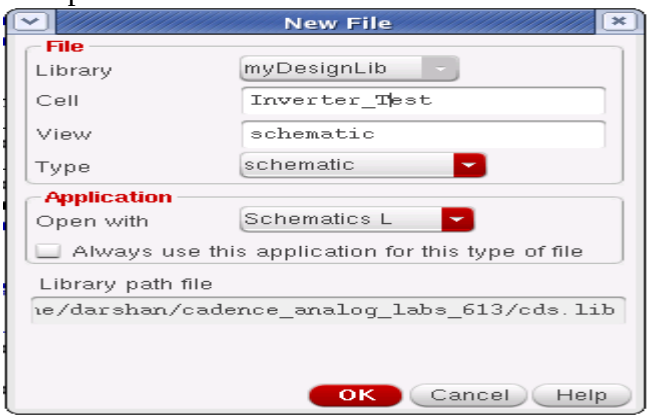
4.2.3 Building the Inverter Design

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.

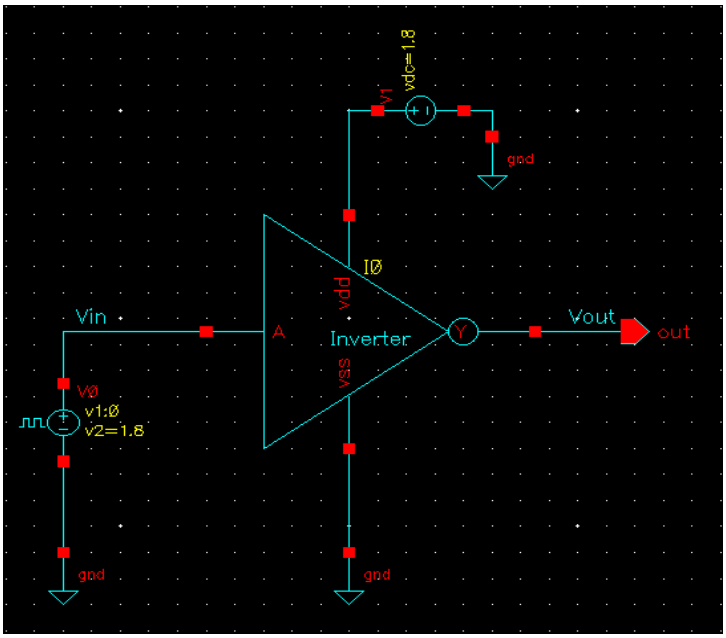
Library name	Cellview name	Properties/Comments
myDesignLib	Inverter	Symbol
analogLib	vpulse	v1=0, v2=1.8,td=0 tr=tf=1ns, ton=10n, T=20n
analogLib	vdc, gnd	vdc=1.8

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.

5. Click on the ***Check and Save*** icon to save the design.
6. The schematic should look like this.



4.2.4 Analog Simulation using 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.

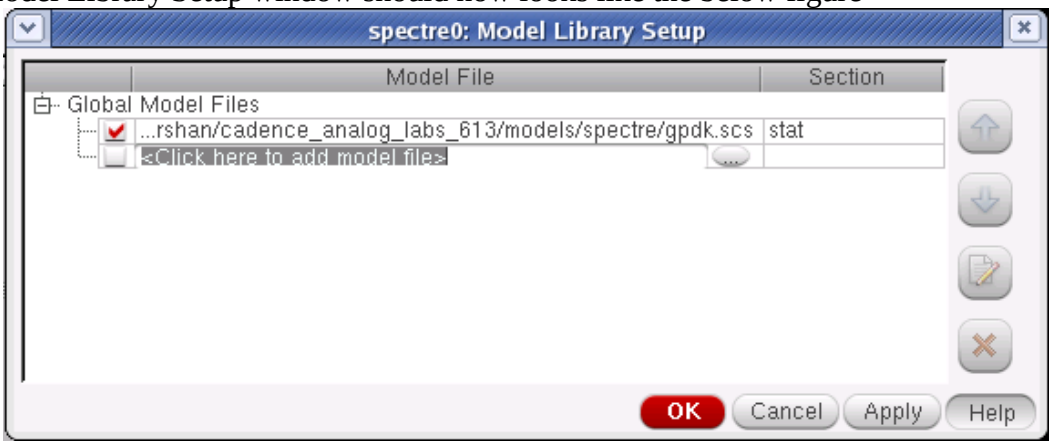
Choosing a Simulator

Set the environment to use the **Spectre® tool**, a high speed, highly accurate analog simulator. Use this simulator with the **Inverte*r_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 looks like the below figure



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

Edit File. 

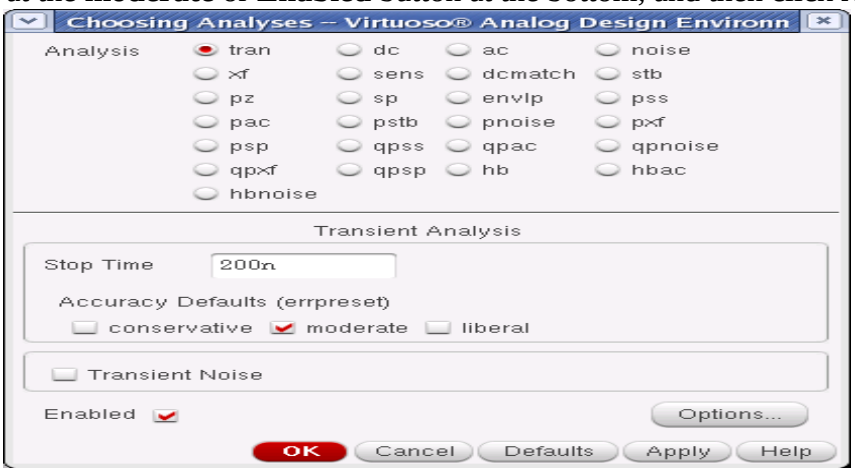
2. To complete the Model Library Setup, move the cursor and click **OK**.
The Model Library Setup allows you to include multiple model files.
It also allows you to use the Edit button to view the model file.

Choosing Analyses

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

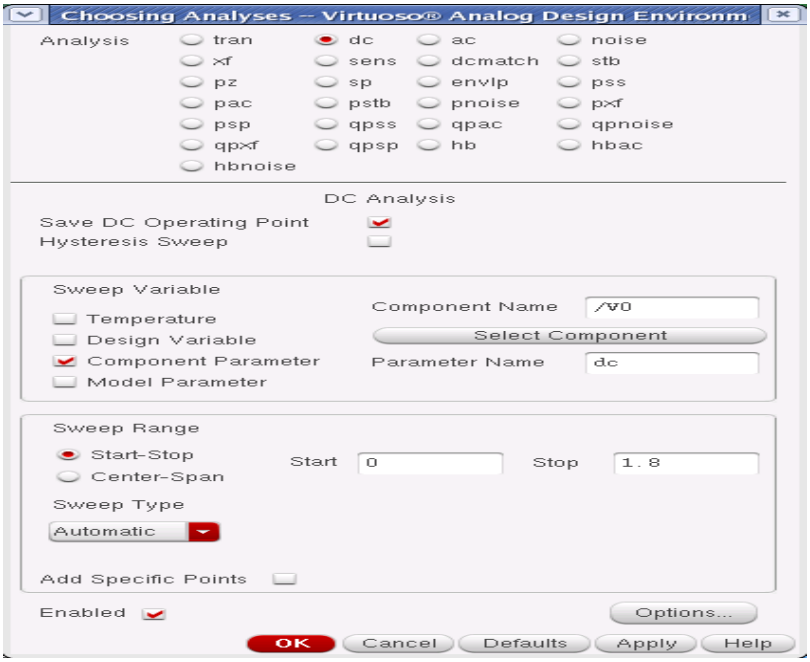
1. In the Simulation window (ADE), click the **Choose - Analyses** icon.
You can also execute **Analyses - Choose.**
The Choosing Analysis form appears. This is a dynamic form, the bottom of the form changes based on the selection above.

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:

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** form and click OK.
g. In the analysis form type **start** and **stop** voltages as **0** to **1.8** respectively.
h. 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

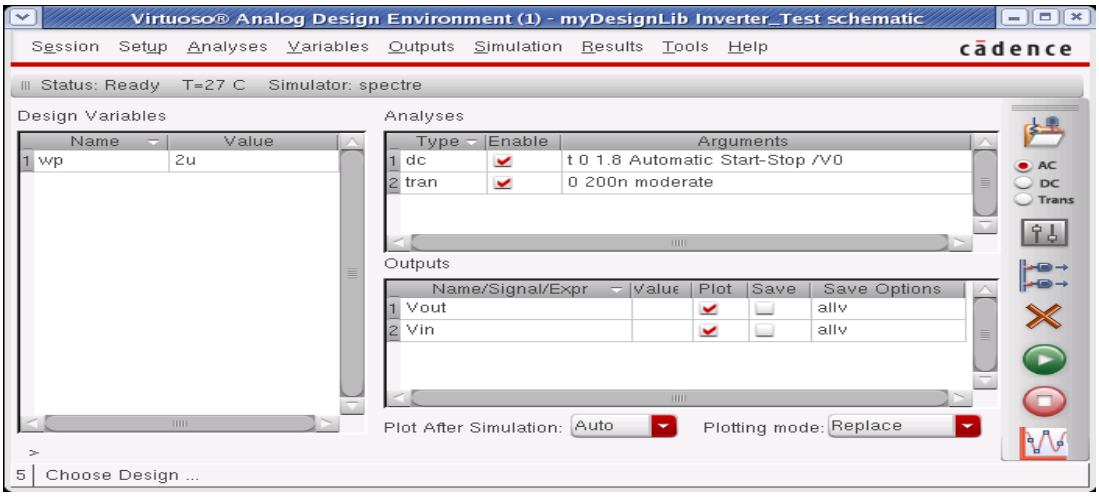
Click **Change** and notice the update in the Table of Design Variables.

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

Selecting Outputs for Plotting

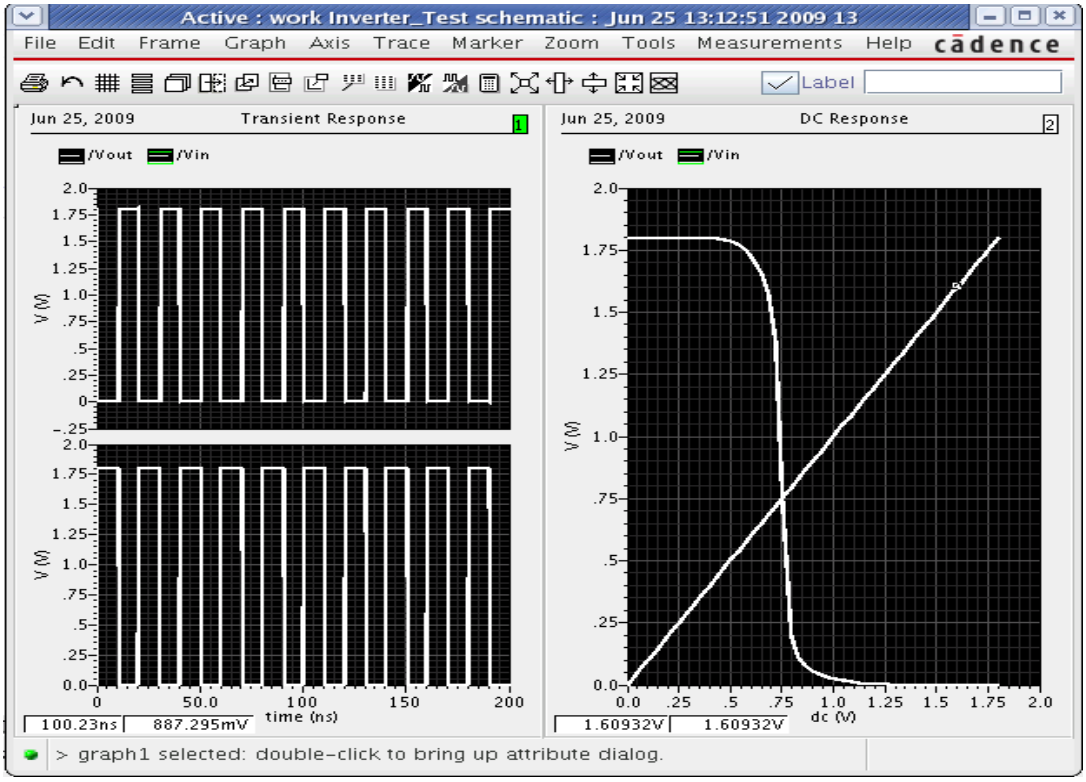
1. Execute **Outputs – To be plotted – Select on Schematic** in the simulation window.
2. Follow the prompt at the bottom of the schematic window, Click on output net **Vout**, input net **Vin** of the Inverter. Press **ESC** with the cursor in the schematic after selecting it.

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.



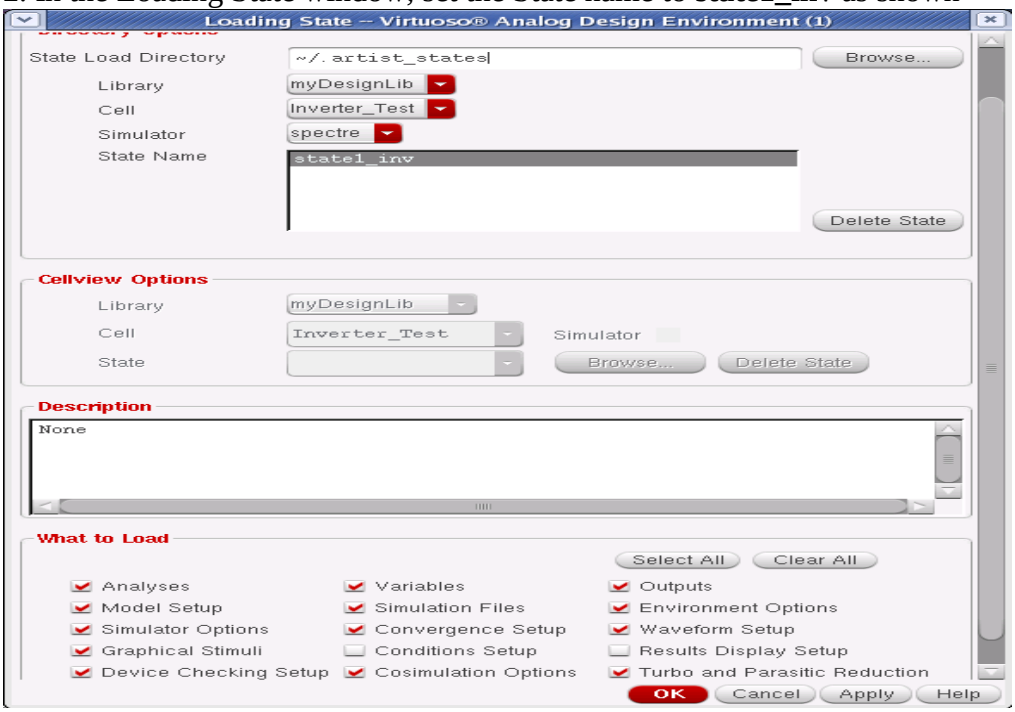
Saving the Simulator State

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.

4.2.5 Parametric analysis

Parametric Analysis yields information similar to that provided by the Spectre® sweep feature, except the data is for a full range of sweeps for eachparametric step. The Spectre sweep feature provides sweep data at only one specified condition.

You will run a parametric DC analysis on the **wp** variable, of the PMOS device of the Inverter design by sweeping the value of **wp**.

Run a simulation before starting the parametric tool. You will start by loading the state from the previous simulation run.

Run the simulation and check for errors. When the simulation ends, a single waveform in the waveform window displays the DC Response at the **Vout** node.

Starting the Parametric Analysis Tool

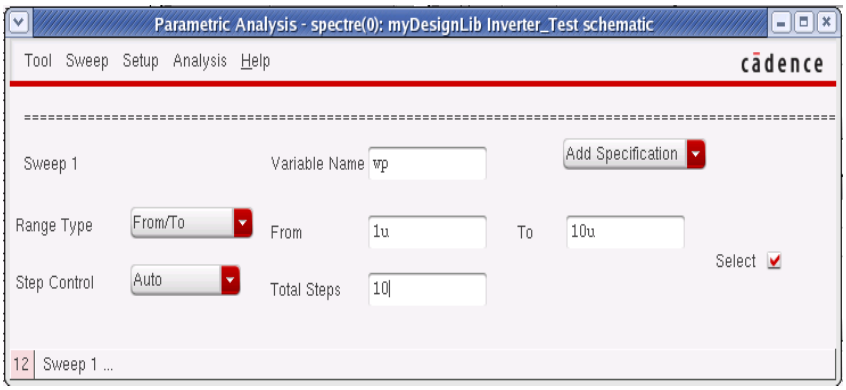
1. In the Simulation window, execute **Tools—Parametric Analysis**. The Parametric Analysis form appears.
2. In the Parametric Analysis form, execute
Setup—Pick Name For Variable—Sweep 1.

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:

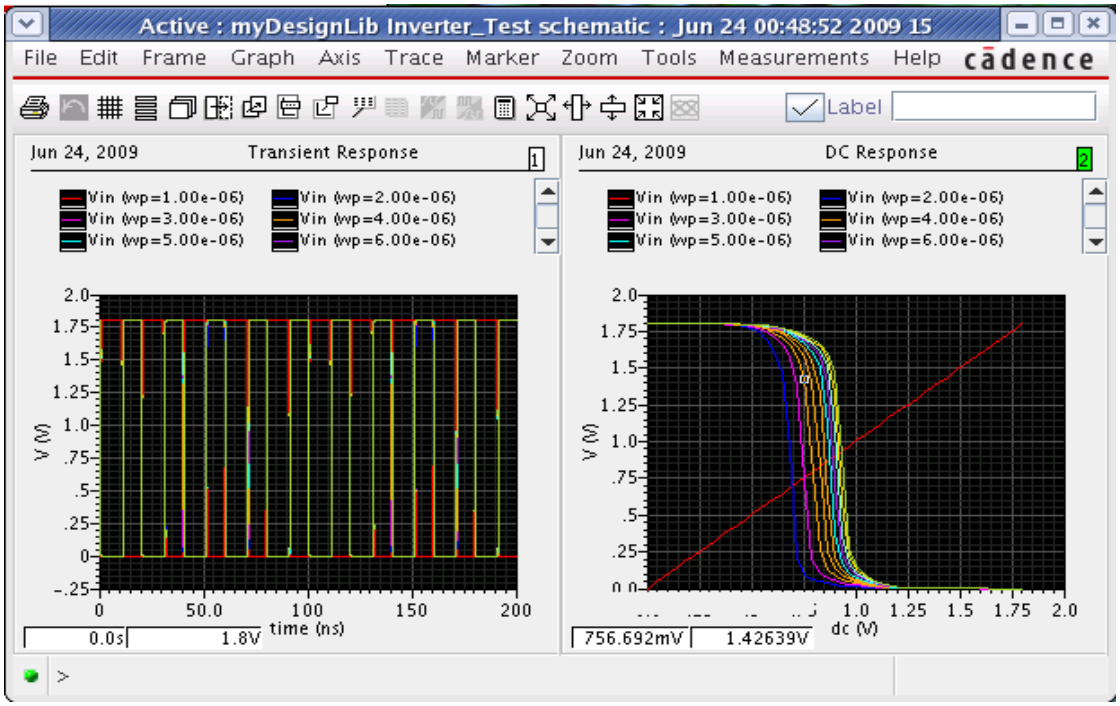
Range Type **From/To** From **1u** To **10u**
Step Control **Auto** Total Steps **10**

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



5. Execute **Analysis—Start**.

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 wavescan window comes up with the plots for different runs.


Active : myDesignLib Inverter_Test schematic : Jun 24 00:48:52 2009 15
File Edit Frame Graph Axis Trace Marker Zoom Tools Measurements Help
cadence
Label
Jun 24, 2009
Transient Response
1
Vin (wp=1.00e-06)
Vin (wp=2.00e-06)
Vin (wp=3.00e-06)
Vin (wp=4.00e-06)
Vin (wp=5.00e-06)
Vin (wp=6.00e-06)
V (V)
time (ns)
0.0s
1.8V
Jun 24, 2009
DC Response
2
Vin (wp=1.00e-06)
Vin (wp=2.00e-06)
Vin (wp=3.00e-06)
Vin (wp=4.00e-06)
Vin (wp=5.00e-06)
Vin (wp=6.00e-06)
V (V)
dc (V)
756.692mV
1.42639V

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. 

4.2.6 Creating Layout

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.

4.2.6.1 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

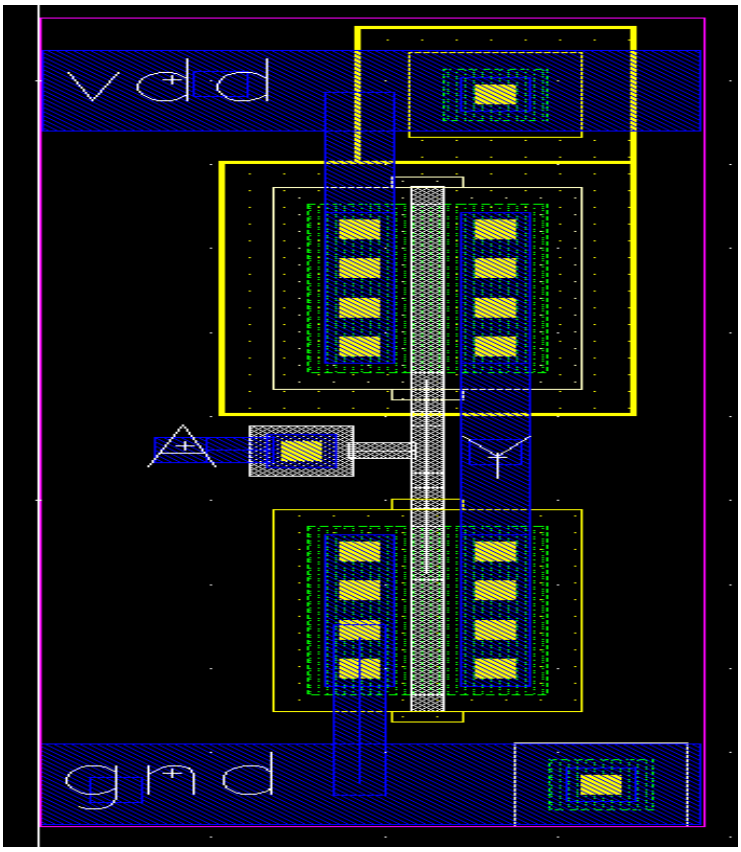
You will use the contacts or vias to make connections between two different layers.

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.



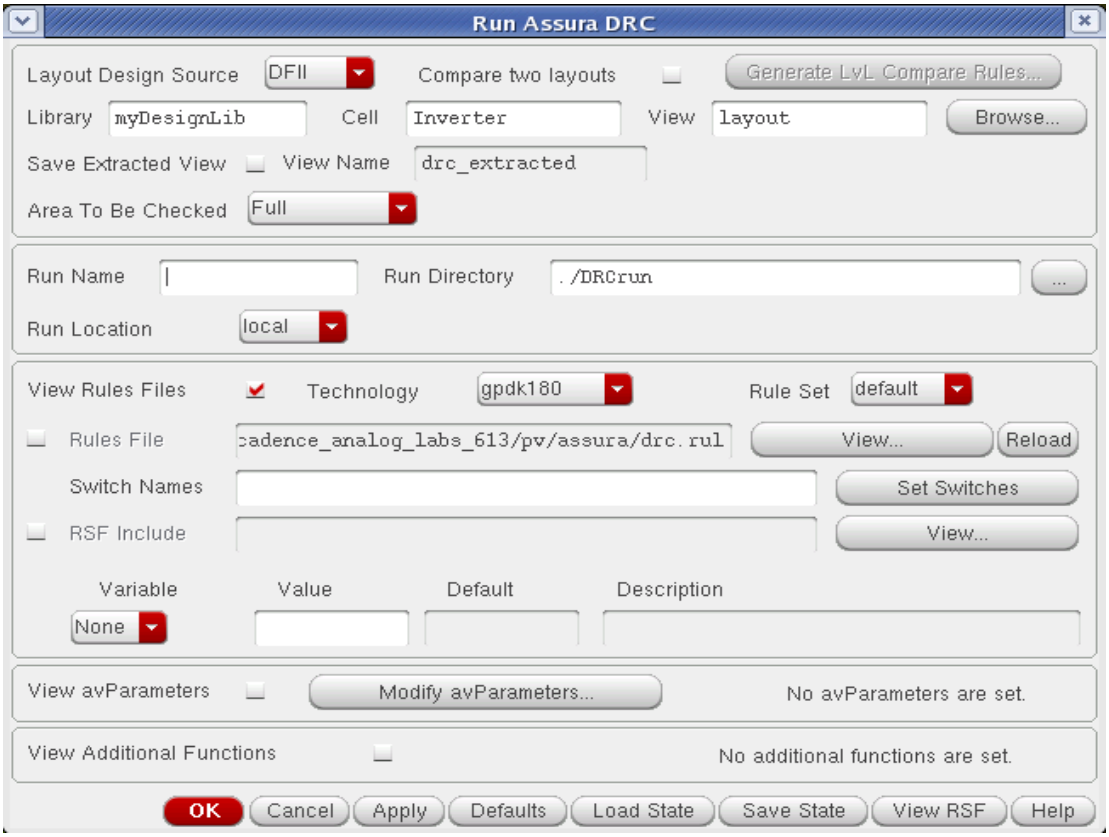
4.2.7 Physical Verification

4.2.7.1 Assura DRC

Running a DRC

1. Open the Inverter layout form 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 **gpdk180.** This automatically loads the rule file.

Your DRC form should appear like this



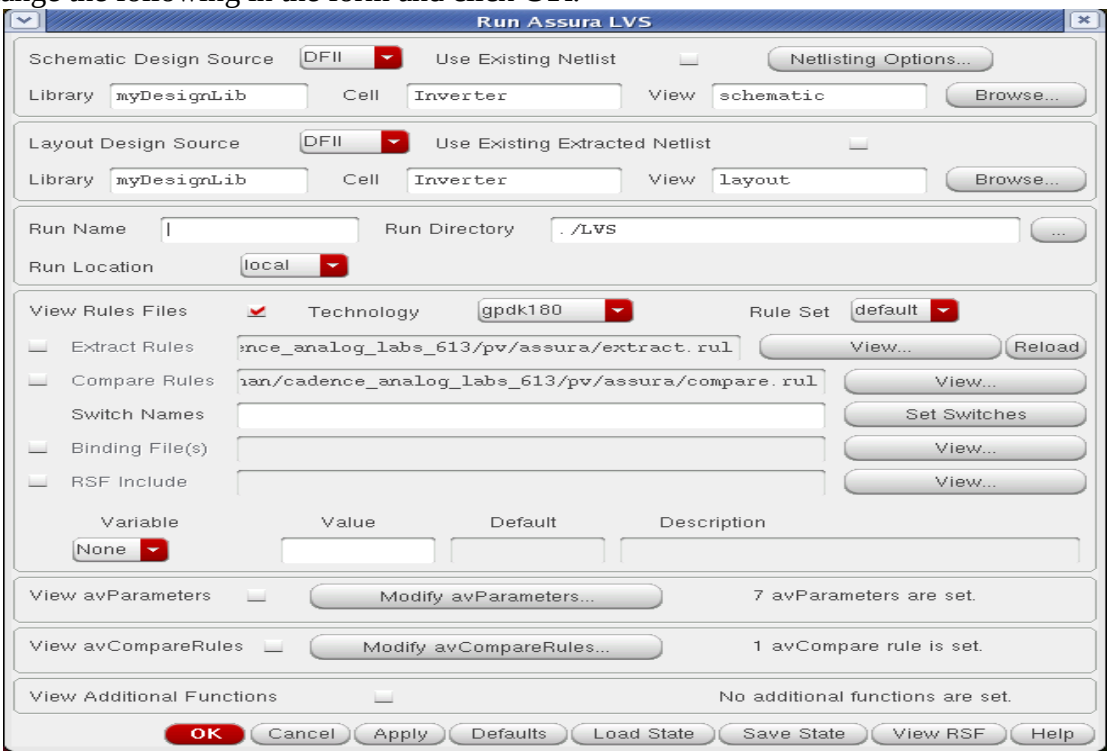
3. Click **OK** to start DRC.
4. A Progress form will appears. You can click on the watch log file to see the log file.
5. When DRC finishes, a dialog box appears asking you if you want to view your DRC results, and then click **Yes** to view the results of this run.
6. If there any DRC error exists 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 found** written in it, click on **close** to terminate the DRC run.

4.2.7.2 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**.



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.

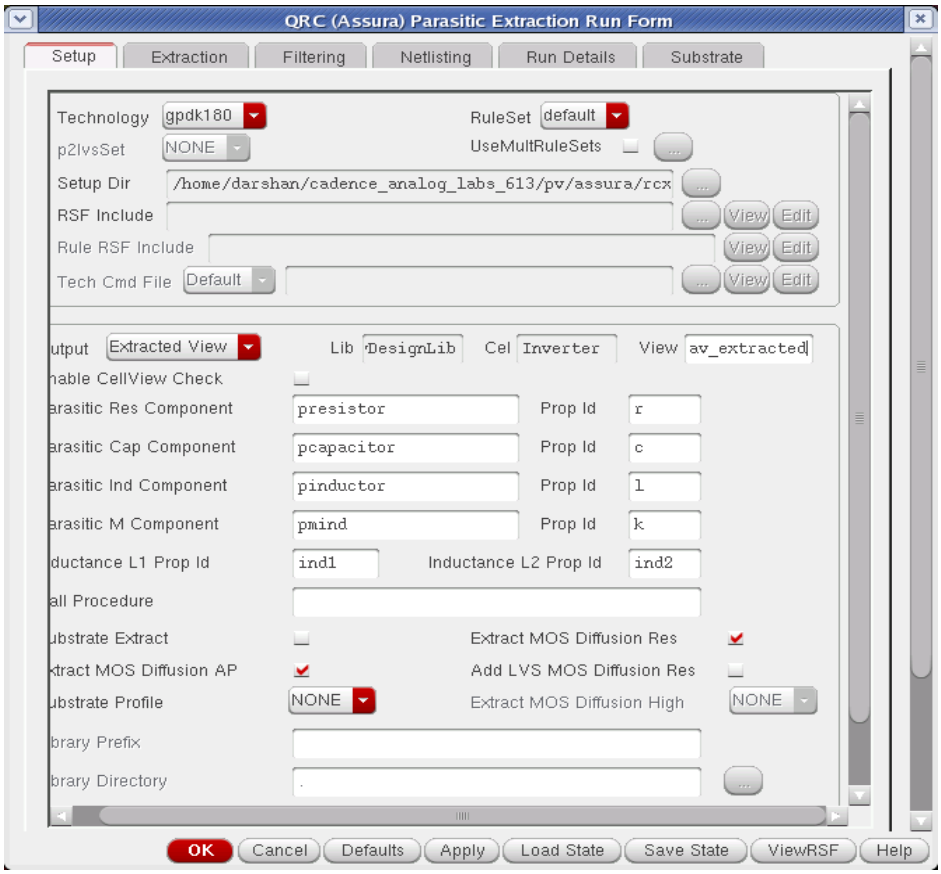
4.2.7.3 Assura RCX

In this section we will extract the RC values from the layout and perform analog circuit simulation on the designs extracted with RCX.

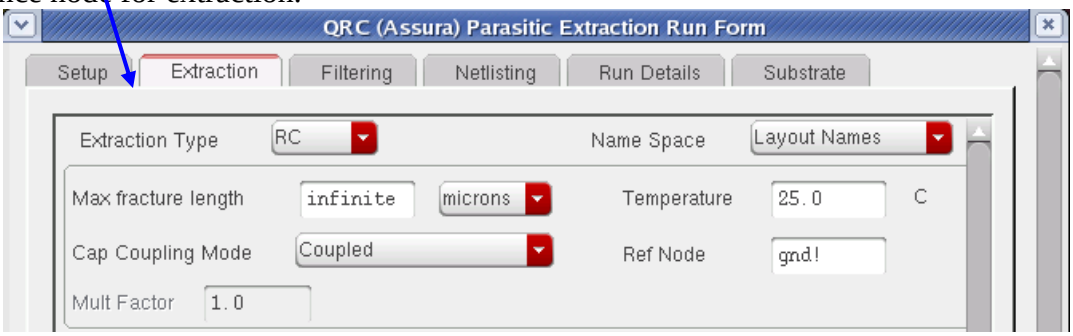
Before using RCX to extract parasitic devices for simulation, the layout should match with schematic completely to ensure that all parasites will be backannoted to the correct schematic nets.

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.



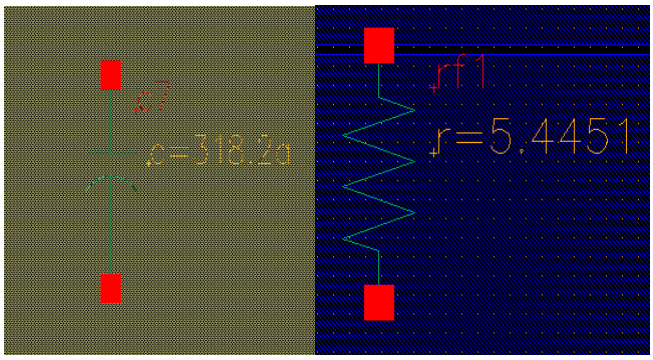
4. In the **Filtering** tab of the form, **Enter Power Nets** as **vdd!, vss!** and **Enter Ground Nets** as **gnd!**

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.**

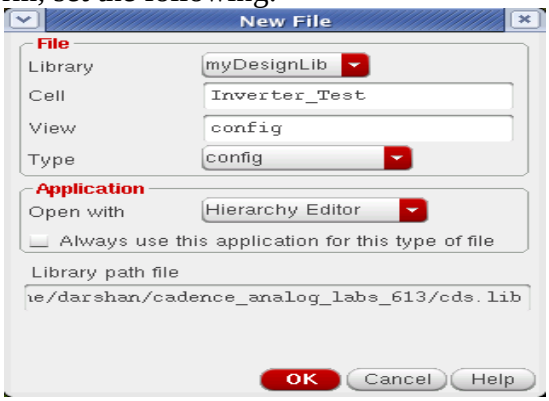
6. You can open the **av_extracted** view from the library manager and view the parasitic.



4.2.8 Creating the Configuration View

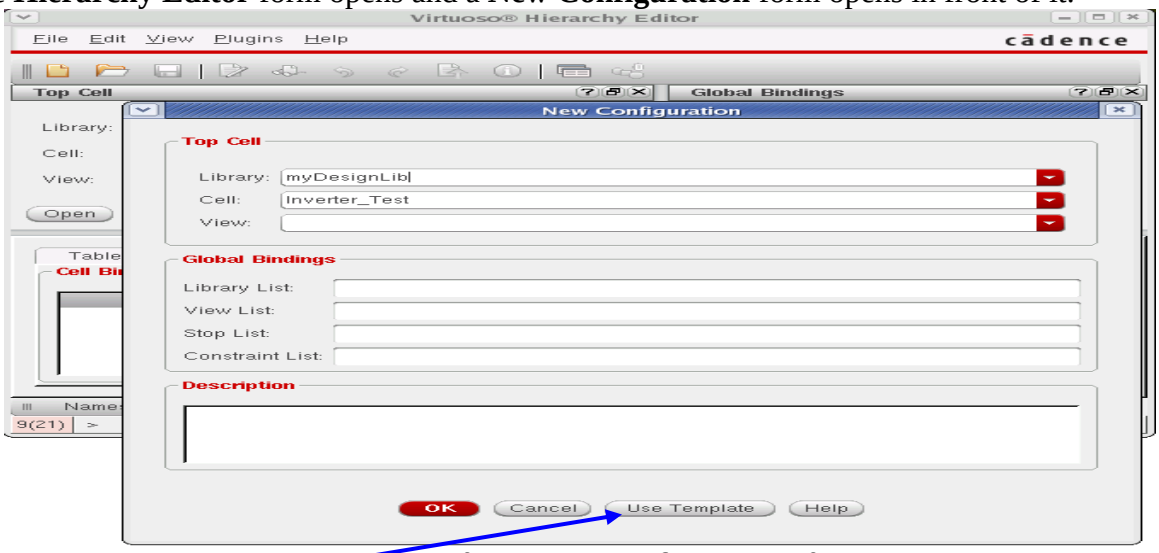
In this section we will create a config view and with this config view we will run the simulation with and without parasitic.

1. In the CIW or Library Manager, execute **File – New – Cellview**
2. In the Create New file form, set the following:



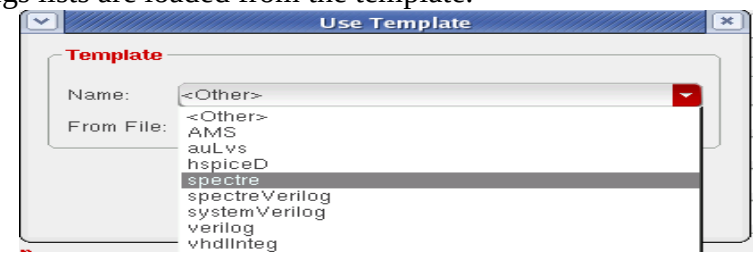
3. Click **OK** in create **New File** form.

The **Hierarchy Editor** form opens and a **New Configuration** form opens in front of it.



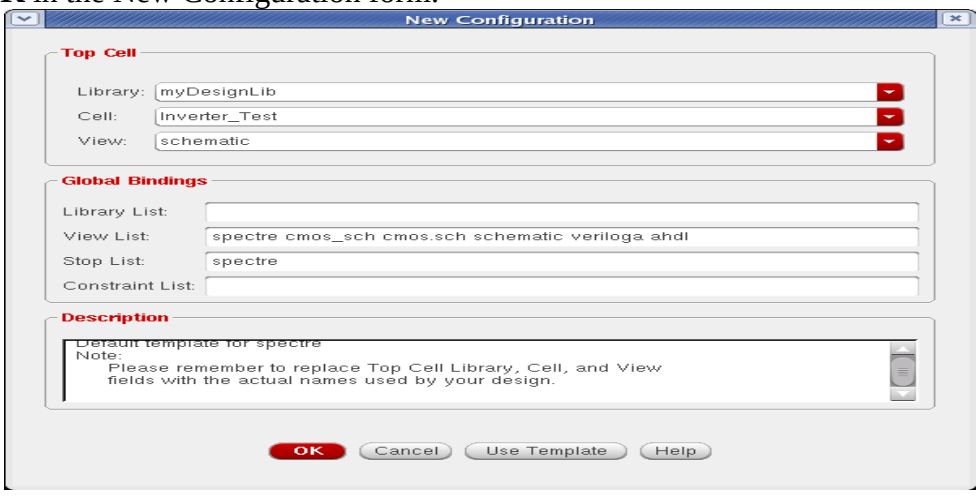
4. Click **Use template** at the bottom of the **New Configuration** form and select **Spectre** in the cyclic field and click **OK**.

The Global Bindings lists are loaded from the template.

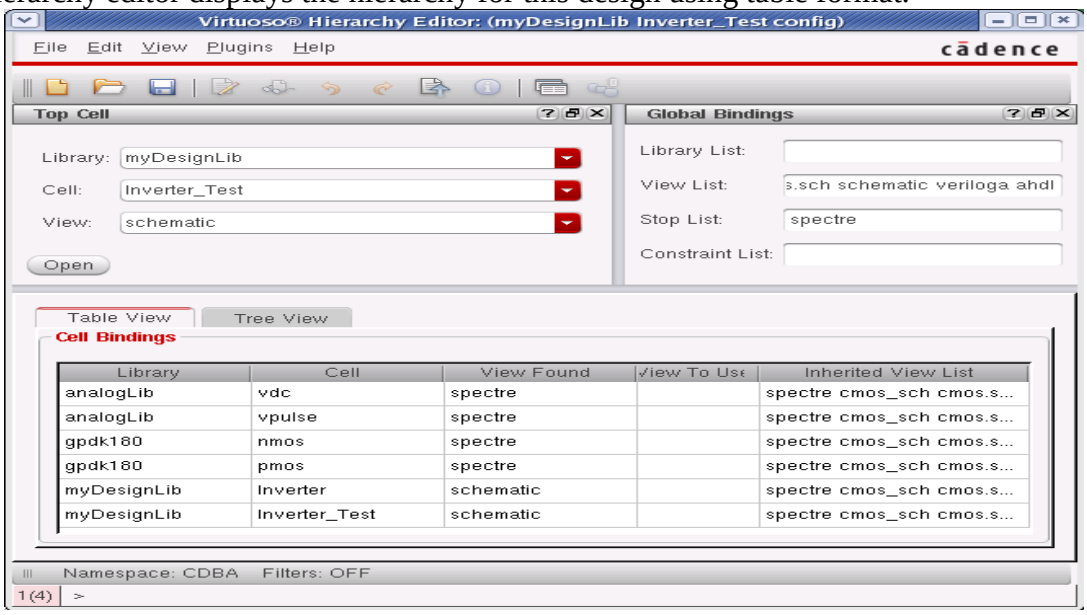


5. Change the **Top Cell** View to **schematic** and remove the default entry from the **Library List** field.

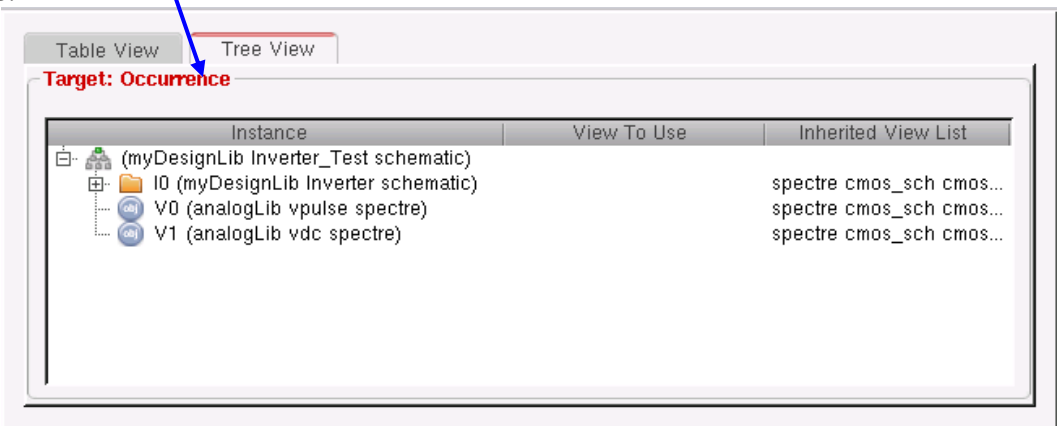
6. Click **OK** in the New Configuration form.



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



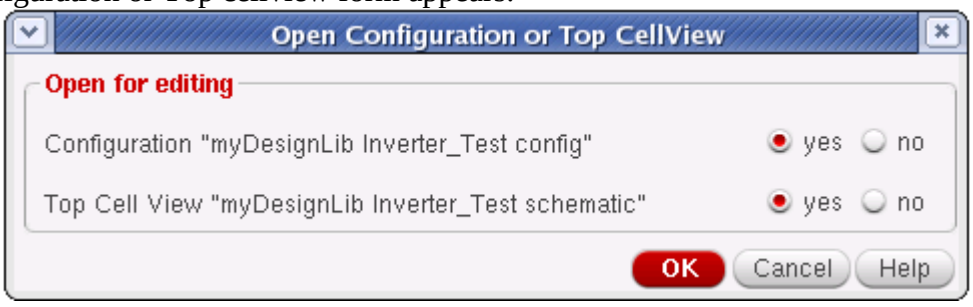
7. Click the **Tree View** tab. The design hierarchy changes to tree format. The form should look like this:



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_Test** Config view.
Open Configuration or Top cellview form appears.



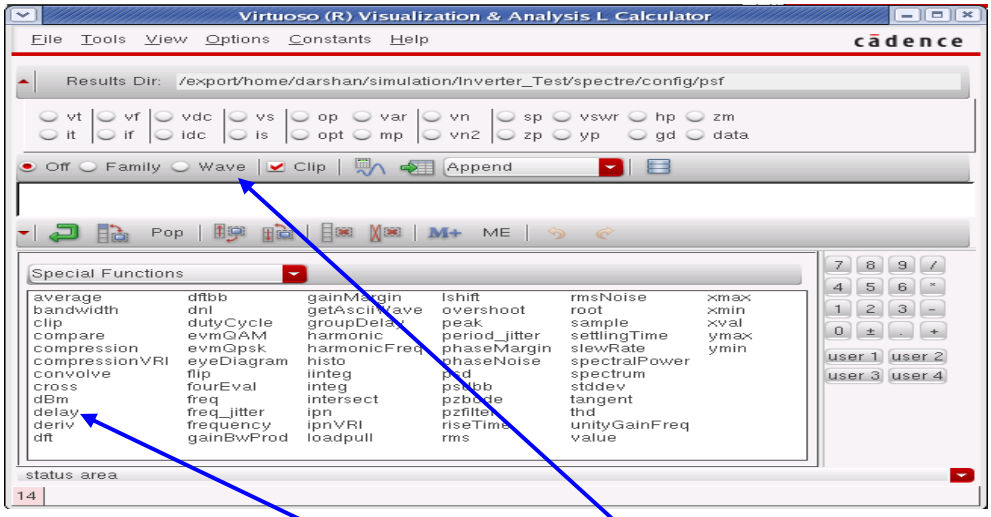
2. In the form, turn on the both cyclic buttons to **Yes** and click **OK.**

The Inverter_Test schematic and Inverter_Test config window appears. Notice the window banner of schematic also states **Config: myDesignLib Inverter_Test config.**
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.



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.

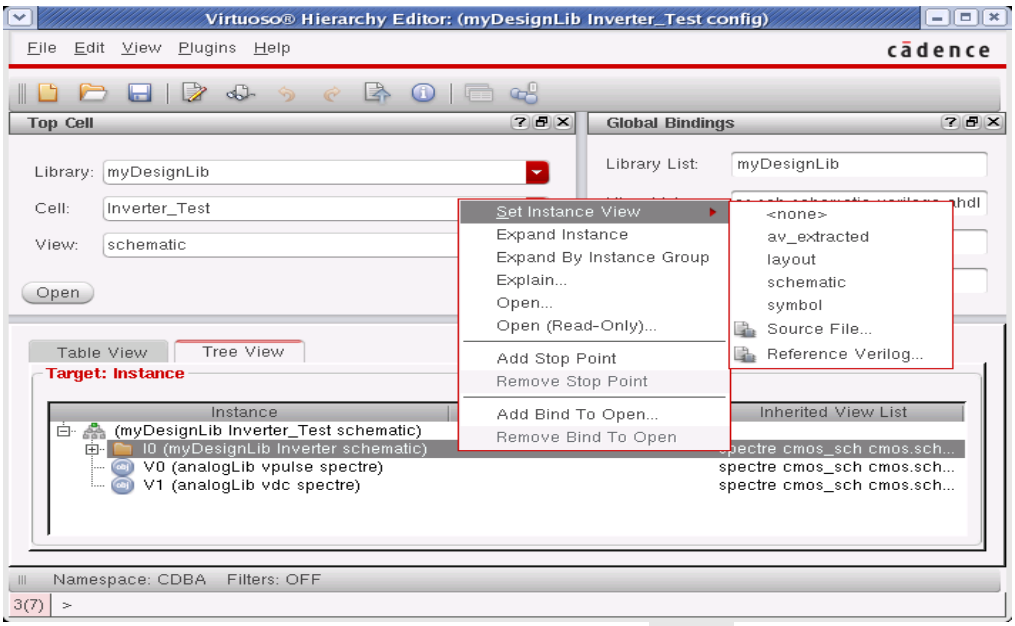
7. Click on **Evaluate the buffer icon**  to perform the calculation, note down the value returned after execution.
8. Close the calculator window.

To run the Circuit with Parasitics

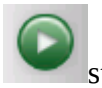
In this exercise, we will change the configuration to direct simulation of the **av_extracted** view which contains the parasites.

1. Open the same Hierarchy Editor form, which is already set for Inverter_Test config.
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.



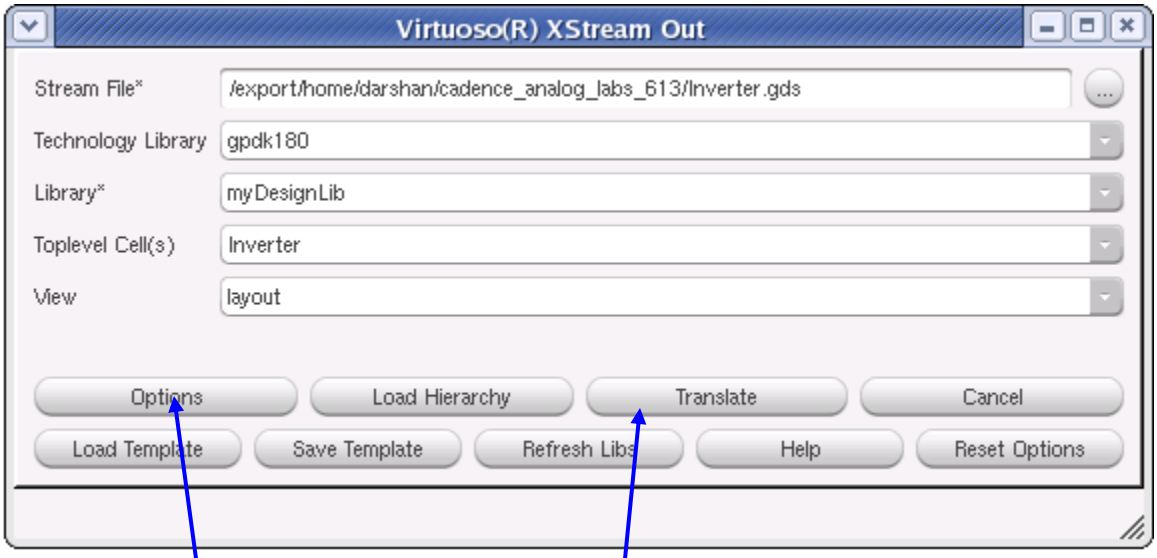
4. Click on the **Recompute the hierarchy** icon,  the configuration is now updated from schematic to av_extracted view.

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.

4.3 Generating Stream Data

Streaming Out the Design

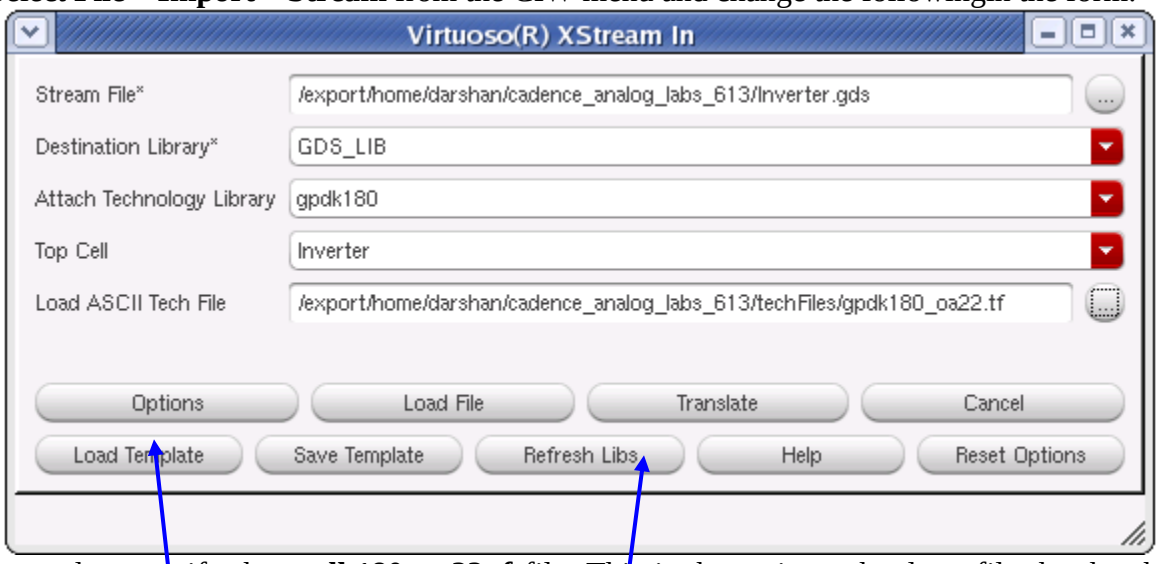
1. Select **File – Export – Stream** from the CIW menu and **Virtuoso Xstream out** form appears change the following in the form.



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.

4.4 Streaming In the Design

1. Select **File – Import – Stream** from the CIW menu and change the followingin the form.



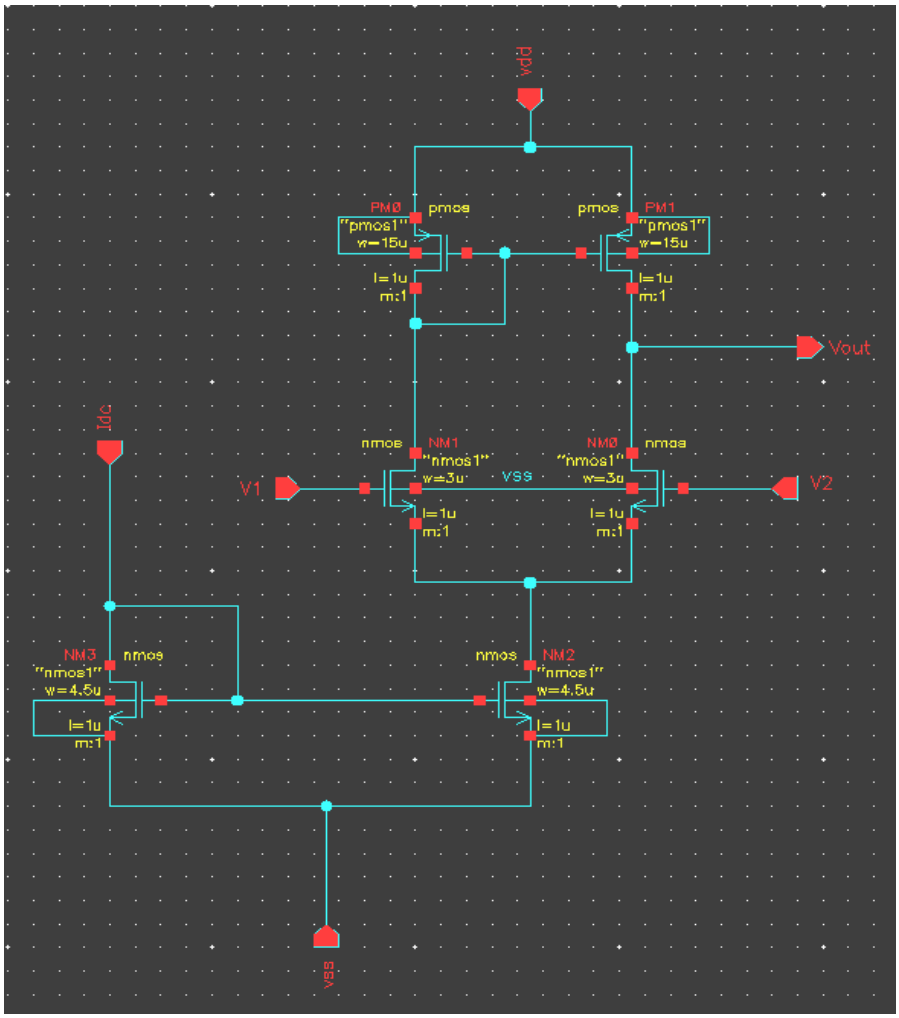
You need to specify the **gpdk180_oa22.tf** file. This is the entire technology file that has been dumped from the design library.

2. Click on the **Options** button.

3. In the **StreamOut-Options** form select 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. 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.

4.5 Tutorial 2: DIFFERENTIAL AMPLIFIER

Schematic Capture



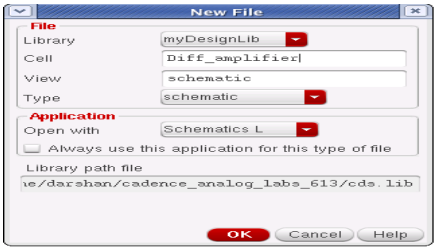
4.5.1 Schematic Entry

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 Differntial_Amplifier design.

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



2. 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.

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/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.

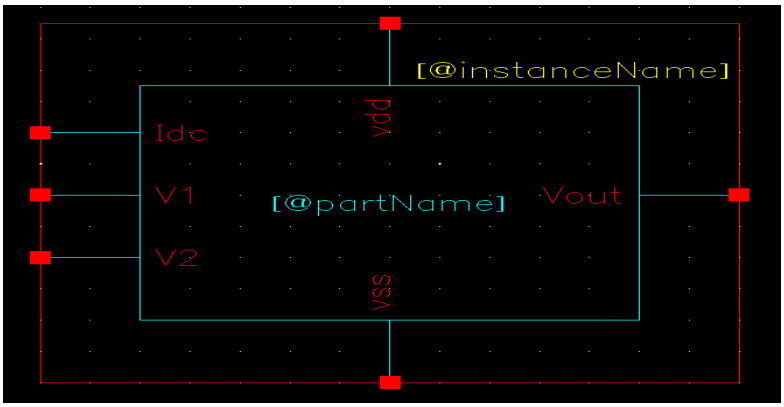
In the schematic window, execute **View— Fit** or press the **f** bindkey.

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.

4.5.2 Symbol Creation

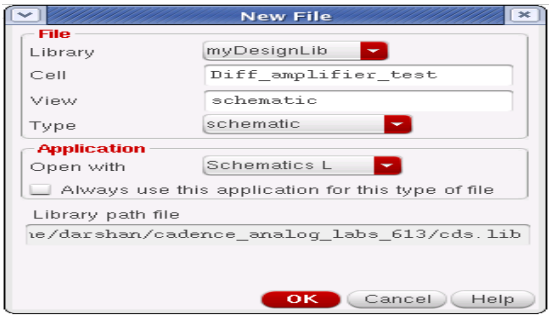
Objective: To create a symbol for the Differential Amplifier
1. In the Differential Amplifier schematic window, execute **Create — Cellview— From Cellview**.
The **Cellview from Cellview** form appears. With the Edit Options function active, you can control the appearance of the symbol to generate.
2. 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.

4.5.3 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:

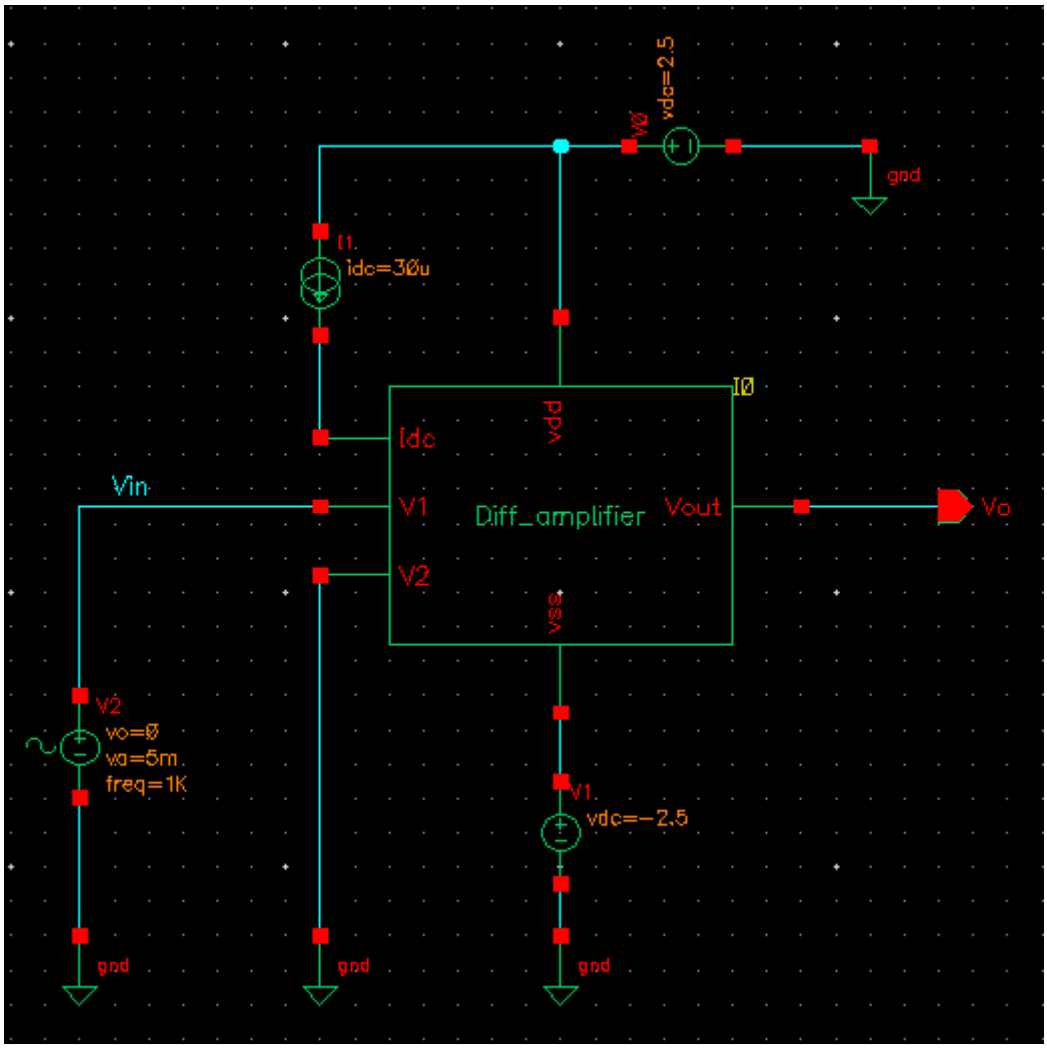


3. Click **OK** when done. A blank schematic window for the Diff_ amplifier_test design appears.

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	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.

4.5.4 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**.

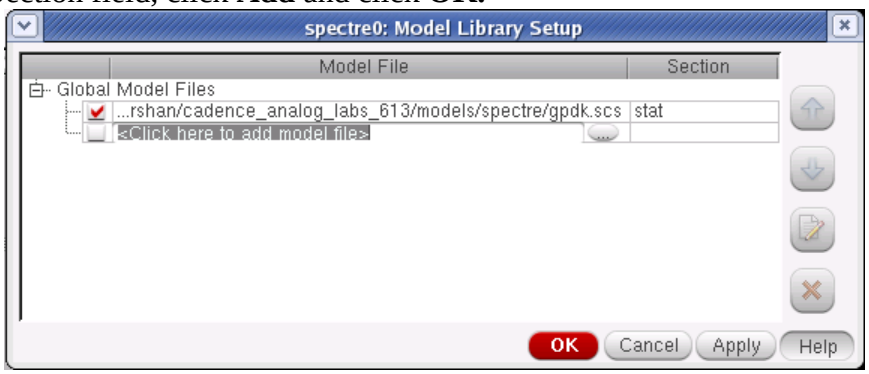
Setting the Model Libraries

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 **gpdk180.scs** file in the **./models/spectre** directory.

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



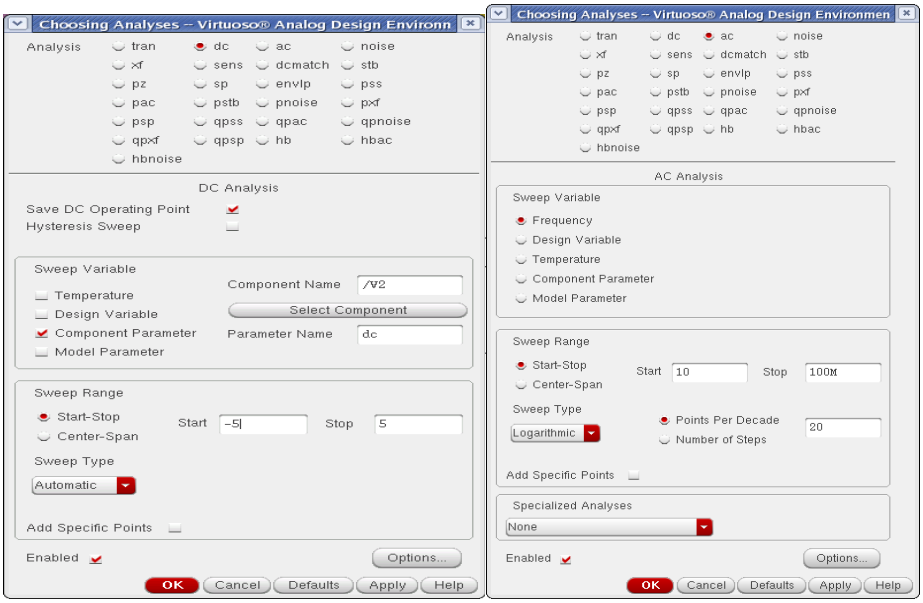
Choosing Analyses

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
    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 **Vsin** for 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**.


Choosing Analyses -- Virtuoso® Analog Design Environ
Analysis
tran
dc
ac
noise
xf
sens
dcmatch
stb
pz
sp
envlp
pss
pac
pstb
pnoise
pxf
psp
qpss
qpac
qpnoise
qpxf
qpsp
hb
hbac
hbnoise
DC Analysis
Save DC Operating Point
Hysteresis Sweep
Sweep Variable
Temperature
Design Variable
Component Parameter
Model Parameter
Component Name
/V2
Select Component
Parameter Name
dc
Sweep Range
Start-Stop
Center-Span
Start
-5
Stop
5
Sweep Type
Automatic
Add Specific Points
Enabled
Options...
OK
Cancel
Defaults
Apply
Help
Choosing Analyses -- Virtuoso® Analog Design Environmen
Analysis
tran
dc
ac
noise
xf
sens
dcmatch
stb
pz
sp
envlp
pss
pac
pstb
pnoise
pxf
psp
qpss
qpac
qpnoise
qpxf
qpsp
hb
hbac
hbnoise
AC Analysis
Sweep Variable
Frequency
Design Variable
Temperature
Component Parameter
Model Parameter
Sweep Range
Start-Stop
Center-Span
Start
10
Stop
100M
Sweep Type
Logarithmic
Points Per Decade
Number of Steps
20
Add Specific Points
Specialized Analyses
None
Enabled
Options...
OK
Cancel
Defaults
Apply
Help

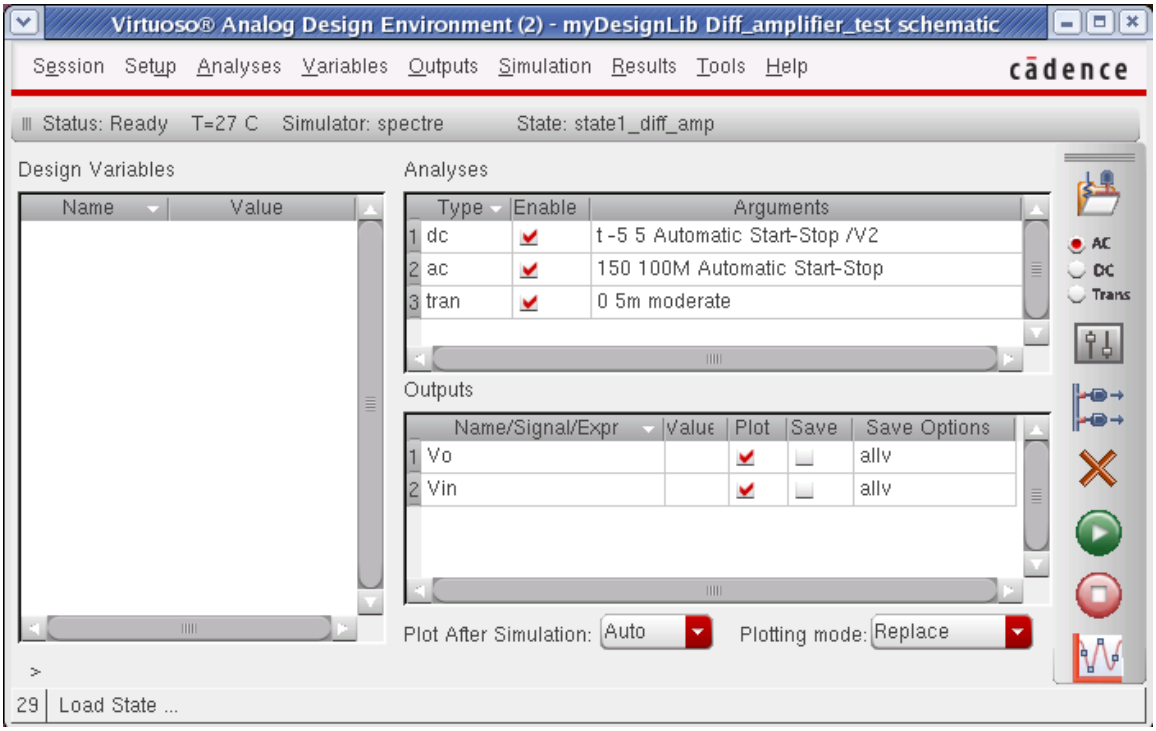
4. To set up for AC Analyses form is shown in the previous page.
    a. In the Analyses section, select **ac**.
    b. In the AC Analyses section, turn on **Frequency.**
    c. In the Sweep Range section select **start** and **stop** frequencies as **150** to **100M**
    d. Select Points per Decade 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.

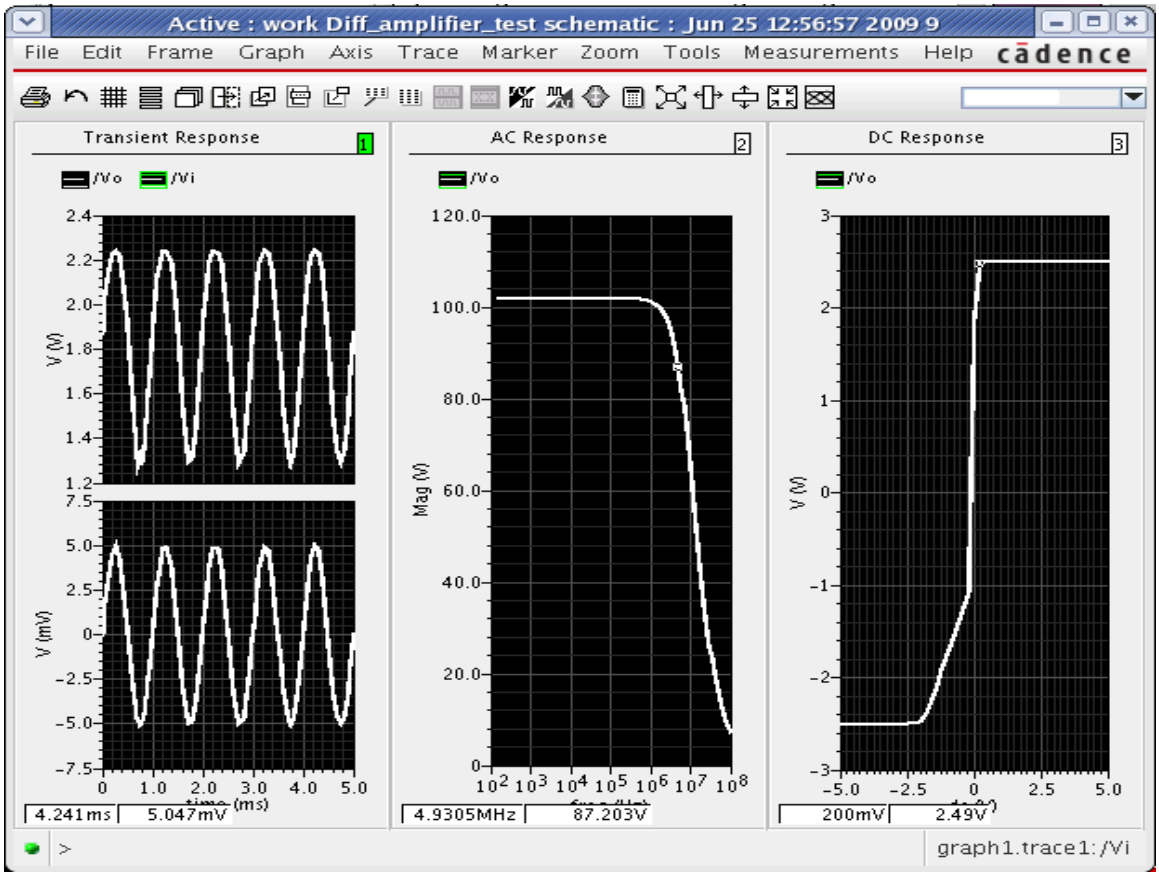
1. Execute **Outputs – To be plotted – Select on Schematic** in the simulation window.
2. Follow the prompt at the bottom of the schematic window, Click on output net **Vo,** input net **Vin** of the Diff_amplifier. Press **ESC** with the cursor in the schematic after selecting node.

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.



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.

4.5.5 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

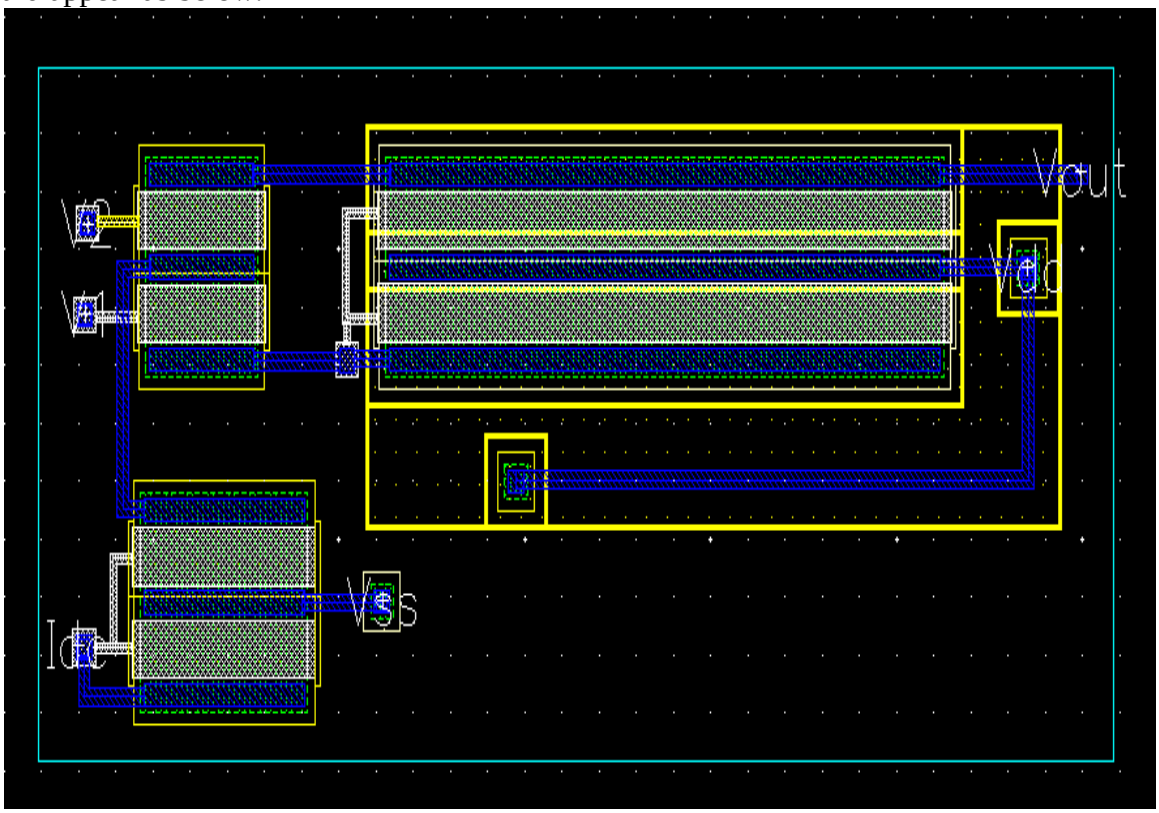
You will use the contacts or vias to make connections between two different layers.

1. Execute **Create — 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.



4.5.6 Physical Verification

4.5.6.1 Assura DRC

Running a DRC

1. Open the Differential_Amplifier layout form 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 **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.


"Diff_amplifier" has completed SUCCESSFULLY!
The DRC run "Diff_amplifier" has completed successfully.
Do you want to view the results of this run?
Yes
No
Help

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.

4.5.6.2 ASSURA LVS

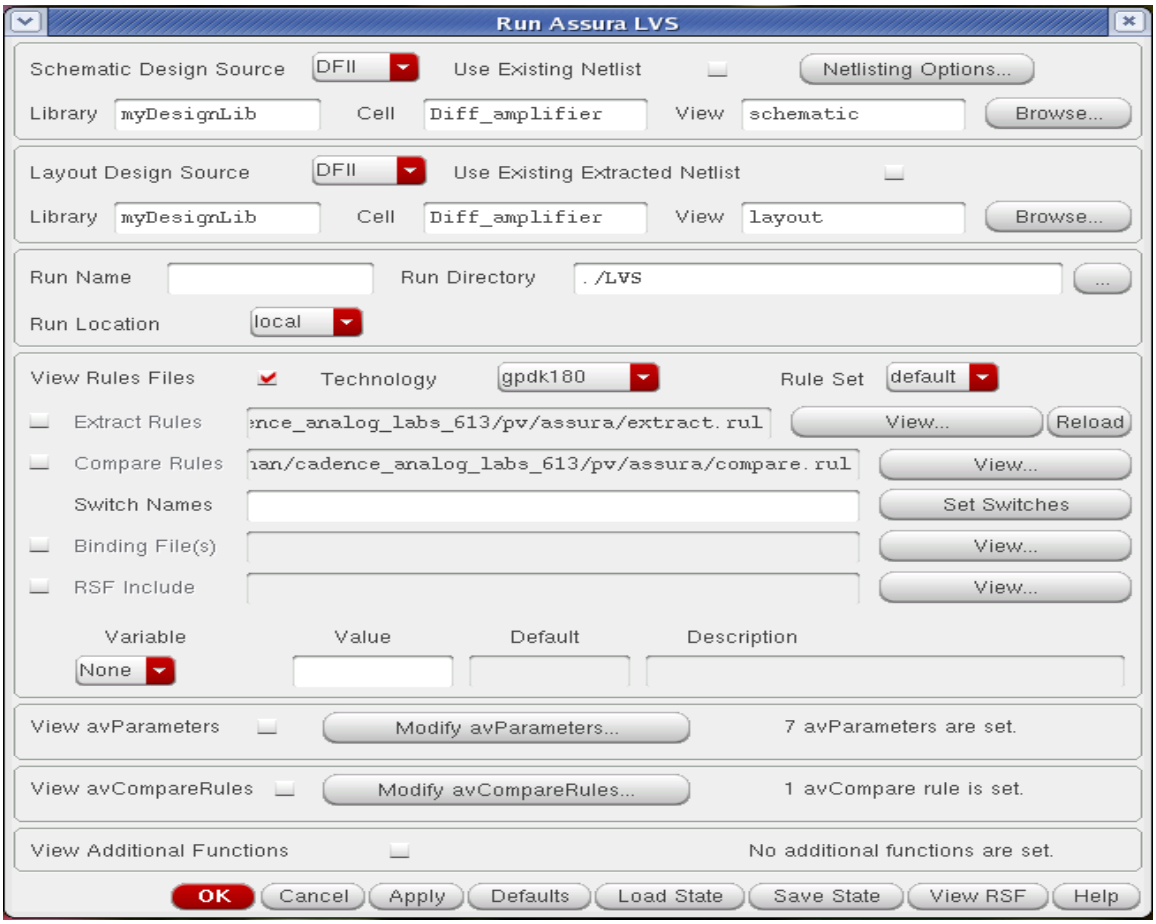
In this section we will perform the LVS check that will compare the schematic netlist and the layout netlist.

Running LVS

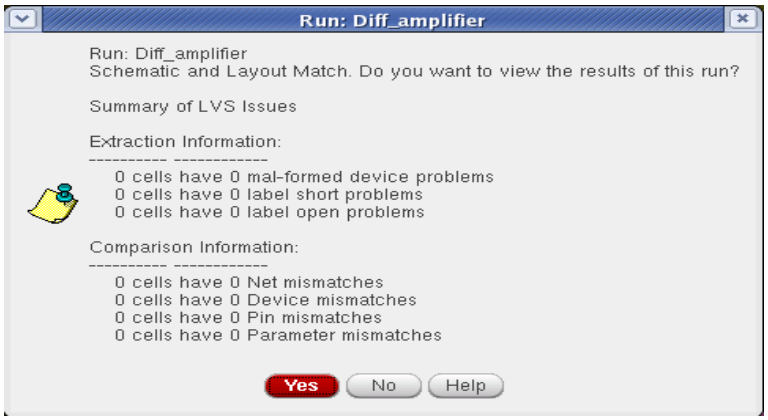
1. Select **Assura – Run LVS** from the layout window.

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.

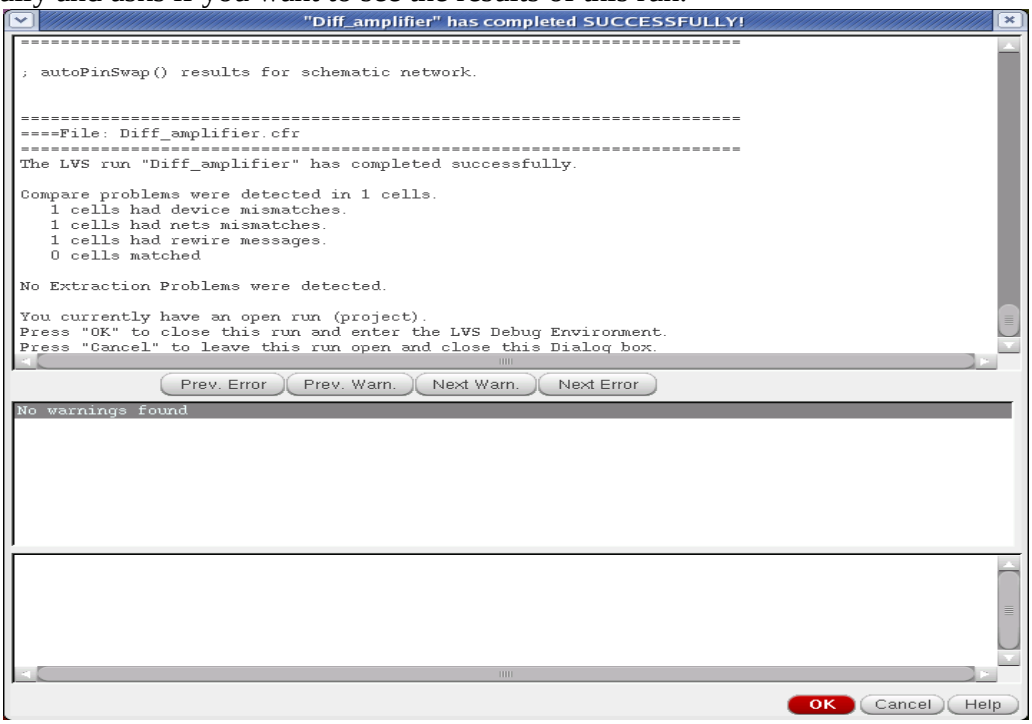
2. Verify the following in the Run Assura LVS form.


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
gpdk180
Rule Set
default
Extract Rules
ence_analog_labs_613/pv/assura/extract.rul
View...
Reload
Compare Rules
ian/cadence_analog_labs_613/pv/assura/compare.rul
View...
Switch Names
Set Switches
Binding File(s)
View...
RSF Include
View...
Variable
Value
Default
Description
None
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.



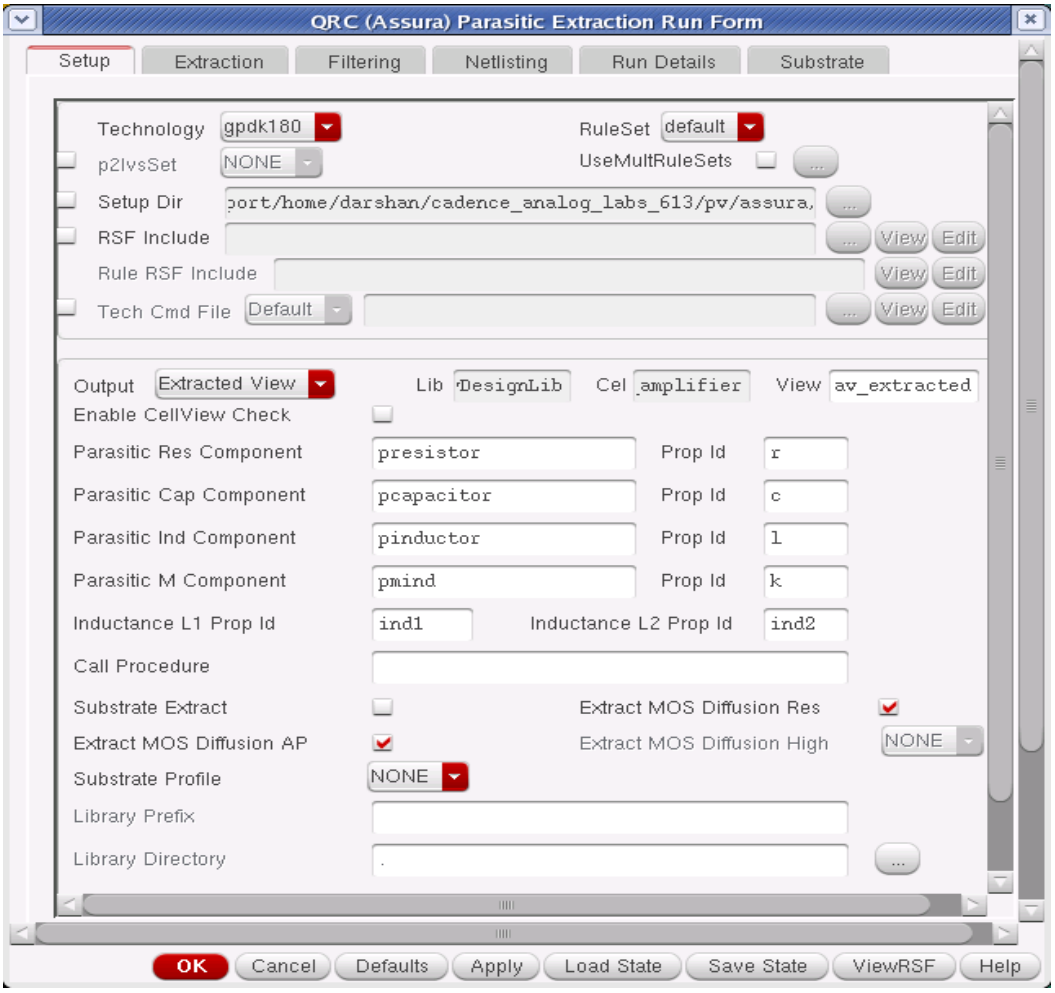
6. Click **Yes** in the form.LVS debug form appears, and you are directed into LVS debug environment.
7. In the **LVS debug form** you can find the details of mismatches and you need to correct all those mismatches and **Re – run** the LVS till you will be able to match the schematic with layout.

4.5.6.3 Assura RCX

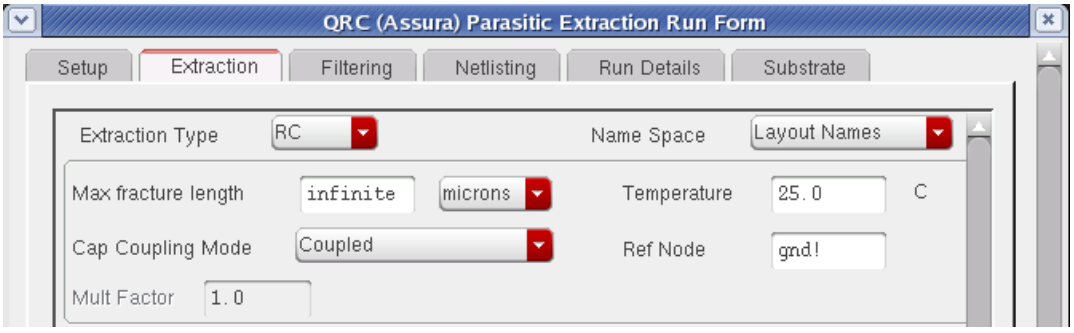
In this section we will extract the RC values from the layout and perform analog circuit simulation on the designs extracted with RCX.
Before using RCX to extract parasitic devices for simulation, the layout should match with schematic completely to ensure that all parasites will be backannoted to the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

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

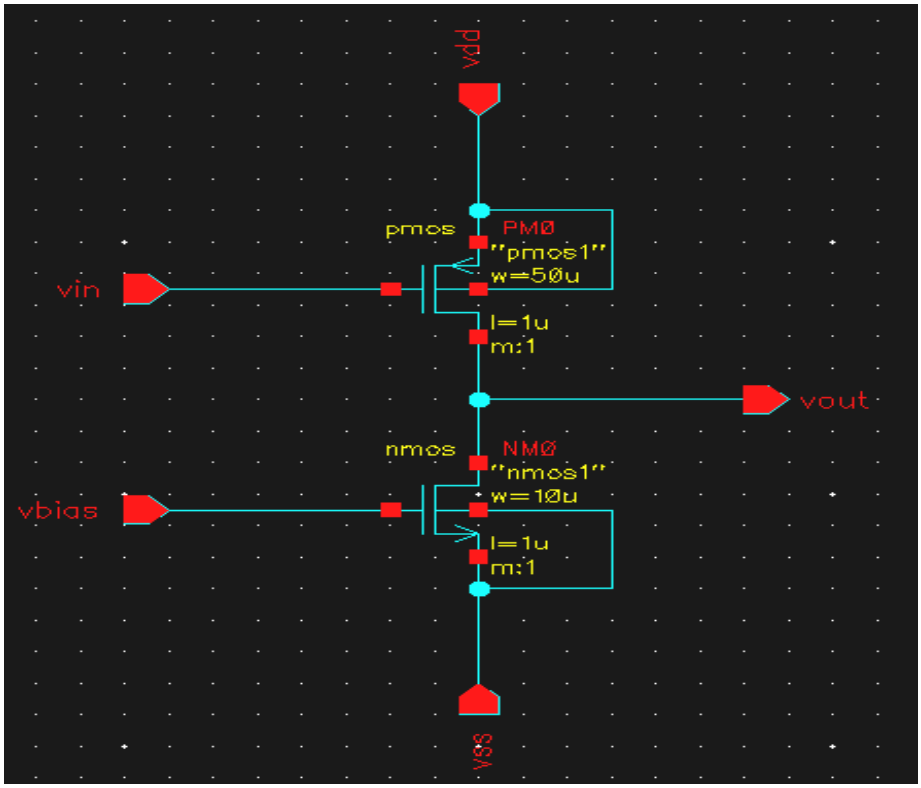
The RCX progress form appears, in the progress form click **Watch log file** to see the output log file.

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.

4.6 Tutorial3: COMMON SOURCE AMPLIFIER

Schematic Capture



4.6.1 Schematic Entry

Objective: To create a new cell view and build Common Source Amplifier
Use the techniques learned in the Tutotial1 and Tutorial2 to complete the schematic of Common Source Amplifier. This is a table of components for building the Common Source Amplifier schematic.

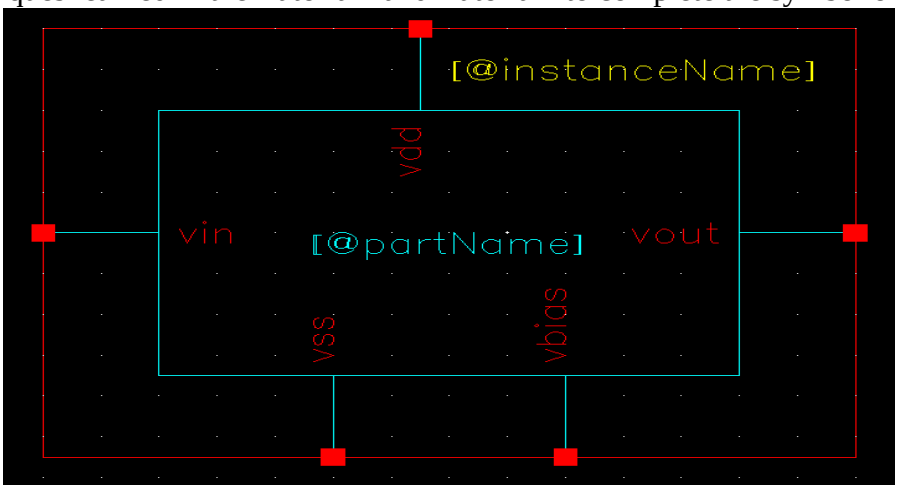
Library name	Cell Name	Properties/Comments
gpdk180	pmos	Model Name = pmos1; W= 50u ; L= 1u
gpdk180	nmos	Model Name =nmos1; W= 10u ; L= 1u

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
vdd vss	Input

4.6.2 Symbol Creation

Objective: To create a symbol for the Common Source Amplifier
Use the techniques learned in the Tutorial1 and Tutorial2 to complete the symbol of cs-amplifier

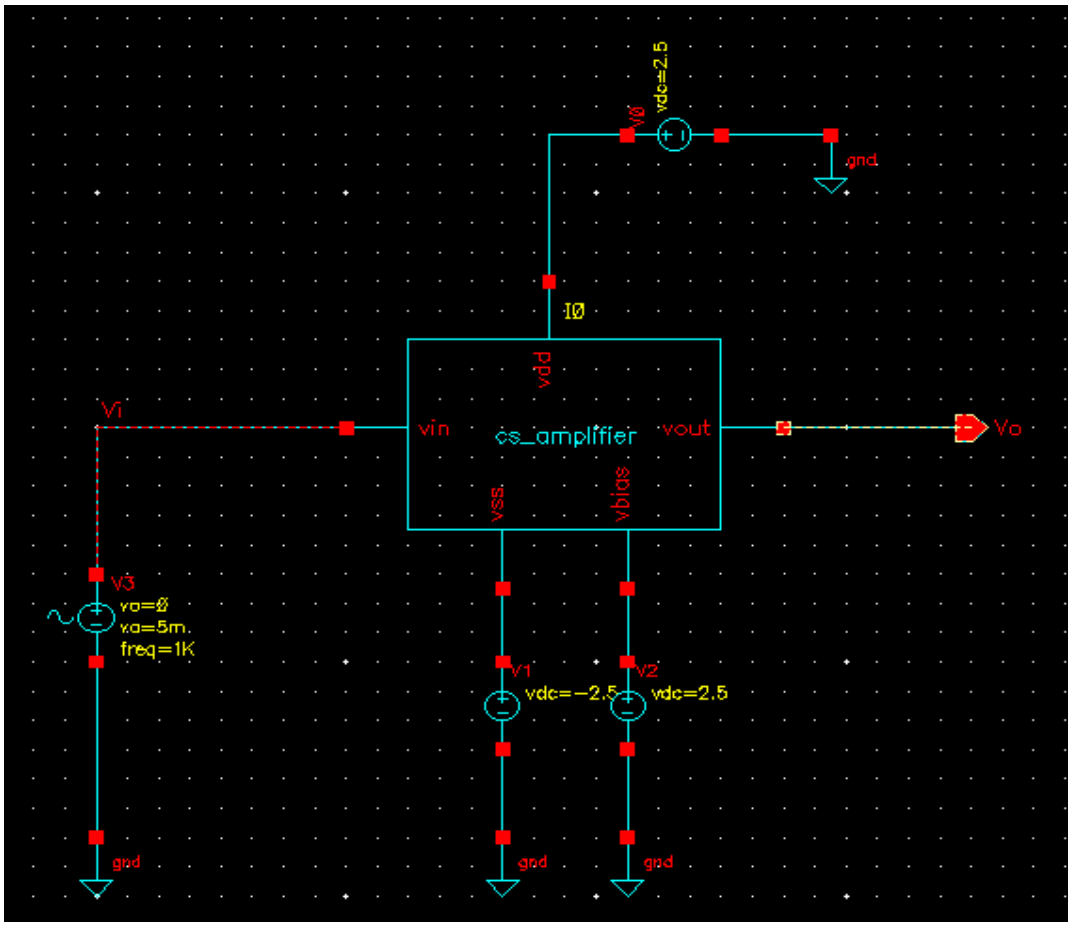


4.6.3 Building the Common Source Amplifier Test Design

Objective: To build cs_amplifier_test circuit using your cs_amplifier

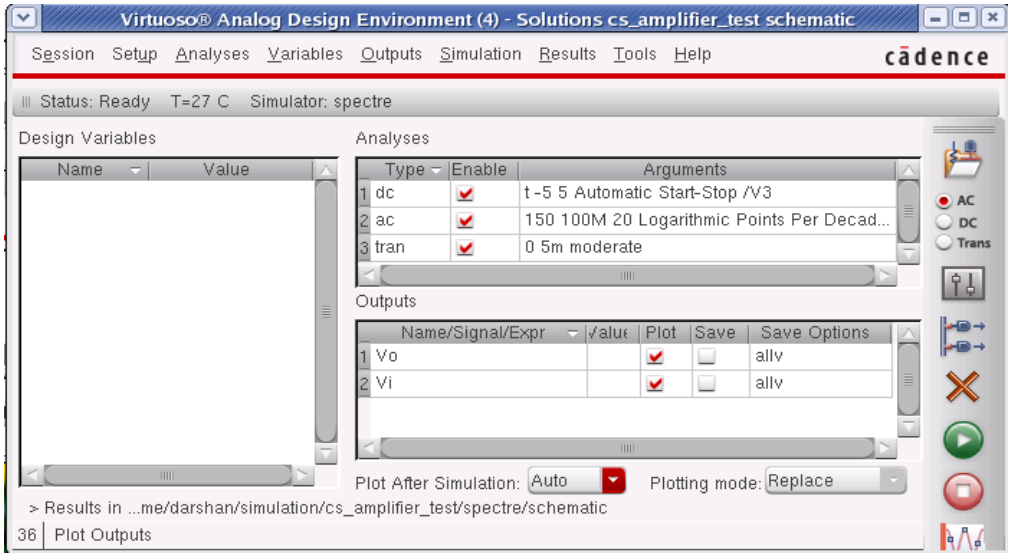
Using the component list and Properties/Comments in the table, build the cs-amplifier_test schematic as shown below.

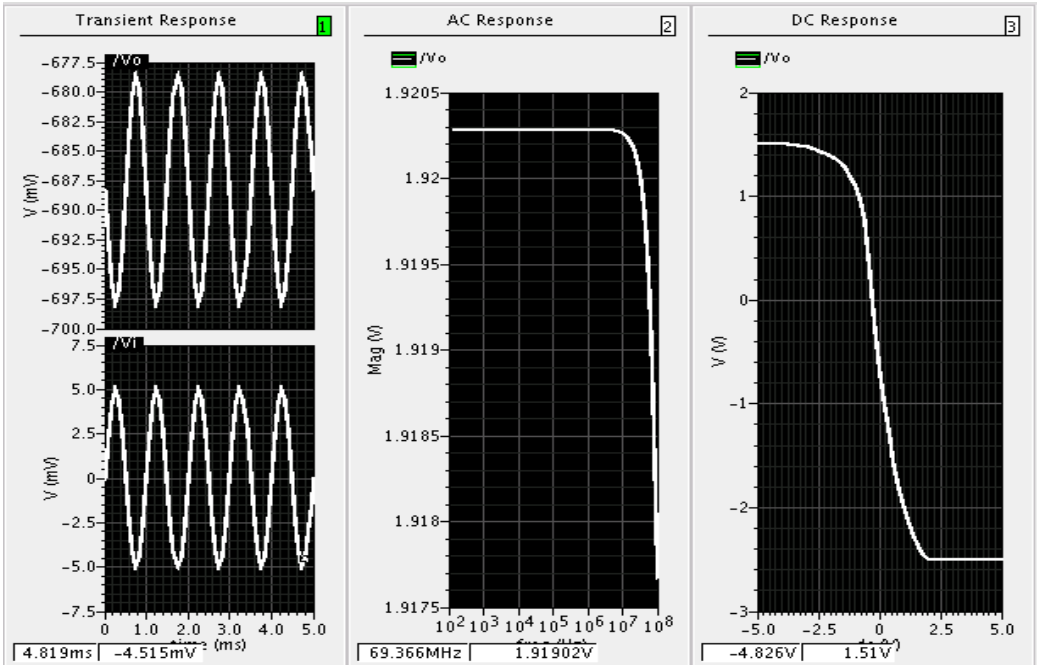
Library name	Cellview name	Properties/Comments
myDesignLib	cs_amplifier	Symbol
analogLib	vsin	Define pulse specification as AC Magnitude= 1; DC Voltage= 0; Offset Voltage= 0; Amplitude= 5m; Frequency= 1K
analogLib	vdd,vss,gnd	vdd=2.5 ; vss= -2.5



4.6.4 Analog Simulation with Spectre

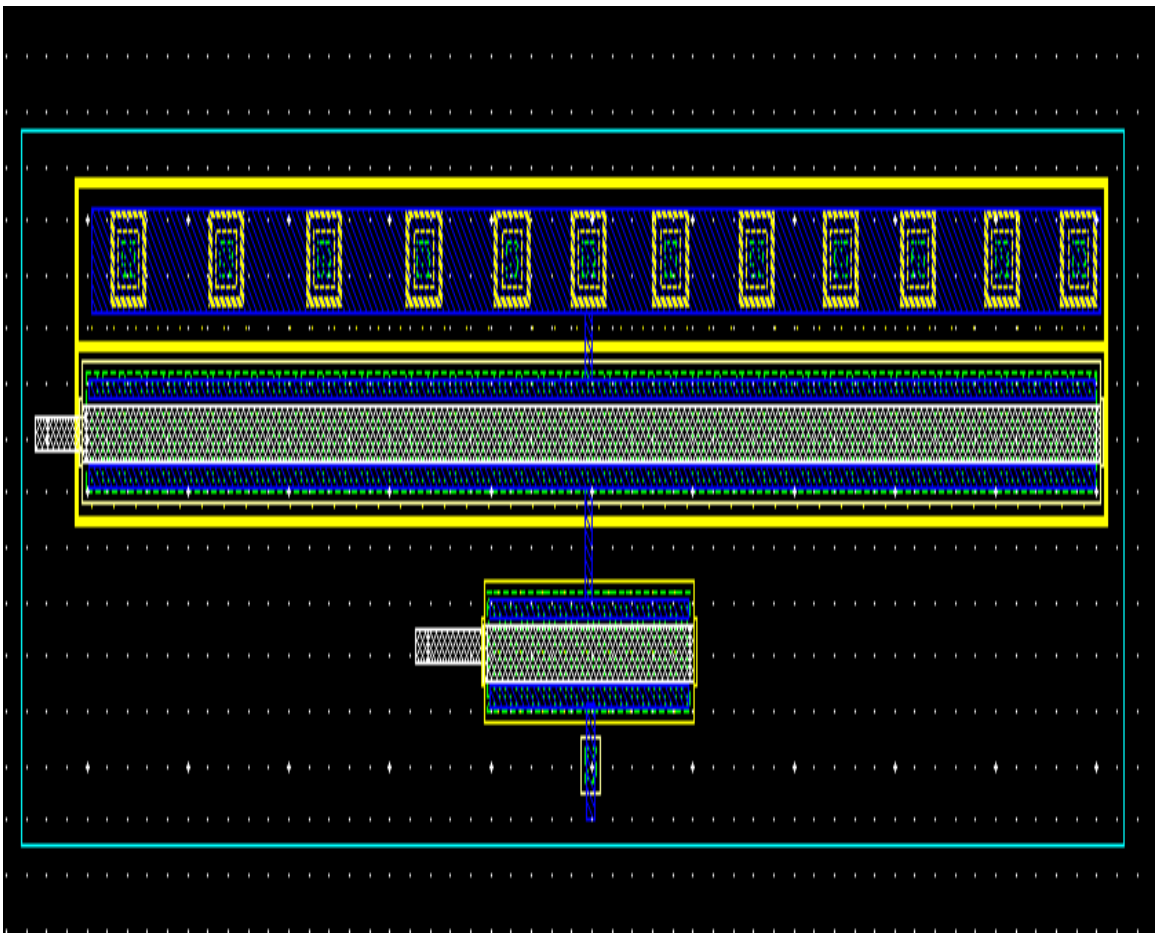
Objective: To set up and run simulations on the cs_amplifier_test design.
Use the techniques learned in the Lab1 and Lab2 to complete the simulation of cs_amplifier, ADE window and waveform should look like below.





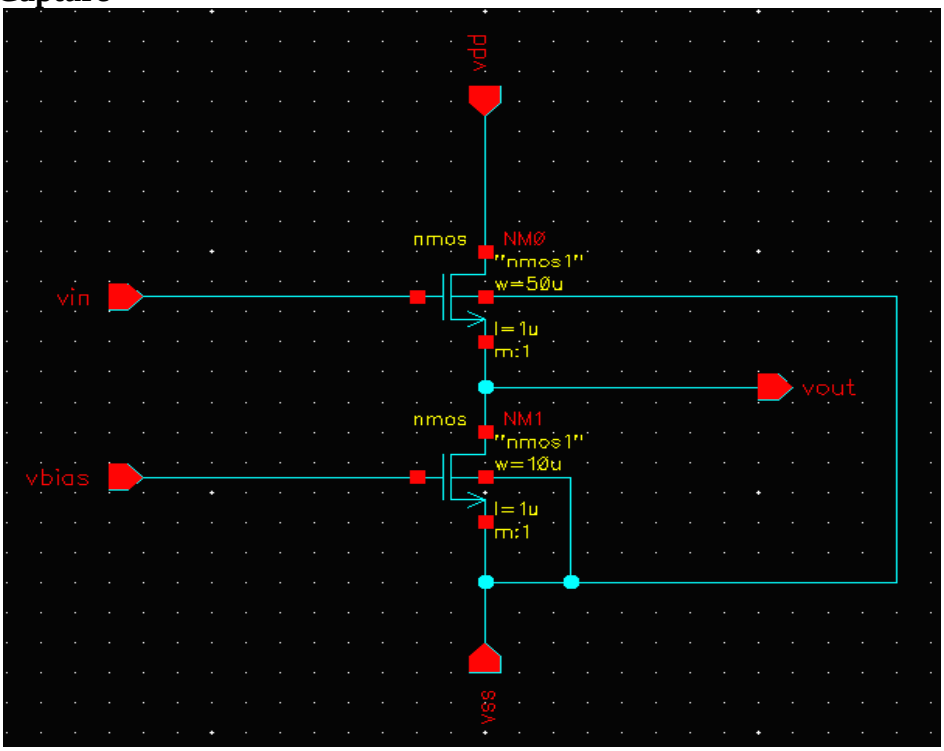
4.6.5 Creating a layout view of Common Source Amplifier

Use the techniques learned in the Tutorial1 and Tutorial2 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.



4.7 Tutorial4: COMMON DRAIN AMPLIFIER

Schematic Capture



4.7.1 Schematic Entry

Objective: To create a new cell view and build Common Drain Amplifier
Use the techniques learned in the Tutorial1 and Tutorial2 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

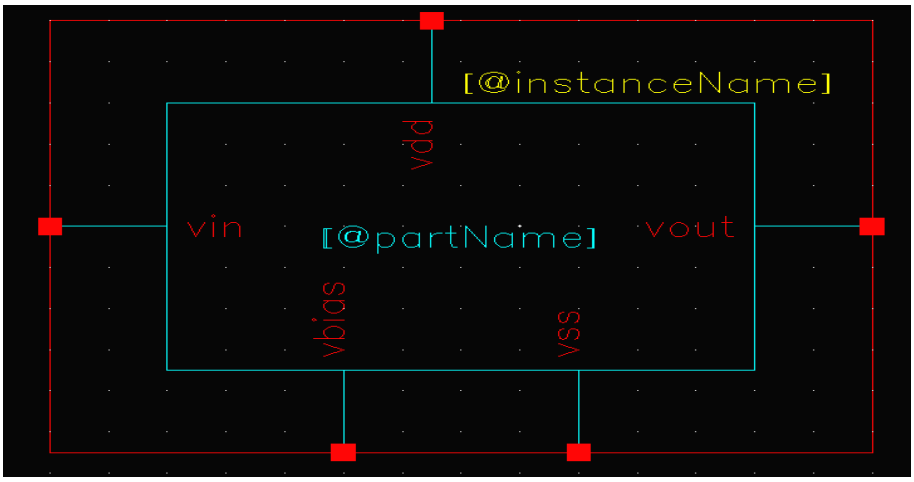
Type the following in the ADD pin form in the exact order leaving space between the pin names.

Pin Names	Direction
vin, vbias	Input
vout	Output
vdd vss	Input

4.7.2 Symbol Creation

Objective: To create a symbol for the Common Drain Amplifier

Use the techniques learned in the Tutorial1 and Tutorial2 to complete the symbol of cd-amplifier

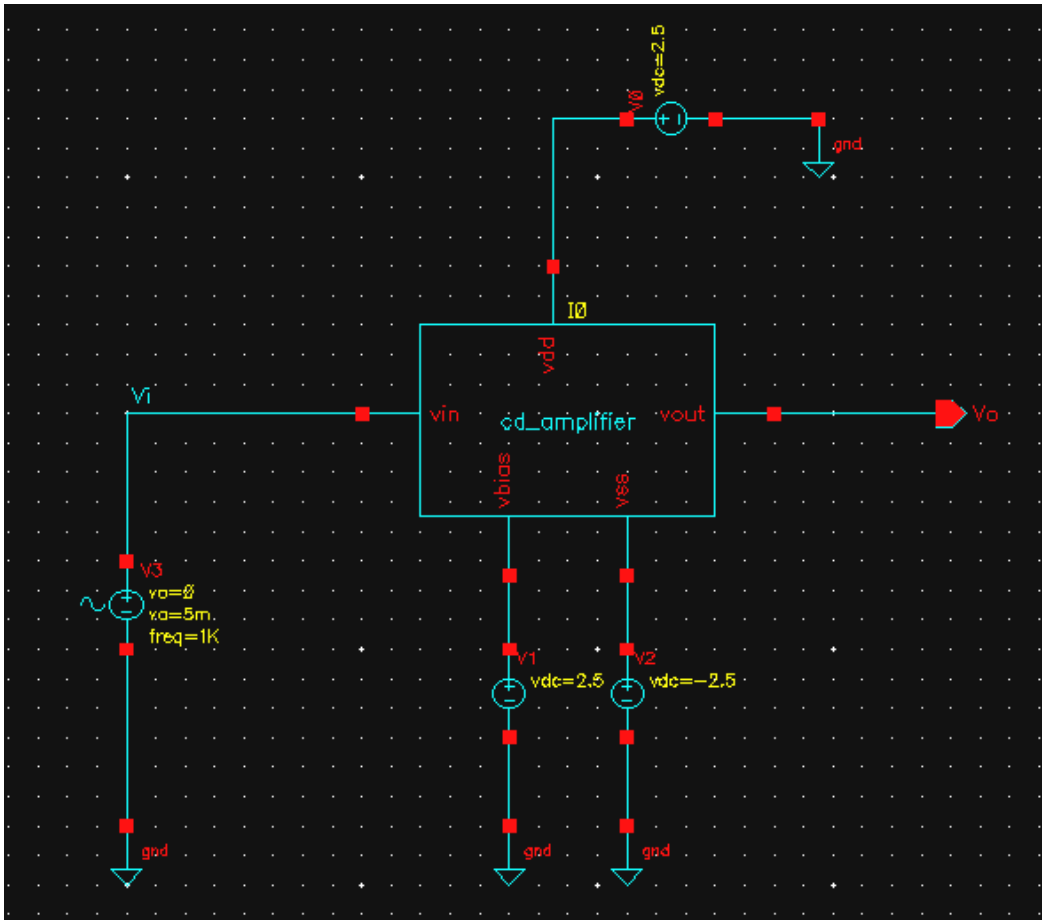


4.7.3 Building the Common Drain Amplifier Test Design

Objective: To build cd_amplifier_test circuit using your cd_amplifier

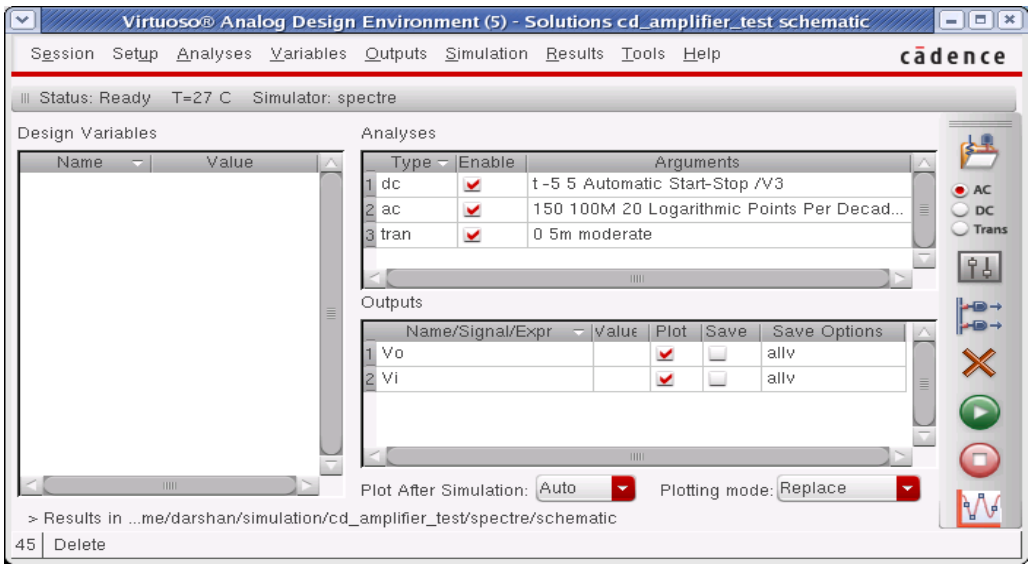
Using the component list and Properties/Comments in the table, build the cd-amplifier_test schematic as shown below.

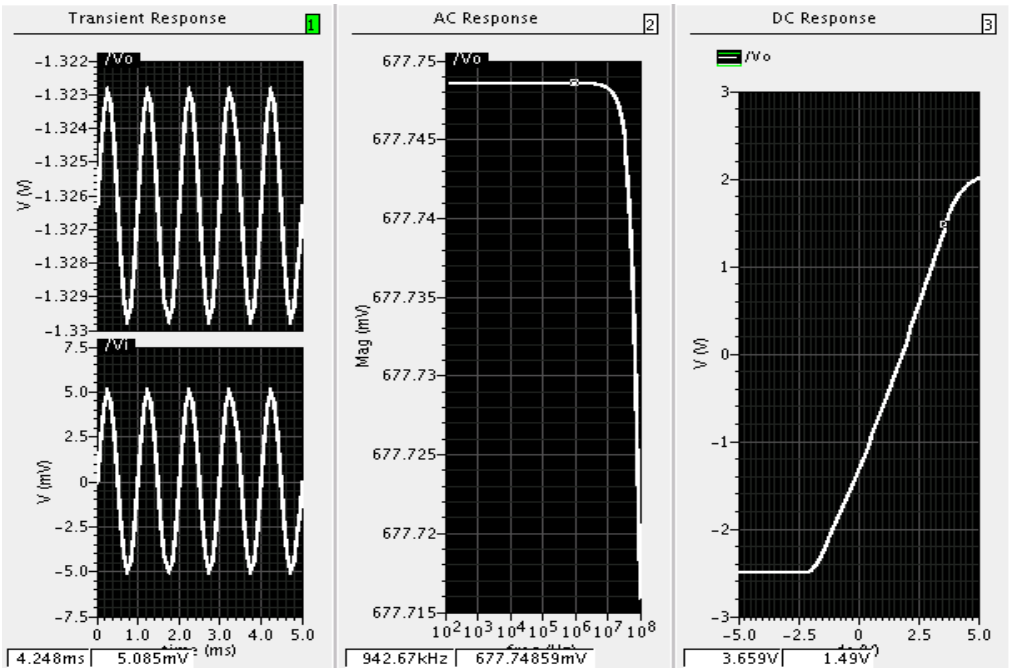
Library name	Cellview name	Properties/Comments
myDesignLib	cd_amplifier	Symbol
analogLib	vsin	Define pulse specification as AC Magnitude= 1; DC Voltage= 0; Offset Voltage= 0; Amplitude= 5m; Frequency= 1K
analogLib	vdd,vss,gnd	vdd=2.5 ; vss= -2.5



4.7.4 Analog Simulation with Spectre

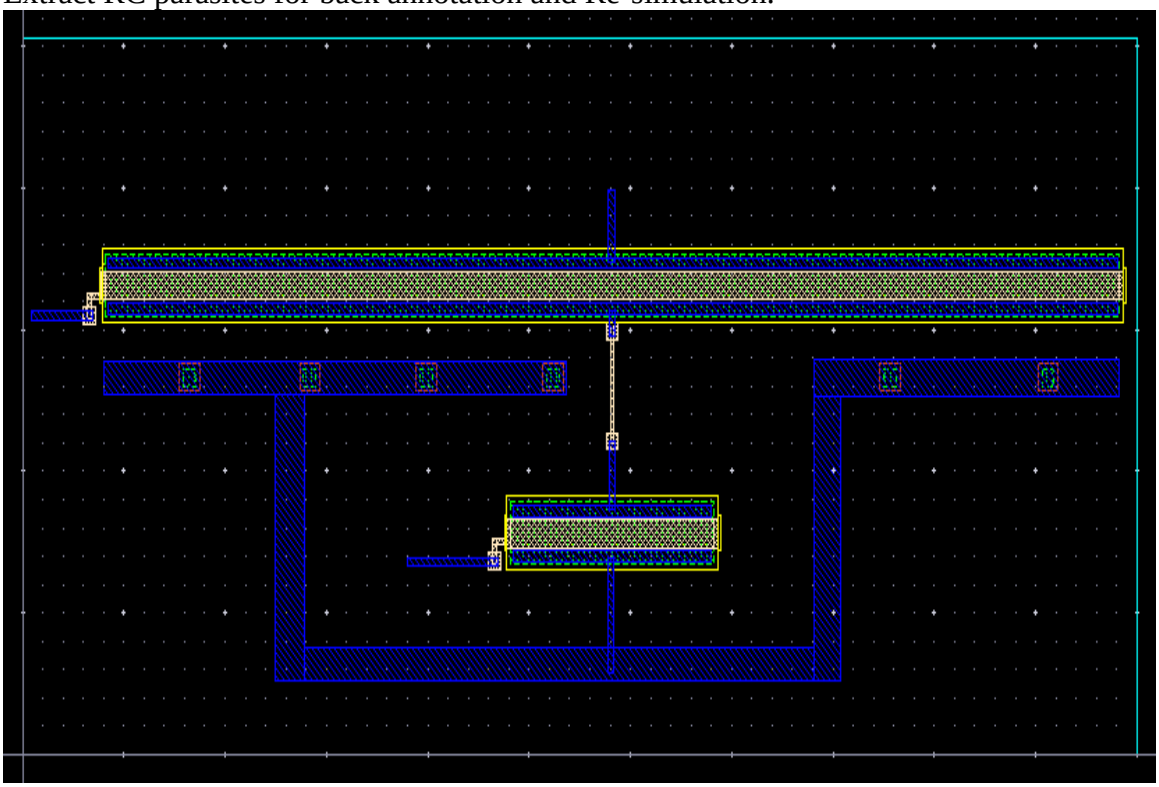
Objective: To set up and run simulations on the cd_amplifier_test design.
Use the techniques learned in the Tutorial1 and Tutorial2 to complete the simulation of cd_amplifier, ADE window and waveform should look like below.





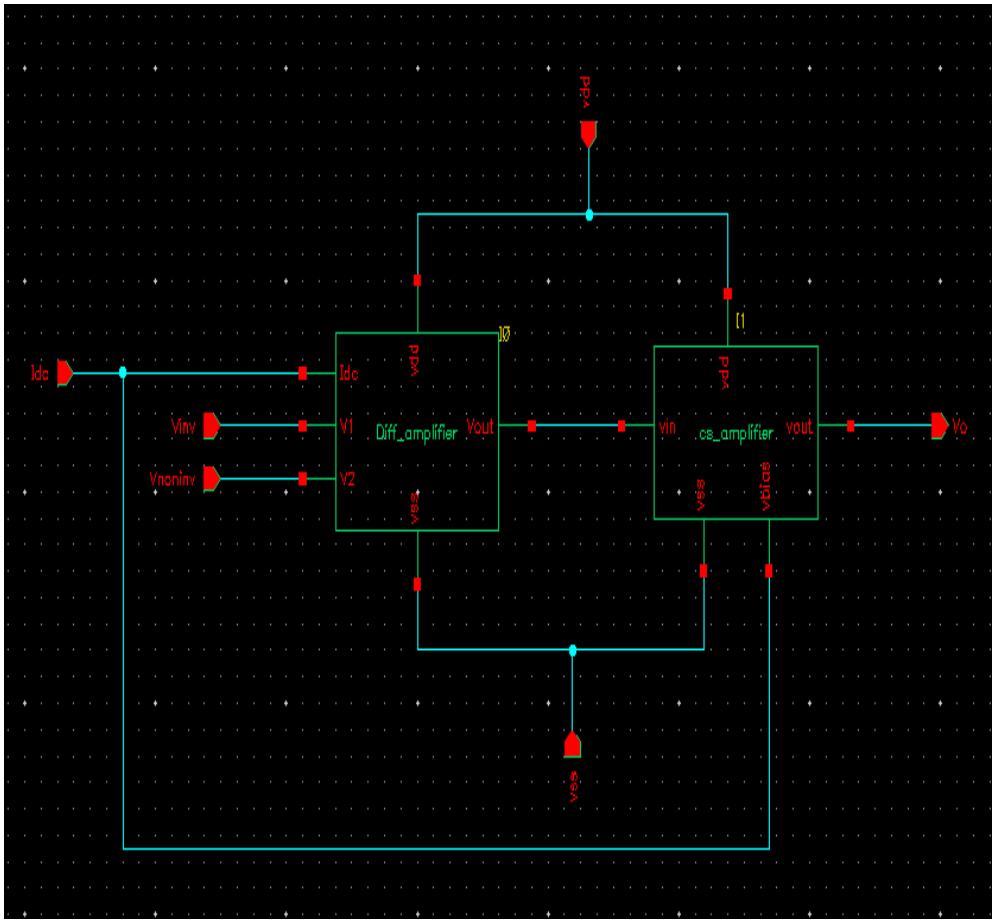
4.7.5 Creating a layout view of Common Drain Amplifier

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.



4.8 Tutorial 5: OPERATIONAL AMPLIFIER

Schematic Capture



4.8.1 Schematic Entry

Objective: To create a new cell view and build Operational Amplifier

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.

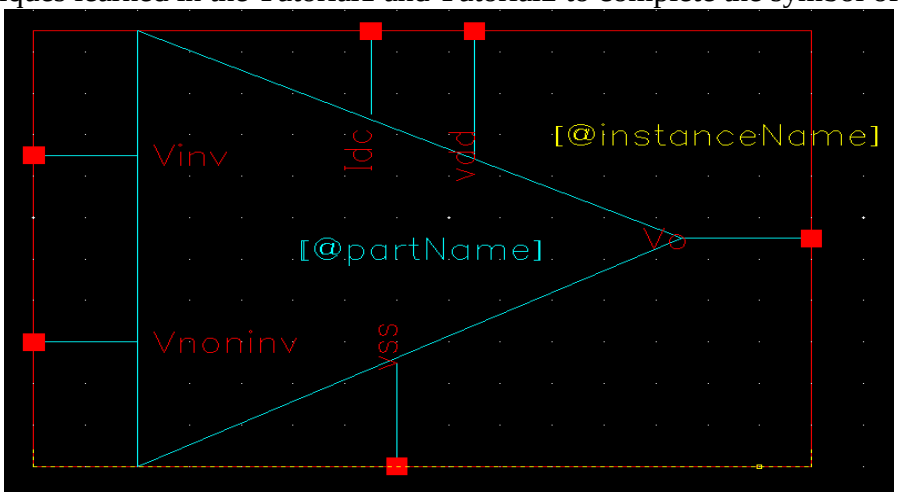
Library name	Cell Name	Properties/Comments
myDesignLib	Diff_amplifier	Symbol
myDesignLib	cs_amplifier	Symbol

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

4.8.2 Symbol Creation

Objective: To create a symbol for the Operational Amplifier
Use the techniques learned in the Tutorial1 and Tutorial2 to complete the symbol of op-amp.

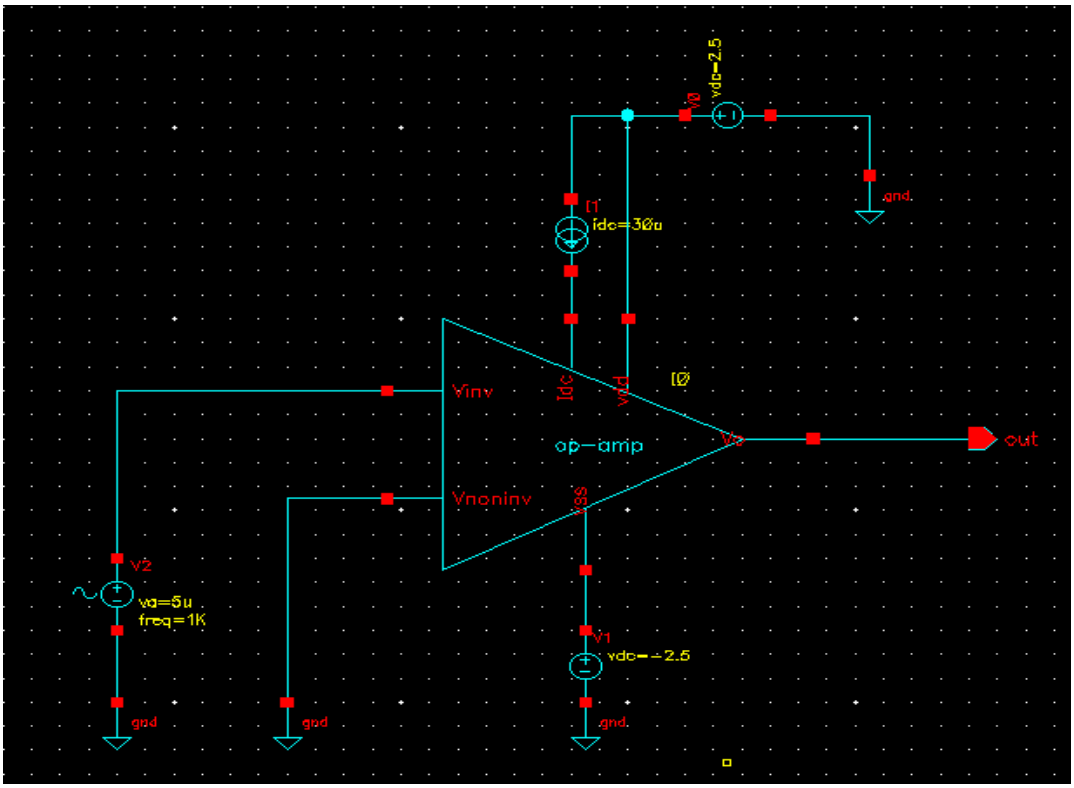


4.8.3 Building the Operational Amplifier Test Design

Objective: To build op-amp_test circuit using your op-amp
Using the component list and Properties/Comments in the table, build the op-amp_test schematic as shown below.

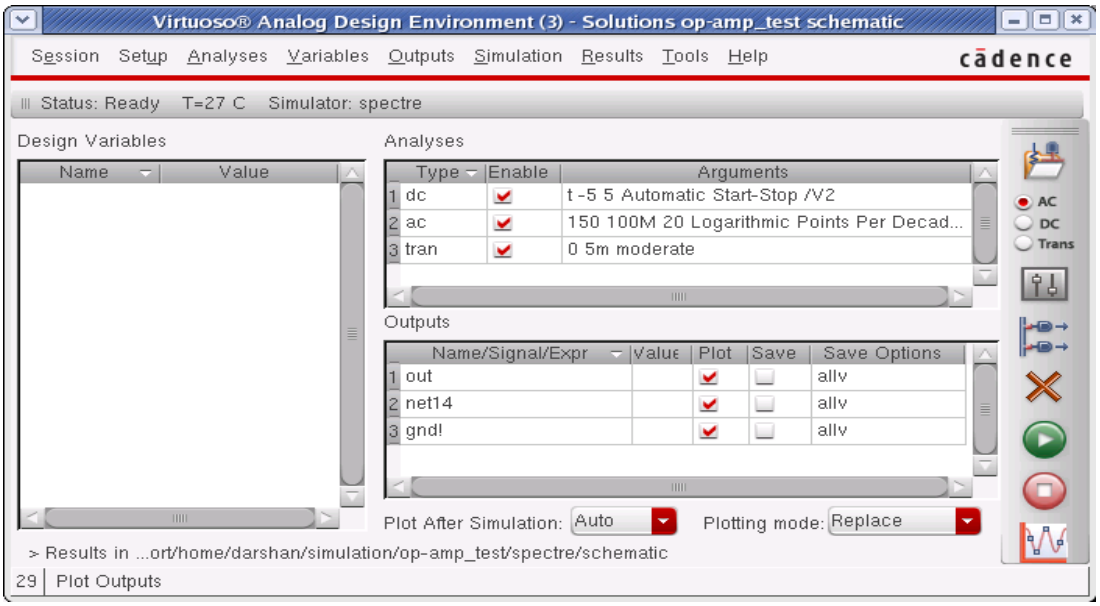
Library name	Cellview name	Properties/Comments
myDesignLib	op-amp	Symbol
analogLib	vsin	Define pulse specification as AC Magnitude= 1; DC Voltage= 0; Offset Voltage= 0; Amplitude= 5m; Frequency= 1K
analogLib	vdc, gnd	vdd=2.5 ; vss= -2.5
analogLib	Idc	Dc current = 30u

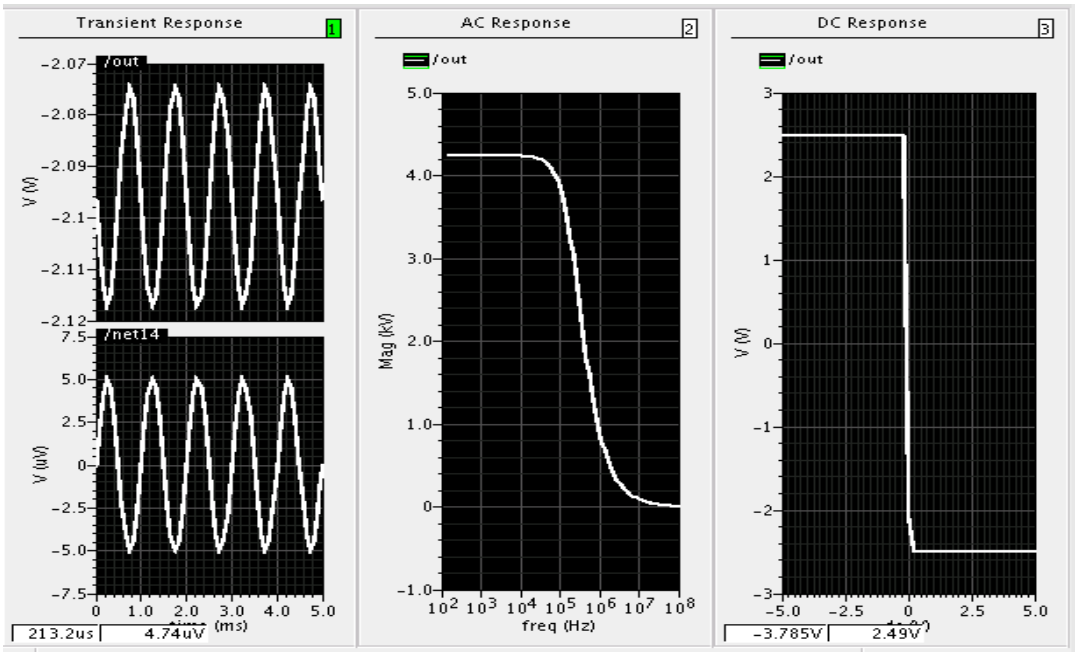
Note: Remember to set the values for **vdd** and **vss**. Otherwise your circuit will have no power.



4.8.4 Analog Simulation with Spectre

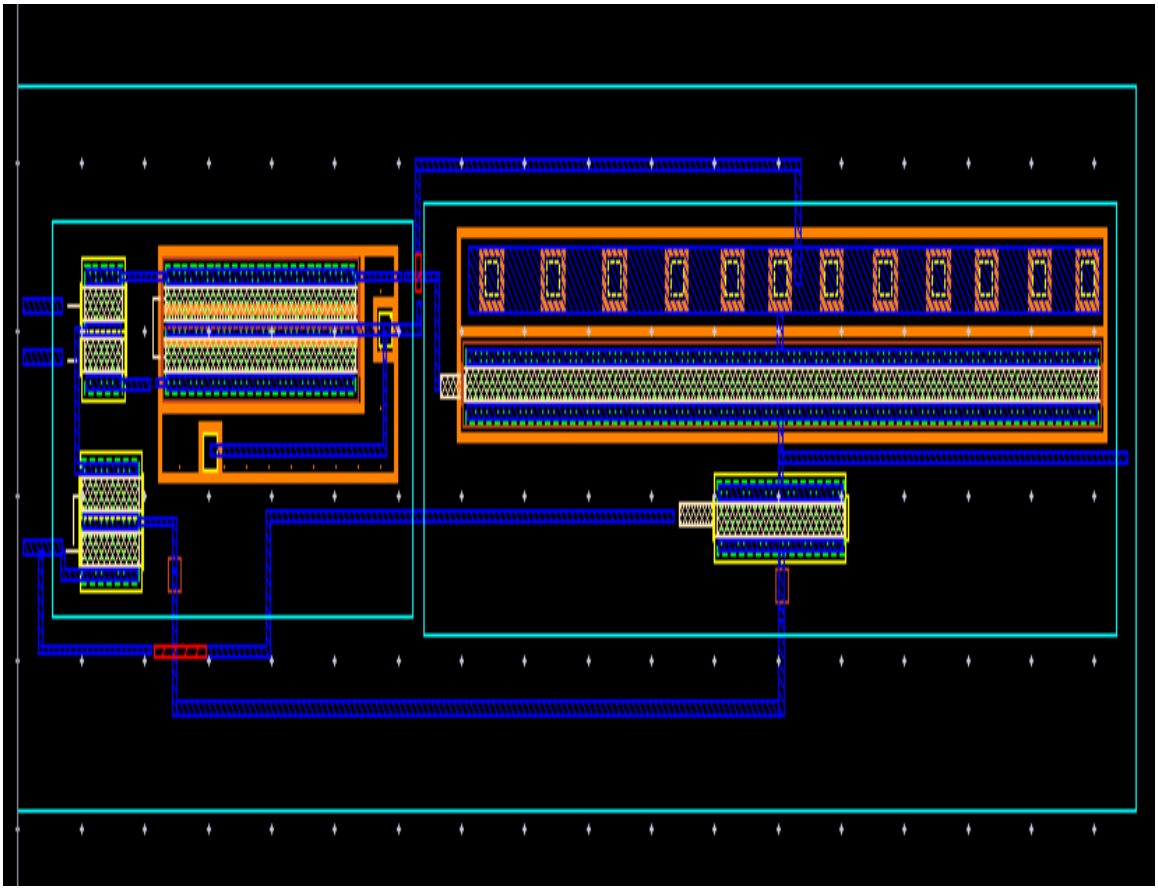
Objective: To set up and run simulations on the op-amp_test design.
Use the techniques learned in the Tutorial1 and Tutorial2 to complete the simulation of op-amp, ADE window and waveform should look like below.



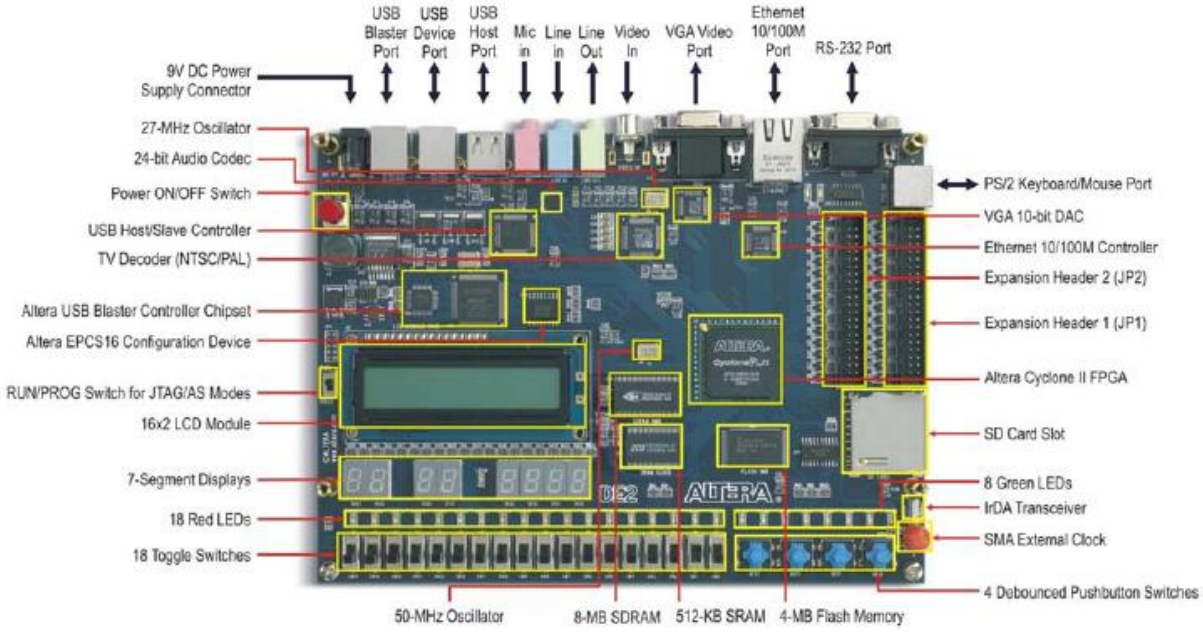


4.8.5 Creating a layout view of Operational Amplifier

Use the techniques learned in the Tutorial1 and Tutorial2 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.



5 Altera DE2 Board (CYCLONE2 FPGA)



The DE2 board has many features that allow the user to implement a wide range of designed circuits, from simple circuits to various multimedia projects.

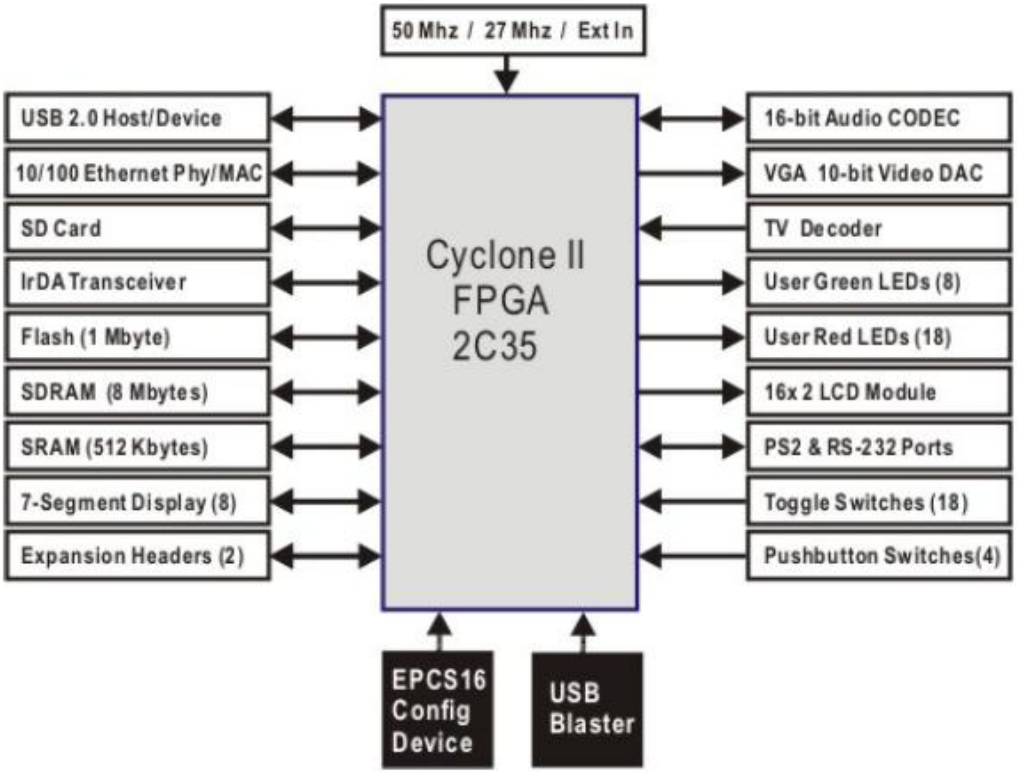
5.1 Components:

The following hardware is provided on the DE2 board:

- Altera Cyclone® II 2C35 FPGA device
- Altera Serial Configuration device - EPCS16
- USB Blaster (on board) for programming and user API control; both JTAG and Active Serial (AS) programming modes are supported
- 512-Kbyte SRAM
- 8-Mbyte SDRAM
- 4-Mbyte Flash memory (1 Mbyte on some boards)
- SD Card socket
- 4 pushbutton switches
- 18 toggle switches
- 18 red user LEDs
- 9 green user LEDs
- 50-MHz oscillator and 27-MHz oscillator for clock sources
- 24-bit CD-quality audio CODEC with line-in, line-out, and microphone-in jacks
- VGA DAC (10-bit high-speed triple DACs) with VGA-out connector
- TV Decoder (NTSC/PAL) and TV-in connector
- 10/100 Ethernet Controller with a connector
- USB Host/Slave Controller with USB type A and type B connectors
- RS-232 transceiver and 9-pin connector
- PS/2 mouse/keyboard connector
- IrDA transceiver
- Two 40-pin Expansion Headers with diode protection

In addition to these hardware features, the DE2 board has software support for standard I/O interfaces and a control panel facility for accessing various components.

5.2 Block Diagram



Cyclone II 2C35 FPGA

- 33,216 LEs
- 105 M4K RAM blocks
- 483,840 total RAM bits
- 35 embedded multipliers
- 4 PLLs
- 475 user I/O pins
- FineLine BGA 672-pin package

Serial Configuration device and USB Blaster circuit

- Altera's EPCS16 Serial Configuration device
- On-board USB Blaster for programming and user API control
- JTAG and AS programming modes are supported

5.3 Power-up the DE2 Board

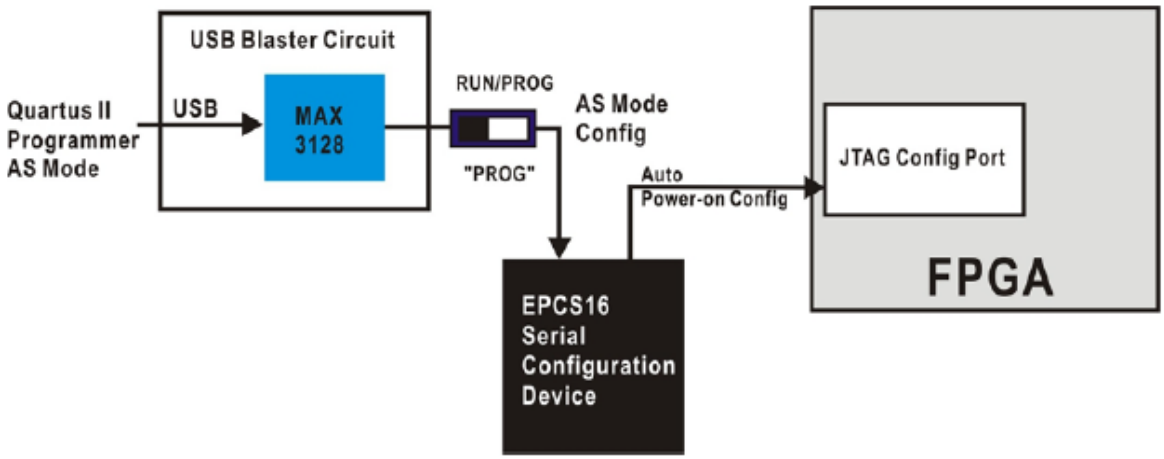
The DE2 board comes with a preloaded configuration bit stream to demonstrate some features of the board. This bit stream also allows users to see quickly if the board is working properly. To power-up the board perform the following steps:

1. Connect the provided USB cable from the host computer to the USB Blaster connector on the DE2 board. For communication between the host and the DE2 board, it is necessary to install the Altera USB Blaster driver software.
2. Connect the 9V adapter to the DE2 board
3. Connect a VGA monitor to the VGA port on the DE2 board
4. Connect your headset to the Line-out audio port on the DE2 board
5. Turn the RUN/PROG switch on the left edge of the DE2 board to RUN position; the PROG position is used only for the AS Mode programming
6. Turn the power on by pressing the ON/OFF switch on the DE2 board

At this point you should observe the following:

- All user LEDs are flashing
- All 7-segment displays are cycling through the numbers 0 to F
- The LCD display shows **Welcome to the Altera DE2 Board**
- The VGA monitor displays the image shown in Figure 2.3.
- Set the toggle switch SW17 to the DOWN position; you should hear a 1-kHz sound
- Set the toggle switch SW17 to the UP position and connect the output of an audio player to
- the Line-in connector on the DE2 board; on your headset you should hear the music played
- from the audio player (MP3, PC, iPod, or the like)
- You can also connect a microphone to the Microphone-in connector on the DE2 board; your
- voice will be mixed with the music played from the audio player

5.4 Configuring the EPCS16 in AS Mode



The above Figure illustrates the AS configuration set up. To download a configuration bit stream into the EPCS16 serial EEPROM device, perform the following steps:

- Ensure that power is applied to the DE2 board
- Connect the supplied USB cable to the USB Blaster port on the DE2 board (see Figure 2.1)
- Configure the JTAG programming circuit by setting the RUN/PROG switch (on the left side
- of the board) to the PROG position.
- The EPCS16 chip can now be programmed by using the Quartus II Programmer module to
- select a configuration bit stream file with the *.pof* filename extension
- Once the programming operation is finished, set the RUN/PROG switch back to the RUN position and then reset the board by turning the power switch off and back on; this action
  causes the new configuration data in the EPCS16 device to be loaded into the FPGA chip.

5.5 Pin Assignments

5.5.1 Pin Assignments for the toggle switches

There are also 18 toggle switches (sliders) on the DE2 board. When a switch is in the DOWN position (closest to the edge of the board) it provides a low logic level (0 volts) to the FPGA, and when the switch is in the UP position it provides a high logic level (3.3 volts).

Signal Name	FPGA Pin No.	Description
SW[0]	PIN_N25	Toggle Switch[0]
SW[1]	PIN_N26	Toggle Switch[1]
SW[2]	PIN_P25	Toggle Switch[2]
SW[3]	PIN_AE14	Toggle Switch[3]
SW[4]	PIN_AF14	Toggle Switch[4]
SW[5]	PIN_AD13	Toggle Switch[5]
SW[6]	PIN_AC13	Toggle Switch[6]
SW[7]	PIN_C13	Toggle Switch[7]
SW[8]	PIN_B13	Toggle Switch[8]
SW[9]	PIN_A13	Toggle Switch[9]
SW[10]	PIN_N1	Toggle Switch[10]
SW[11]	PIN_P1	Toggle Switch[11]
SW[12]	PIN_P2	Toggle Switch[12]
SW[13]	PIN_T7	Toggle Switch[13]
SW[14]	PIN_U3	Toggle Switch[14]
SW[15]	PIN_U4	Toggle Switch[15]
SW[16]	PIN_V1	Toggle Switch[16]
SW[17]	PIN_V2	Toggle Switch[17]

5.5.2 Pin Assignments for the pushbutton switches

The DE2 board provides four pushbutton switches. Each switch provides a high logic level (3.3 volts) when it is not pressed, and provides a low logic level (0 volts) when depressed.

Signal Name	FPGA Pin No.	Description
KEY[0]	PIN_G26	Pushbutton[0]
KEY[1]	PIN_N23	Pushbutton[1]
KEY[2]	PIN_P23	Pushbutton[2]
KEY[3]	PIN_W26	Pushbutton[3]

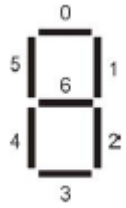
5.5.3 Pin Assignments for Output LED's

There are 27 user-controllable LEDs on the DE2 board. Eighteen red LEDs are situated above the 18 toggle switches, and eight green LEDs are found above the pushbutton switches (the 9th green LED is in the middle of the 7-segment displays). Each LED is driven directly by a pin on the Cyclone II FPGA; driving its associated pin to a high logic level turns the LED on, and driving the pin low turns it off

Signal Name	FPGA Pin No.	Description
LEDR[0]	PIN_AE23	LED Red[0]
LEDR[1]	PIN_AF23	LED Red[1]
LEDR[2]	PIN_AB21	LED Red[2]
LEDR[3]	PIN_AC22	LED Red[3]
LEDR[4]	PIN_AD22	LED Red[4]
LEDR[5]	PIN_AD23	LED Red[5]
LEDR[6]	PIN_AD21	LED Red[6]
LEDR[7]	PIN_AC21	LED Red[7]
LEDR[8]	PIN_AA14	LED Red[8]
LEDR[9]	PIN_Y13	LED Red[9]
LEDR[10]	PIN_AA13	LED Red[10]
LEDR[11]	PIN_AC14	LED Red[11]
LEDR[12]	PIN_AD15	LED Red[12]
LEDR[13]	PIN_AE15	LED Red[13]
LEDR[14]	PIN_AF13	LED Red[14]
LEDR[15]	PIN_AE13	LED Red[15]
LEDR[16]	PIN_AE12	LED Red[16]
LEDR[17]	PIN_AD12	LED Red[17]
LEDG[0]	PIN_AE22	LED Green[0]
LEDG[1]	PIN_AF22	LED Green[1]
LEDG[2]	PIN_W19	LED Green[2]
LEDG[3]	PIN_V18	LED Green[3]
LEDG[4]	PIN_U18	LED Green[4]
LEDG[5]	PIN_U17	LED Green[5]
LEDG[6]	PIN_AA20	LED Green[6]
LEDG[7]	PIN_Y18	LED Green[7]
LEDG[8]	PIN_Y12	LED Green[8]

5.5.4 Using 7-segment Displays

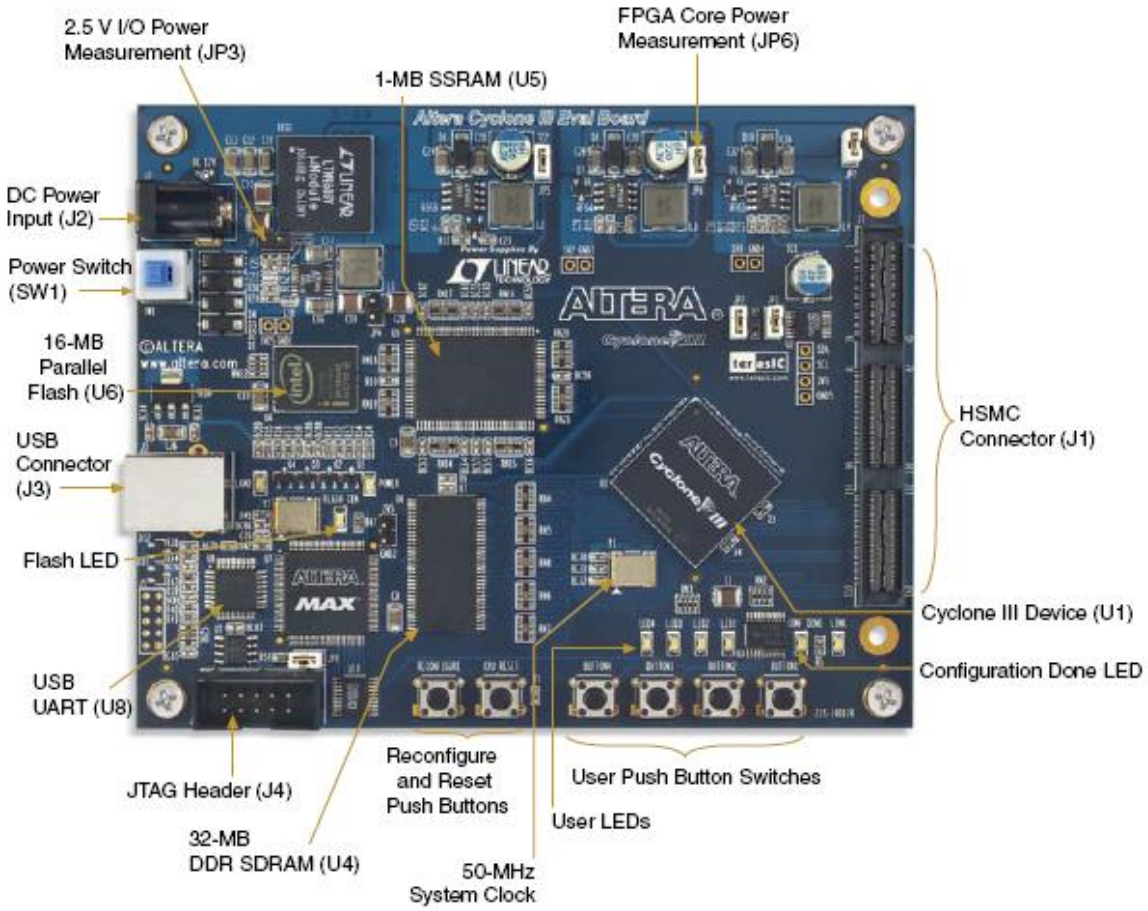
DE2 Board has eight 7-segment displays. These displays are arranged into two pairs and a group of four, with the intent of displaying numbers of various sizes. Applying a low logic level to a segment causes it to light up, and applying a high logic level turns it off.
Each segment in a display is identified by an index from 0 to 6, with the positions given in Figure below. Note that the dot in each display is unconnected and cannot be used.



The assignments of FPGA pins to the 7-segment displays are also provided below

Signal Name	FPGA Pin No.	Description
HEX0[0]	PIN_AF10	Seven Segment Digit 0[0]
HEX0[1]	PIN_AB12	Seven Segment Digit 0[1]
HEX0[2]	PIN_AC12	Seven Segment Digit 0[2]
HEX0[3]	PIN_AD11	Seven Segment Digit 0[3]
HEX0[4]	PIN_AE11	Seven Segment Digit 0[4]
HEX0[5]	PIN_V14	Seven Segment Digit 0[5]
HEX0[6]	PIN_V13	Seven Segment Digit 0[6]
HEX1[0]	PIN_V20	Seven Segment Digit 1[0]
HEX1[1]	PIN_V21	Seven Segment Digit 1[1]
HEX1[2]	PIN_W21	Seven Segment Digit 1[2]
HEX1[3]	PIN_Y22	Seven Segment Digit 1[3]
HEX1[4]	PIN_AA24	Seven Segment Digit 1[4]
HEX1[5]	PIN_AA23	Seven Segment Digit 1[5]
HEX1[6]	PIN_AB24	Seven Segment Digit 1[6]
HEX2[0]	PIN_AB23	Seven Segment Digit 2[0]
HEX2[1]	PIN_V22	Seven Segment Digit 2[1]
HEX2[2]	PIN_AC25	Seven Segment Digit 2[2]
HEX2[3]	PIN_AC26	Seven Segment Digit 2[3]
HEX2[4]	PIN_AB26	Seven Segment Digit 2[4]
HEX2[5]	PIN_AB25	Seven Segment Digit 2[5]
HEX2[6]	PIN_Y24	Seven Segment Digit 2[6]
HEX3[0]	PIN_Y23	Seven Segment Digit 3[0]
HEX3[1]	PIN_AA25	Seven Segment Digit 3[1]
HEX3[2]	PIN_AA26	Seven Segment Digit 3[2]
HEX3[3]	PIN_Y26	Seven Segment Digit 3[3]
HEX3[4]	PIN_Y25	Seven Segment Digit 3[4]
HEX3[5]	PIN_U22	Seven Segment Digit 3[5]
HEX3[6]	PIN_W24	Seven Segment Digit 3[6]
HEX4[0]	PIN_U9	Seven Segment Digit 4[0]
HEX4[1]	PIN_U1	Seven Segment Digit 4[1]
HEX4[2]	PIN_U2	Seven Segment Digit 4[2]
HEX4[3]	PIN_T4	Seven Segment Digit 4[3]
HEX4[4]	PIN_R7	Seven Segment Digit 4[4]
HEX4[5]	PIN_R6	Seven Segment Digit 4[5]
HEX4[6]	PIN_T3	Seven Segment Digit 4[6]

6 CYCLONE III FPGA



The main features of the Cyclone III starter board are:

- Low-power consumption Altera Cyclone III EP3C25 chip in a 324-pin FineLine BGA (FBGA) package
- Expandable through HSMC connector
- 32-megabyte (MB) DDR SDRAM
- 16-MB parallel flash device for configuration and storage
- 1-MB high-speed SSRAM memory
- Four user push-button switches
- Four user LEDs

The main advantages of the Cyclone III starter board are:

- Facilitates a fast and successful FPGA design experience with helpful example designs and demonstrations.
- Directly configure and communicate with the Cyclone III device via the on-board USB-Blaster™ circuitry and JTAG header
- Active parallel flash configuration
- Low power consumption
- Cost-effective modular design

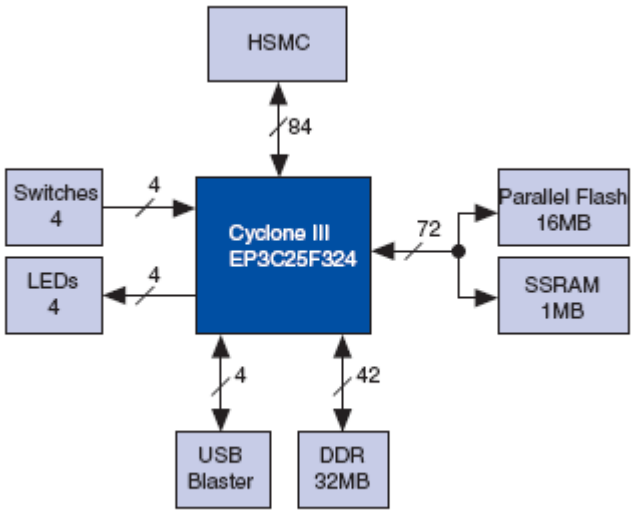
6.1 COMPONENTS

The Board Component blocks are

- Altera Cyclone III EP3C25F324 FPGA
  - 25K logic elements (LEs)
  - 66 M9K memory blocks (0.6 Mb)
  - 16 18x18 multiplier blocks
  - Four PLLs
  - 214 I/Os
- Clock management system
  - One 50-MHz clock oscillator to support a variety of protocols
  - The Cyclone III device distributes the following clocks from its on-board PLLs:
    1. DDR clock
    2. SSRAM clock
    3. Flash clock
- HSMC connector
  - Provides 12 V and 3.3 V interface for installed daughtercards
  - Provides up to 84 I/O pins for communicating with HSMC daughtercards
- General user-interface
  - Four user LEDs
  - Two board-specific LEDs
  - Push-buttons:
    - System reset
    - User reset
    - Four general user push-buttons
- Memory subsystem
  - Synchronous SRAM device
    - 1-MB standard synchronous SRAM
    - 167-MHz
    - Shares bus with parallel flash device
  - Parallel flash device
    - 16-MB device for active parallel configuration and storage
    - Shares bus with SRAM device
  - DDR SDRAM device
    - 56-pin, 32-MB DDR SDRAM
    - 167-MHz
    - Connected to FPGA via dedicated 16-bit bus
- Built-in USB-Blaster interface
  - Using the Altera EPM3128A CPLD
  - For external configuration of Cyclone III device
  - For system debugging using the SignalTap® and Nios® debugging console

- Communications port for Board Diagnostic graphical user interface (GUI)

6.2 BLOCK DIAGRAM



6.3 PIN ASSIGNMENTS

6.3.1 Pin Settings for pushbutton

The board has system reset, user reset, and 4 user push-buttons. The push-buttons are in logic "1" until depressed. The lists the pinout for all push-buttons

Signal Name	FPGA Pin	Direction	Type
KEY0	F1	Input	2.5 V
KEY1	F2	Input	2.5 V
KEY2	A10	Input	2.5 V
KEY3	B10	Input	2.5 V
CPU_RESET_N	N2	Input	2.5 V
RECONFIGURE	H5 (nConfig)	Input	2.5 V

6.3.2 Pin settings for LEDs

There are 4 user LEDs . A logic '0' illuminates the LEDs

Signal Name	FPGA Pin Name	Direction	Type
LED0	P13	Output	2.5 V
LED1	P12	Output	2.5 V
LED2	N12	Output	2.5 V
LED3	N9	Output	2.5 V
Power LED	—	—	—
MAX Load LED	—	—	—

7 LAB EXPERIMENTS: Verilog Code

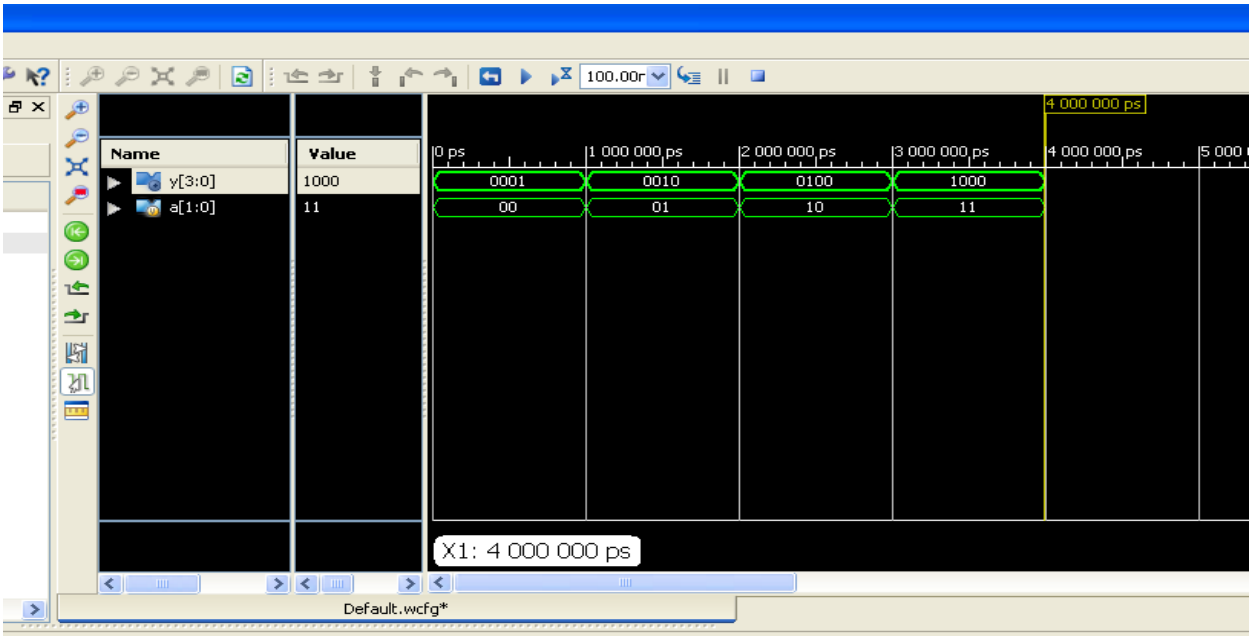
7.1 LOGIC GATES

//Verilog code for gates

```
`timescale 1ns/1ps
module logicgates(
        input A,
        input B,
        output notA,
        output andAB,
        output orAB,
        output nandAB,
        output norAB,
        output xorAB,
        output xnorAB
        );
        /* Seven different logic gates acting on four bit busses */
        assign notA= ~A; // NOT gate
        assign andAB= A & B; // AND gate
        assign orAB = A | B; //  OR gate
        assign nandAB = ~(A & B); // NAND gate
        assign norAB = ~(A | B); // NOR gate
        assign xorAB = A ^ B; // XOR gate
        assign xnorAB= ~(A ^ B); // XNOR gate
        endmodule
```

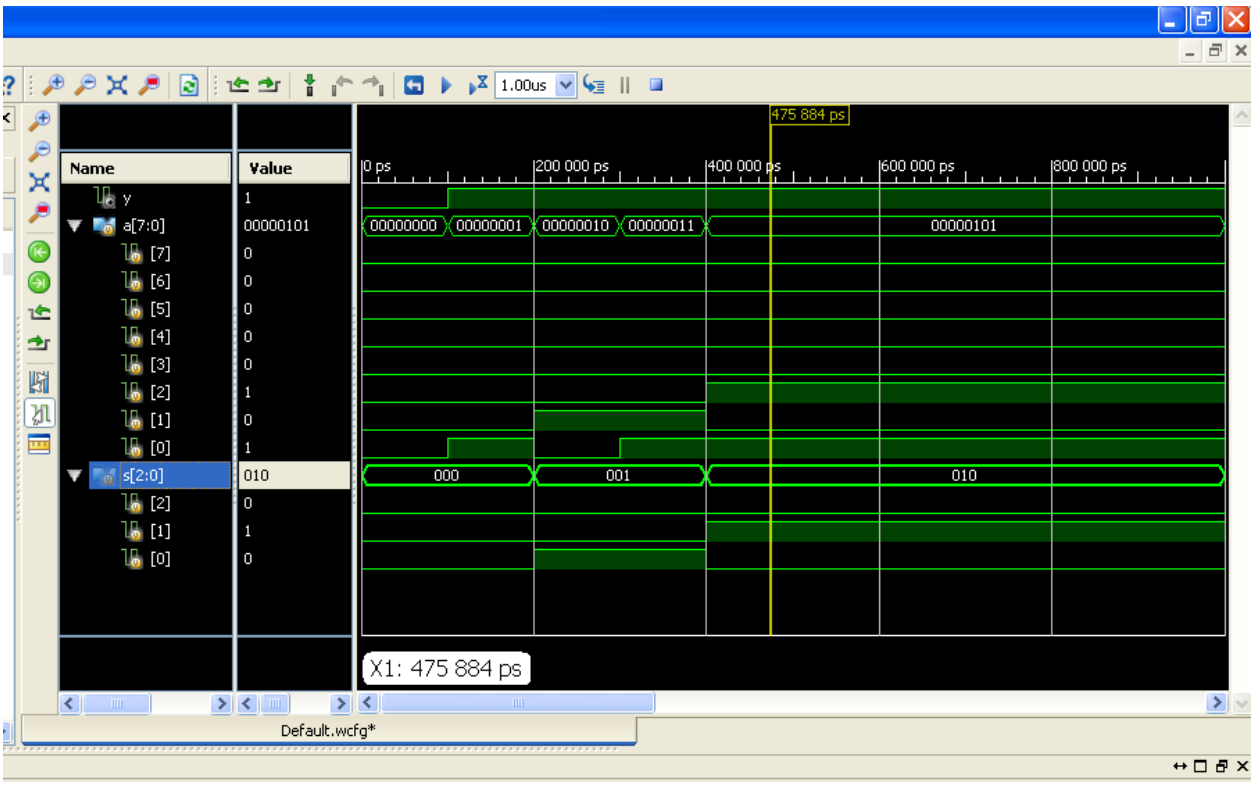
7.2 2-to-4 Decoder

```
`timescale 1ns/1ps
module decoder(
        input [1:0] a,
        output reg [3:0] y
        );
        always@(a)
        case(a)
        2'b00: y= 4'b0001;
        2'b01: y= 4'b0010;
        2'b10: y= 4'b0100;
        2'b11: y= 4'b1000;
        endcase
        endmodule
```



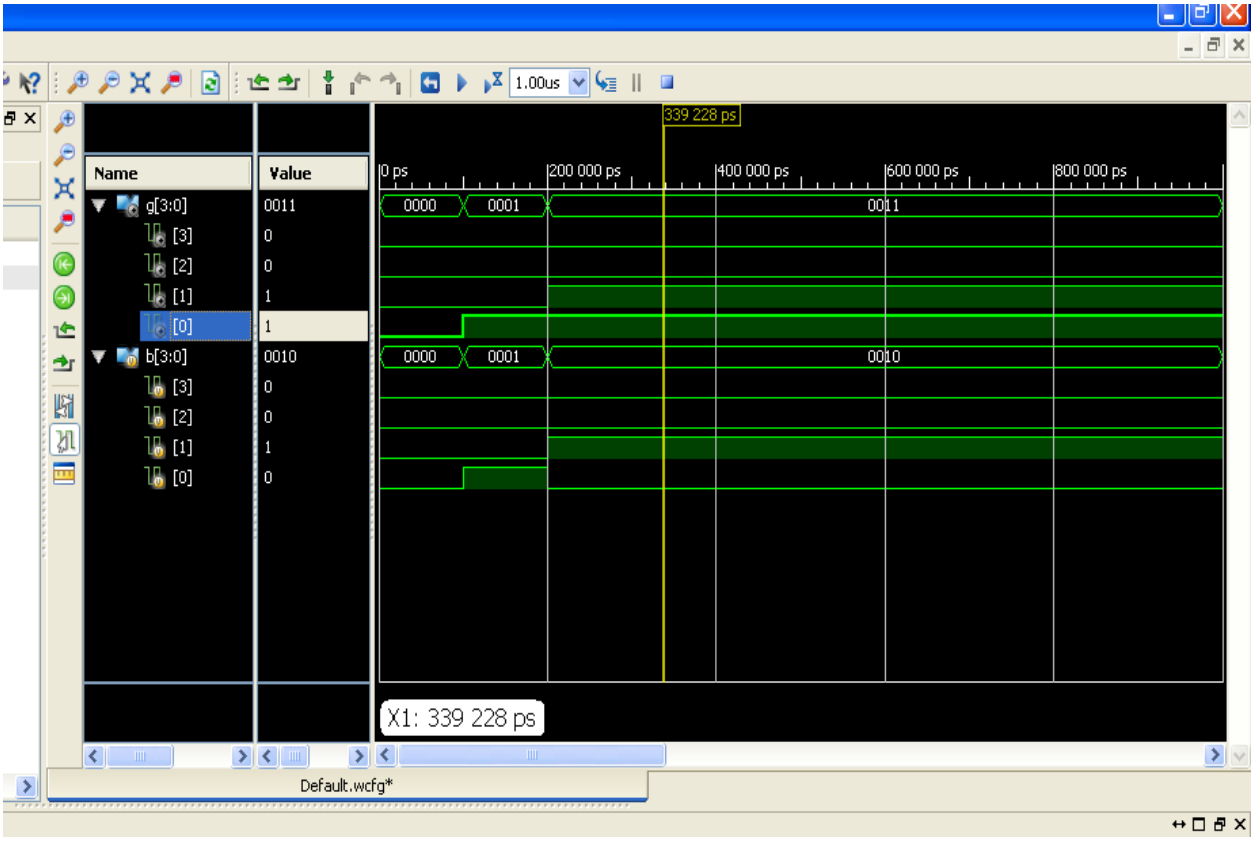
7.3 8-to-1 Multiplexer

```
`timescale 1ns/1ps
module mux8to1(
        input [7:0] a,
        input [2:0] s,
        output reg y
        );
        always@(a,s)
        case(s)
        3'b000: y=a[0];
        3'b001: y=a[1];
        3'b010: y=a[2];
        3'b011: y=a[3];
        3'b100: y=a[4];
        3'b101: y=a[5];
        3'b110: y=a[6];
        3'b111: y=a[7];
        endcase
        endmodule
```



7.4 4Bit Binary to Gray converter

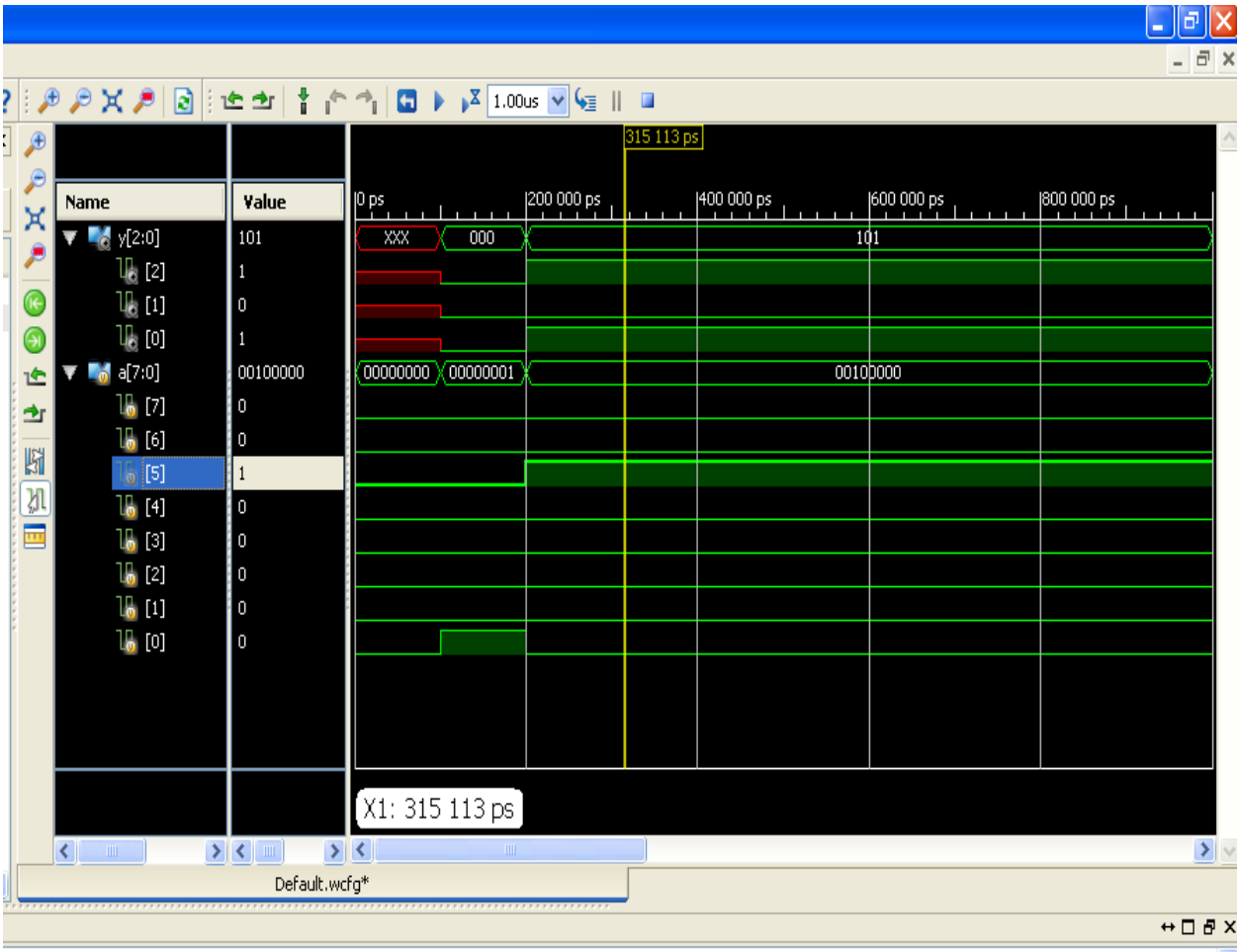
```
`timescale 1ns/1ps
module b2g_4b(
        input [3:0] b,
        output [3:0] g
);
        assign g[3]=b[3];
        assign g[2]=b[3]^b[2];
        assign g[1]=b[2]^b[1];
        assign g[0]=b[1]^b[0];
        endmodule
```



7.5 8-to-3 encoder

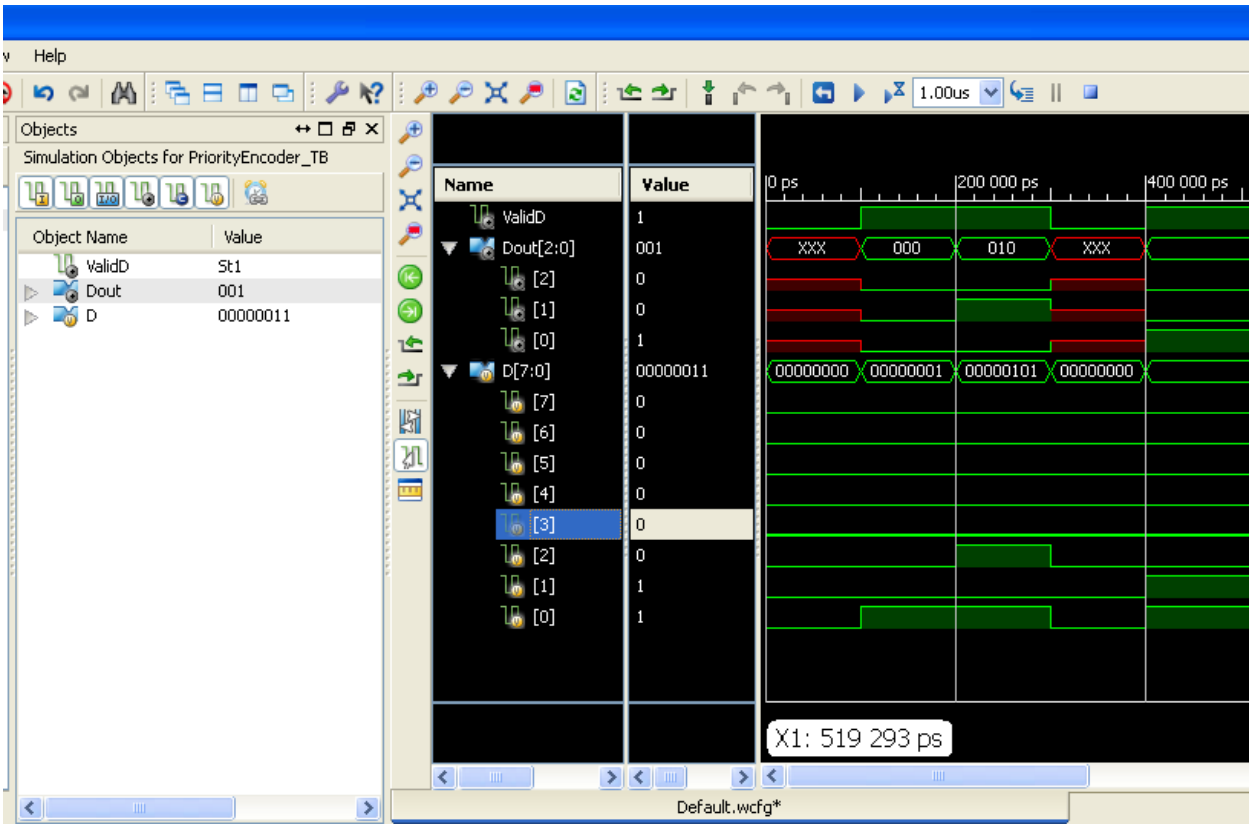
7.5.1 Encoder without parity

```
`timescale 1ns/1ps
module encoder8x3(
   input [7:0] a,
   output reg [2:0] y
   );
always @(a)
begin
if     (a==8'b00000001) y=3'b000;
else if (a==8'b00000010) y=3'b001;
else if (a==8'b00000100) y=3'b010;
else if (a==8'b00001000) y=3'b011;
else if (a==8'b00010000) y=3'b100;
else if (a==8'b00100000) y=3'b101;
else if (a==8'b01000000) y=3'b110;
else if (a==8'b10000000) y=3'b111;
else y=3'bX;
end
endmodule
```



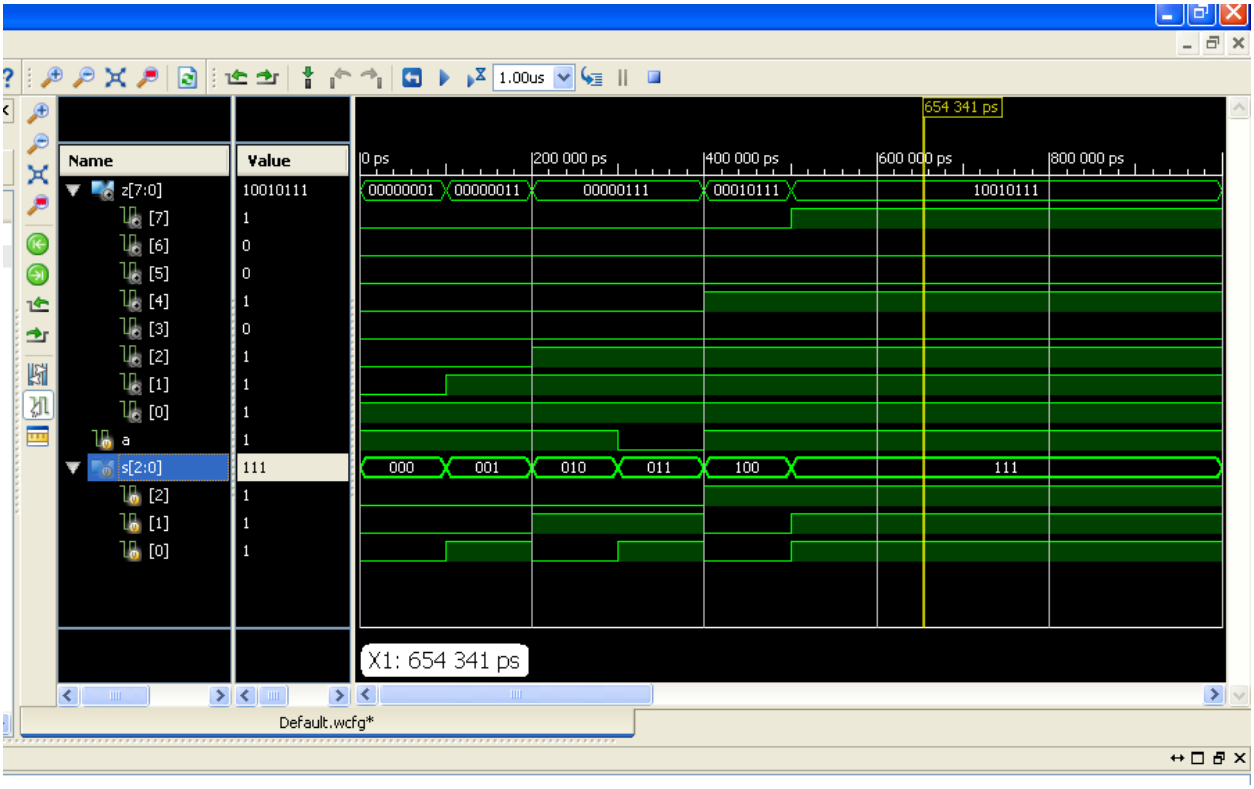
7.5.2 Priority encoder

```
`timescale 1ns/1ps
module PriorityEncoder(
   input [7:0] D,
   output ValidD,
   output reg [2:0] Dout
   );

assign ValidD= |D; // Use of "reduction or" operator
always @ (D)
begin
casex (D)
8'b1xxxxxxx : Dout=7;
8'b01xxxxxx : Dout=6;
8'b001xxxxx : Dout=5;
8'b0001xxxx : Dout=4;
8'b00001xxx : Dout=3;
8'b000001xx : Dout=2;
8'b0000001x : Dout=1;
8'b00000001 : Dout=0;
default     : Dout=3'bx;

endcase
end
endmodule
```


Help
1.00us
Objects
Simulation Objects for PriorityEncoder_TB
Object Name
Value
ValidD
St1
Dout
001
D
00000011
Name
Value
ValidD
1
Dout[2:0]
001
[2]
0
[1]
0
[0]
1
D[7:0]
00000011
[7]
0
[6]
0
[5]
0
[4]
0
[3]
0
[2]
0
[1]
1
[0]
1
0 ps
200 000 ps
400 000 ps
XXX
000
010
XXX
00000000
00000001
00000101
00000000
X1: 519 293 ps
Default.wcfg*

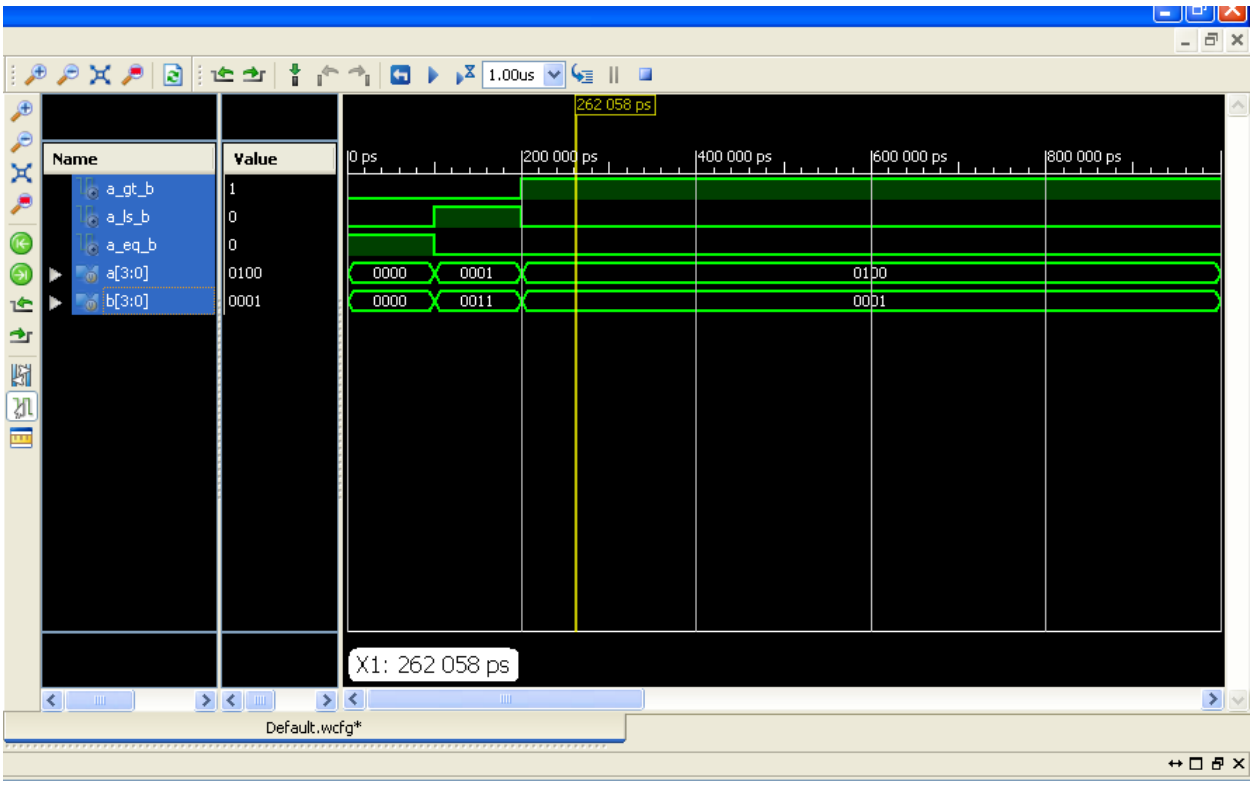
7.6 Demux 1-to-8

```
`timescale 1ns/1ps
module Demux8to1(
    input a,
    input [2:0] s,
    output reg [7:0] z =0    );
always@(a,s)
if (s==0) z[0] = a;
else if (s==1) z[1] = a;
else if (s==2) z[2] = a;
else if (s==3) z[3] = a;
else if (s==4) z[4] = a;
else if (s==5) z[5] = a;
else if (s==6) z[6] = a;
else if (s==7) z[7] = a;
endmodule
```



7.7 Comparator

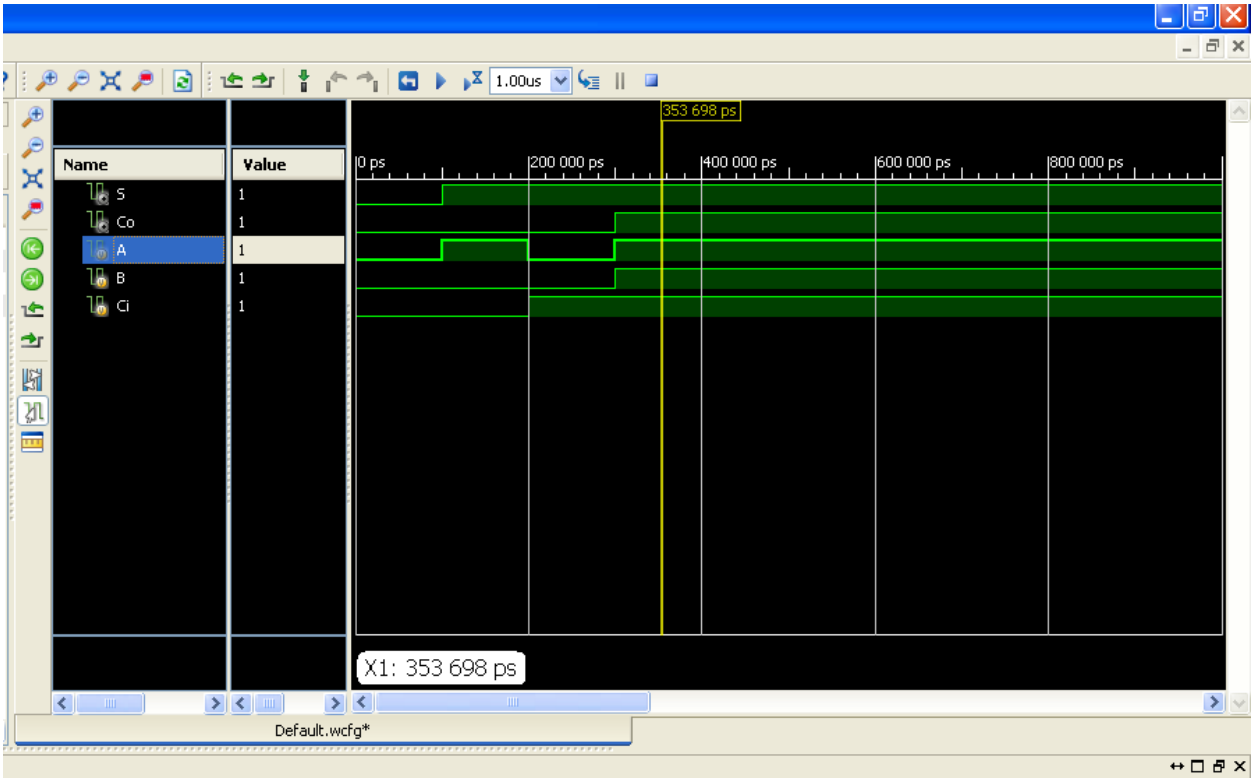
```
`timescale 1ns/1ps
module Comparator(
    input [3:0] a,
    input [3:0] b,
    output a_gt_b,
    output a_ls_b,
    output a_eq_b
    );
assign a_gt_b = (a > b);
assign a_ls_b = (a < b);
assign a_eq_b = (a == b);
endmodule
```



7.8 Full adder

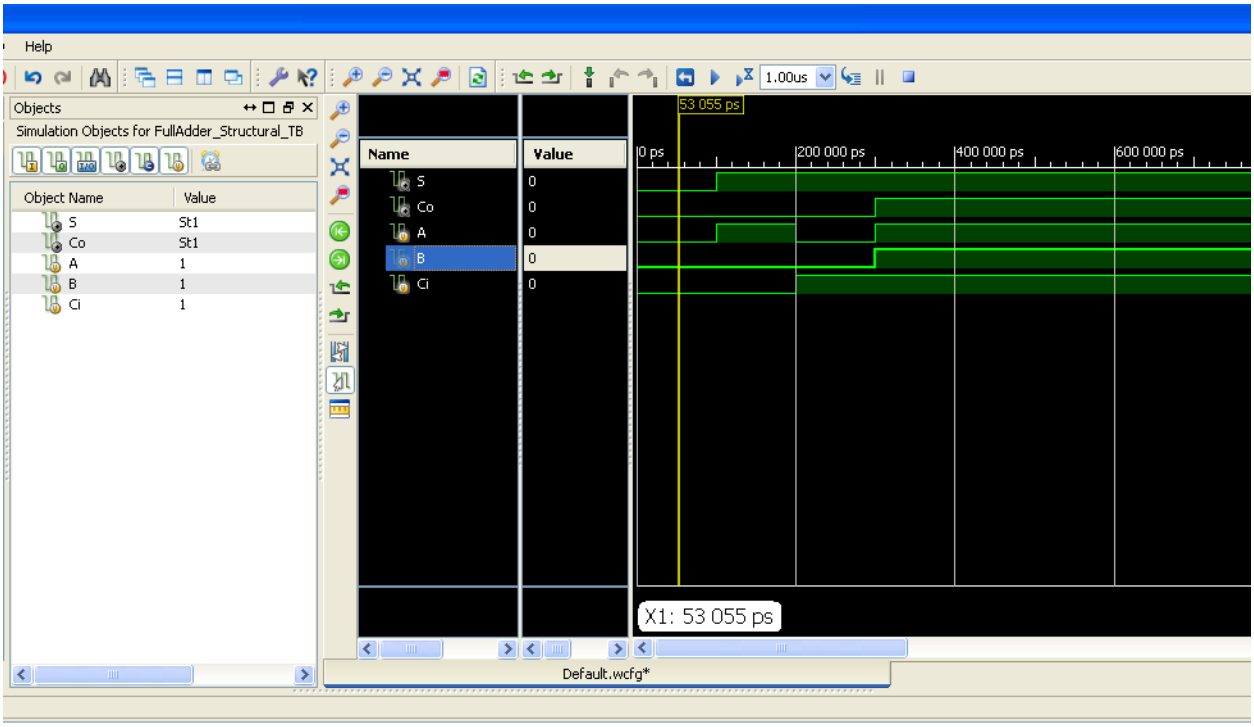
7.8.1 Full adder: Continuous Assignment

```
`timescale 1ns/1ps
module FullAdder_CtAssign(
    input A,
    input B,
    input Ci,
    output S,
    output Co
    );
assign S =A^B^Ci;
assign Co= (A&B) | (Ci&A) | (Ci&B);
endmodule
```



7.8.2 Full adder: Structural

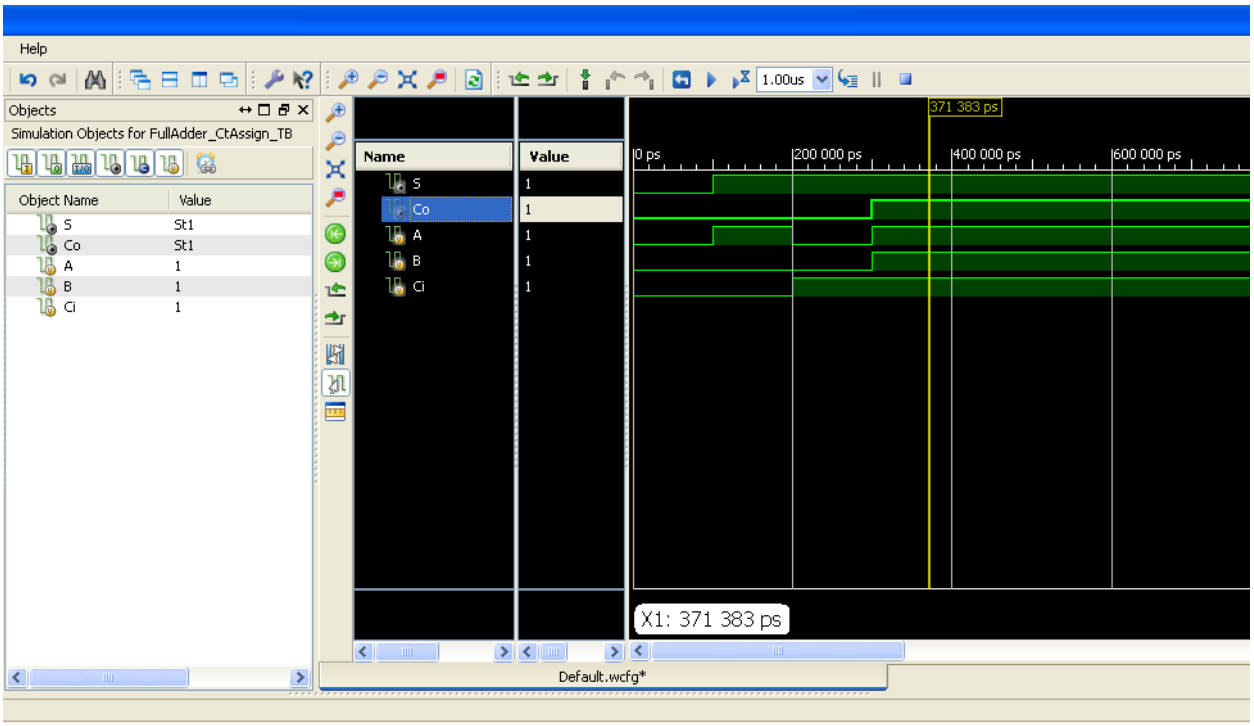
```
`timescale 1ns/1ps
module FullAdder_Structural(
    input A,
    input B,
    input Ci,
    output S,
    output Co
    );
wire z1,z2,z3,z4;
and And1(z1,A,B);
and And2(z2,A,Ci);
and And3(z3,B,Ci);
or Or1(Co, z1, z2, z3);
xor Xor1(z4, A, B);
xor Xor2(S,z4,Ci);
endmodule
```



ion Manager to check out a full ISim license.

7.8.3 Full adder: Procedural

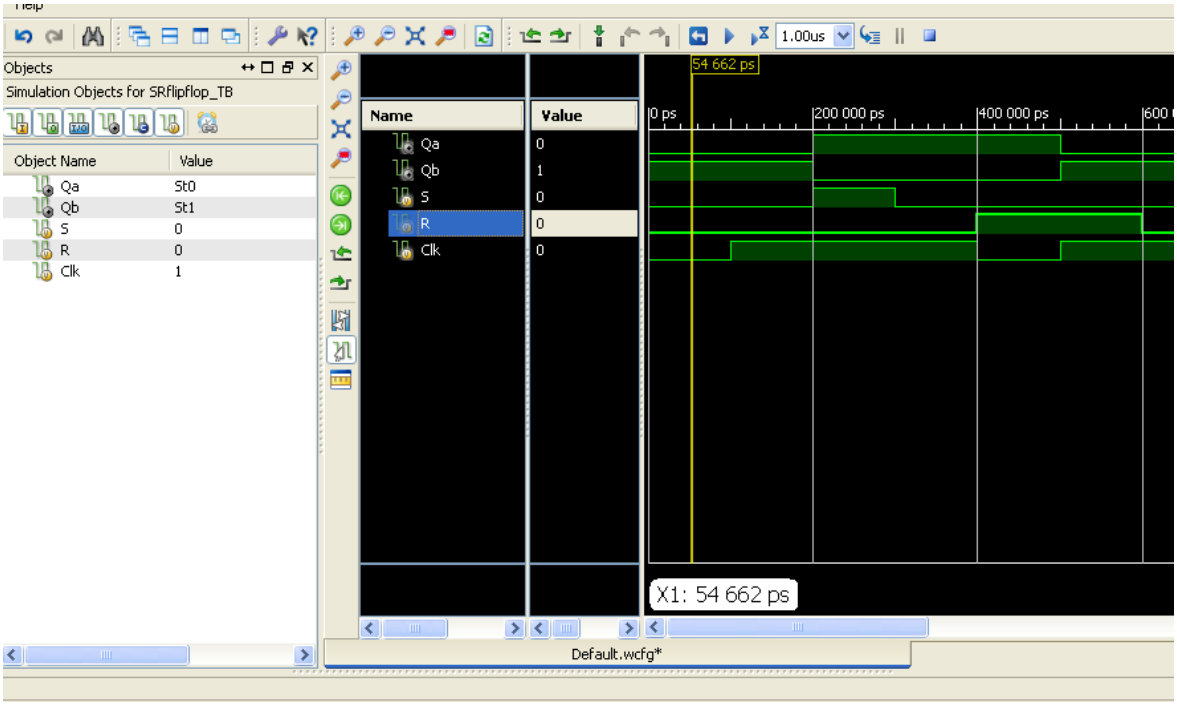
```
`timescale 1ns/1ps
module FullAdder_Procedural(
    input A,
    input B,
    input Ci,
    output reg S,
    output reg Co
    );

always @(Ci, A, B)
begin
case({Ci, A, B})
3'b000: {Co,S}= 'b00;
3'b001: {Co,S}= 'b01;
3'b010: {Co,S}= 'b01;
3'b011: {Co,S}= 'b10;
3'b100: {Co,S}= 'b00;
3'b101: {Co,S}= 'b10;
3'b110: {Co,S}= 'b10;
3'b111: {Co,S}= 'b11;
endcase
end
endmodule
```



7.9 Flip-flops

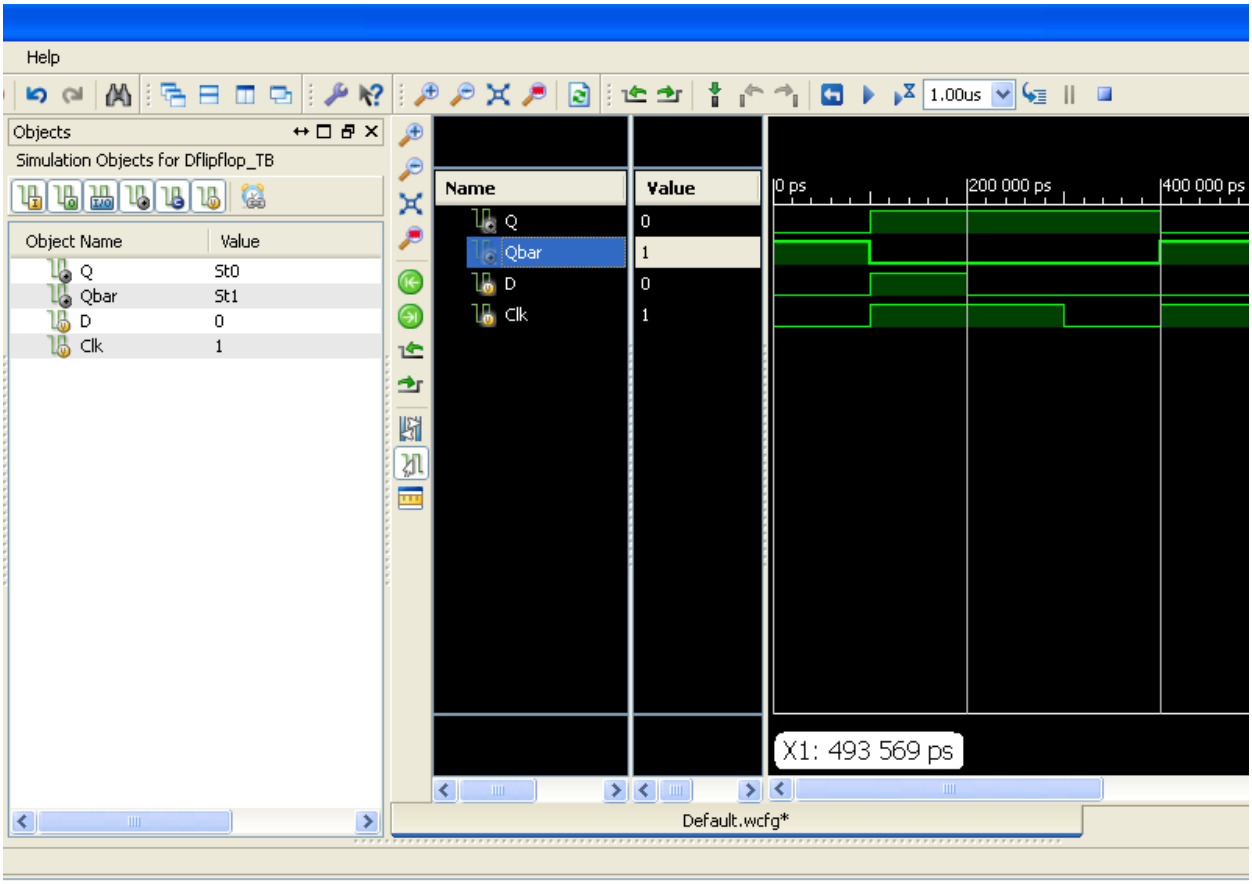
7.9.1 SR FLIPFLOP

```
`timescale 1ns/1ps
module SRflipflop(
    input S,
    input R,
    input Clk,
    output reg Qa =1'b0,
    output reg Qb =1'b1
    );

//wire S,R,Clk;
reg a,b;

always@(S,R,Clk)
begin
a=Qa;
b=Qb;
if (Clk==1'b1)
case({S,R})
'b00: {Qa,Qb} ={a,b};
'b01: {Qa,Qb} = 'b01;
'b10: {Qa,Qb} = 'b10;
endcase
else if(Clk==1'b0)
{Qa,Qb} ={a,b};
end
endmodule
```



7.9.2 D FLIPFLOP

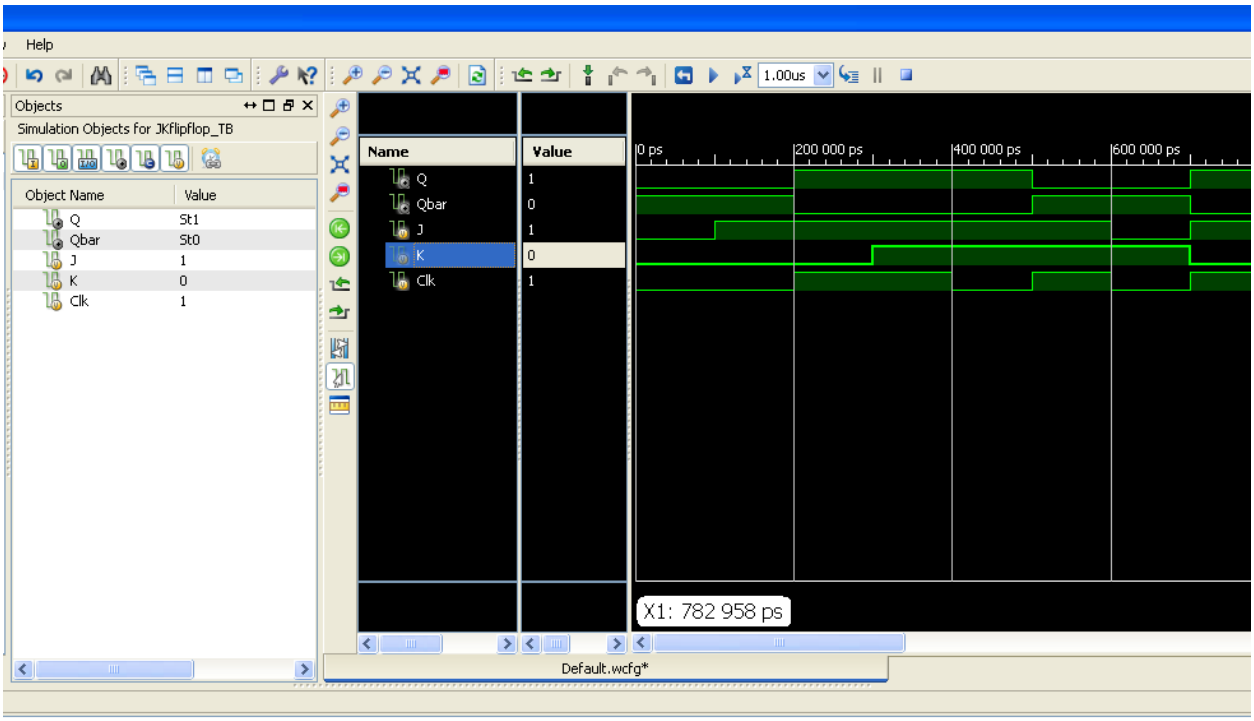
```
`timescale 1ns/1ps
module Dflipflop(
    input wire D,
    input wire Clk,
    output reg Q=0,
    output reg Qbar=1
    );
always@(posedge Clk)
begin
Q=D;
Qbar=~Q;
end
endmodule
```



on Manager to check out a full ISim license.
er to the ISim documentation for more information on the differences between the Lite and the Full version.

7.9.3 JK FlipFlop

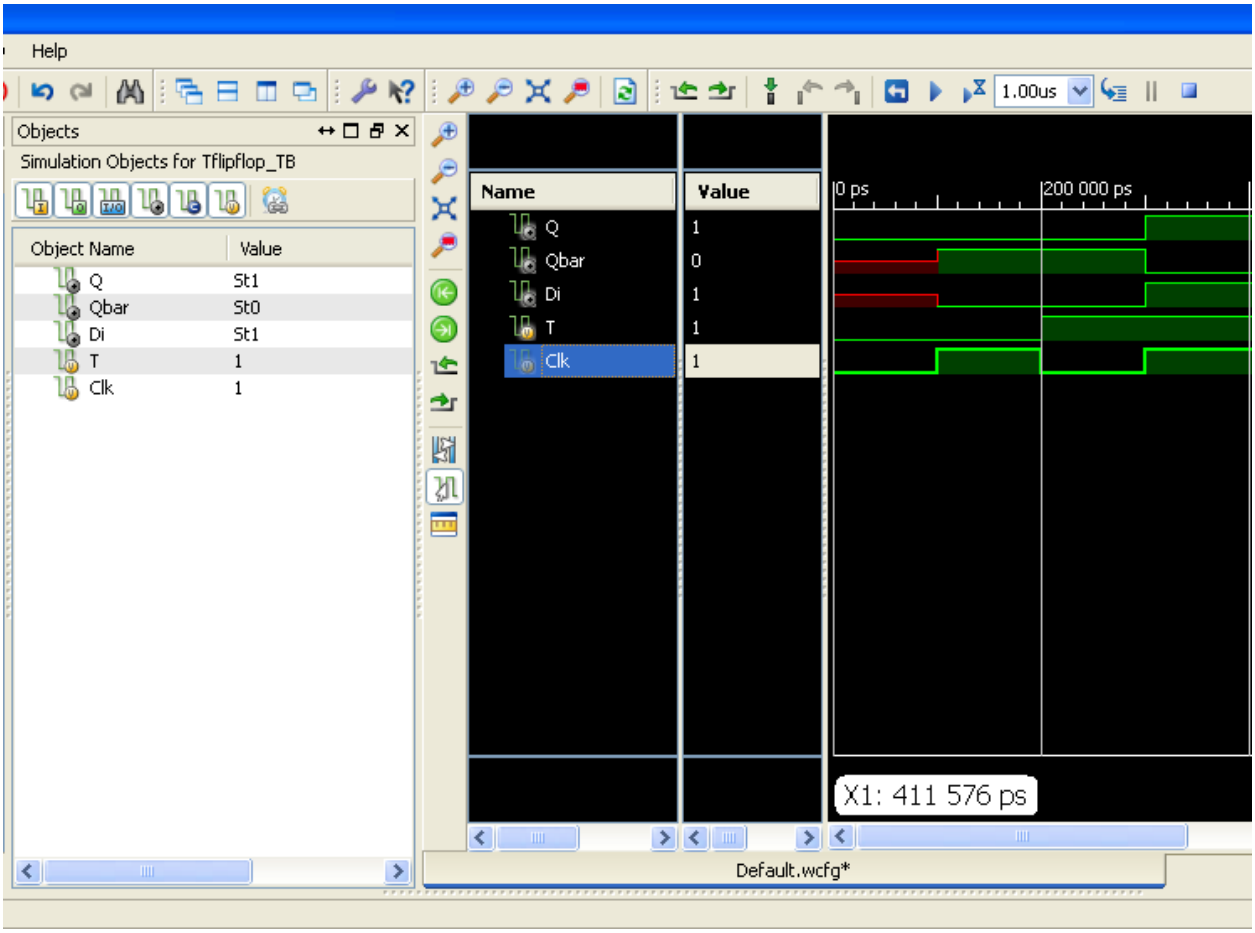
```
`timescale 1ns/1ps
module JKflipflop(
   input wire J,
   input wire K,
   input wire Clk,
   output reg Q=0,
   output reg Qbar=1
   );
reg a,b;
always@(posedge Clk)
begin
a=Q;  b=~Q;
case({J,K})
        'b00: {Q,Qbar} = {a,b};
        'b01:
                begin
                Q=0; a=Q; b=~Q; Qbar =b;
                end
        'b10:
                begin
                Q=1; a=Q;   b=~Q;  Qbar =b;
                end
        'b11: {Q,Qbar} = {b,a};
endcase
end
endmodule
```



ion Manager to check out a full ISim license.
er to the ISim documentation for more information on the differences between the Lite and the Full version.

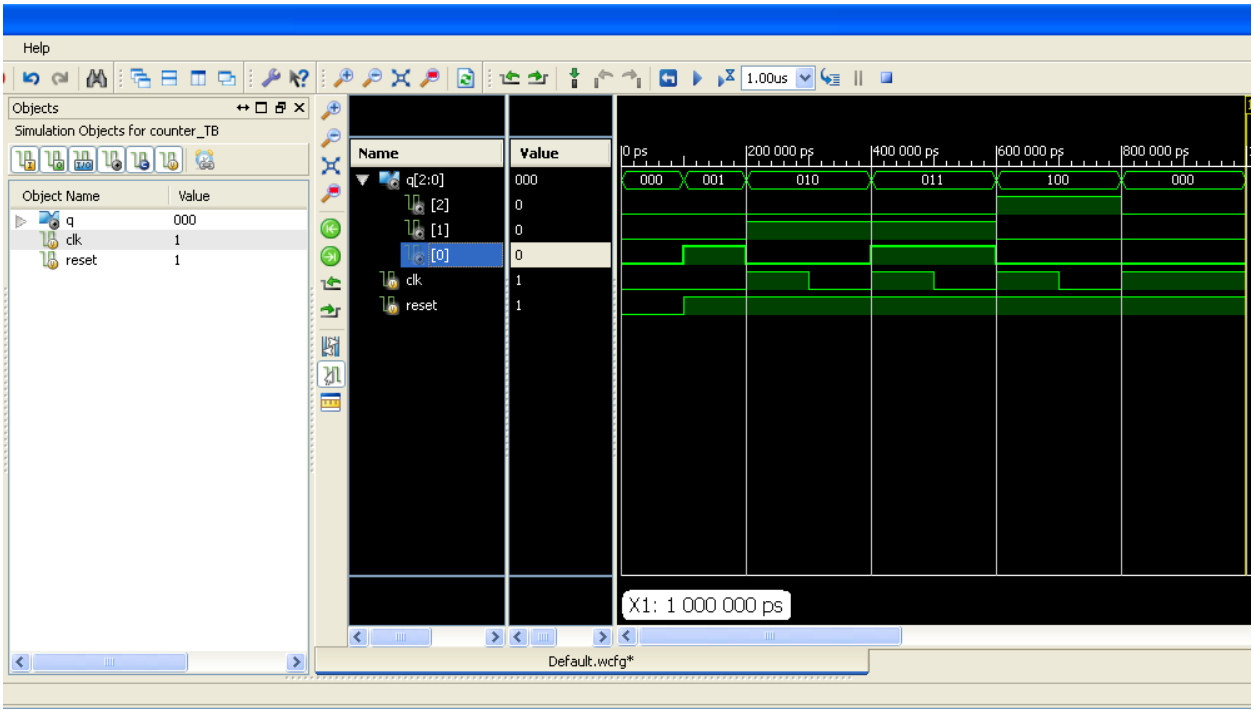
7.9.4 T FLIPFLOP

```
`timescale 1ns/1ps
module Tflipflop(
    input T,
    input Clk,
    output reg Q=0,
    output reg Qbar,
    output reg Di
    );
always@(posedge Clk)
begin
Di=T^Q;  Q=Di; Qbar=~Q;
end
endmodule
```



7.10 Counter

```
`timescale 1ns/1ps
module counter(
    input clk,
    input reset,
    output reg [2:0] q
    );
always @(posedge clk,reset)
if ((reset==0)|(q>=3'b 100))
q=3'b000;
else  q = q+1;
endmodule
```

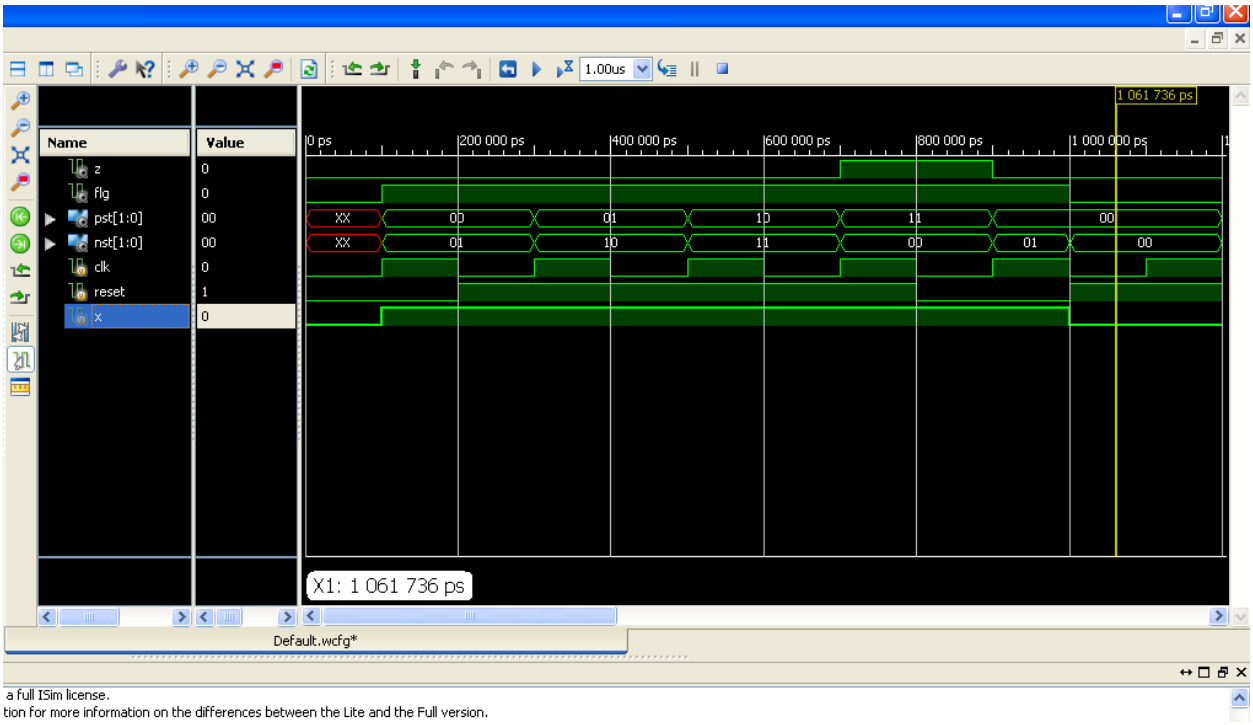


7.11 Finite State Machine

A finite state machine is implemented which will output(z) 'high' if all the 4 states (a,b,c,d) are executed. For a change of state an output is flagged (flg) indicating the same

```
`timescale 1ns/1ps
module fsm(clk, reset, x, z, flg, pst, nst);
   input clk;
   input reset;
   input x;
   output reg z=0;
                output reg flg=0;
         output reg[1:0]pst,nst;

parameter a=2'b00;
parameter b=2'b01;
parameter c=2'b10;
parameter d=2'b11;

always@(posedge clk)
if(reset==0)
pst<=a;

else
pst<=nst;
always@(x or pst)
case({pst})
a:
if(x) begin
nst=b; z=0; flg=1;
end
else begin
nst=a;z=0;flg=0;
end

b:
if(x) begin
nst=c; z=0; flg=1;
end
else begin
nst=b;z=0;flg=0;
end

c:
if(x) begin
nst=d; z=0; flg=1;
end
```

```
else begin
nst=c;z=0;flg=0;
end

d:
if (x) begin
nst=a; z=1;flg=1;
end
else begin
nst=d;z=0; flg=0;
end
endcase
endmodule
```



8 LAB EXERCISES

2. Implement 3-to-8 decoder
3. Implement 3-to-8 decoder using 2-to-4 decoder
4. Implement 8-to-1 mux using 4-to-1 mux
5. Implement 4bit Gray to Binary Converter
6. Implement 1-to-4 Demux
7. Implement 1-to-8 de-mux using 1-to-4 demux
8. Implement 8bit Comparator
9. Implement full-adder using half-adders
10. Implement full-adder using mux
11. Implement UP-DOWN Counter
12. Implement FSM for the sequence 10111010

9 Work Book and Observation