

CREATE A LIBRARY:

In virtuoso's top menu, select "**Tools**" and select "**Library Manager**". The Cadence Library Manager appears as given in Figure-7(b).

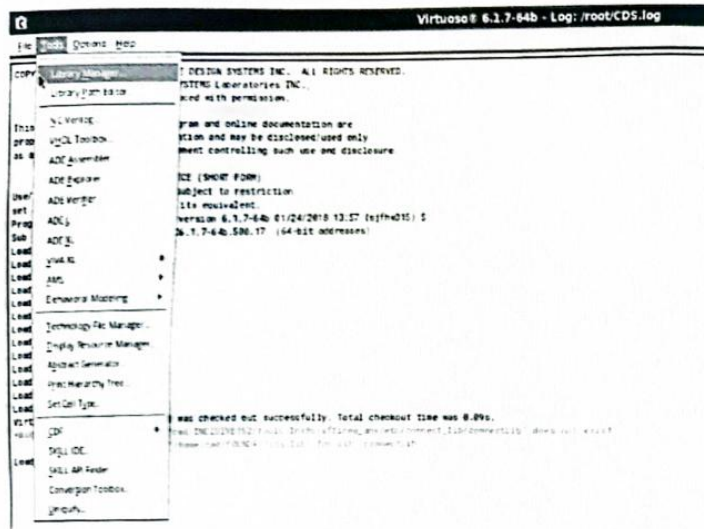


Figure-7(a)

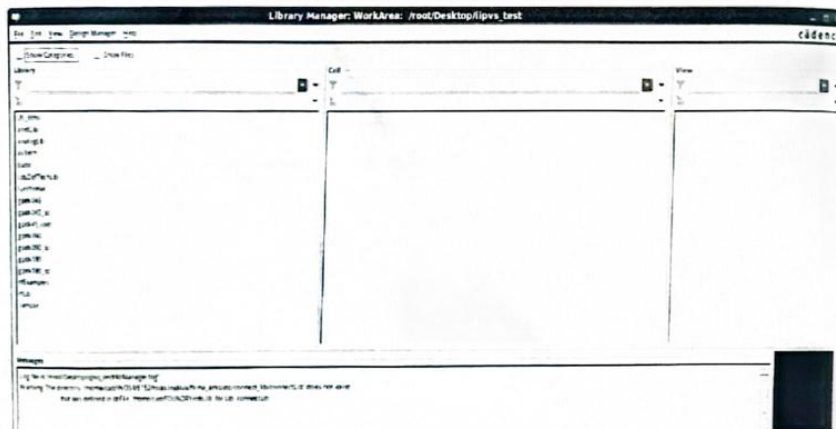


Figure-7(b)

From the top menu of the Library Manager, select **File -> New -> Library** to create a new library for a new design as given in Figure-8.

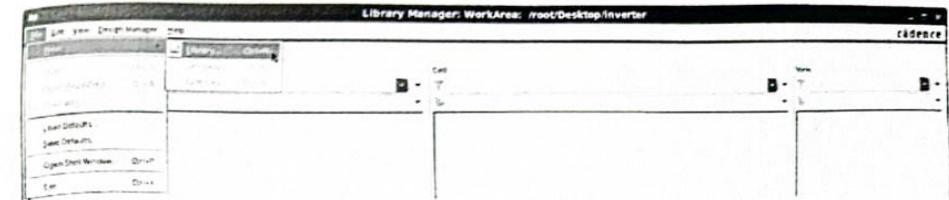


Figure-8

We'll get a "New Library" form where we can name the library that we create as given in Figure-9.



Figure-9

Select **“OK”** after the library is named.

Select “**Technology File..**” tab that keeps blinking at the bottom of the screen (Figure-10) to map the created library to a technology node based on the specification.

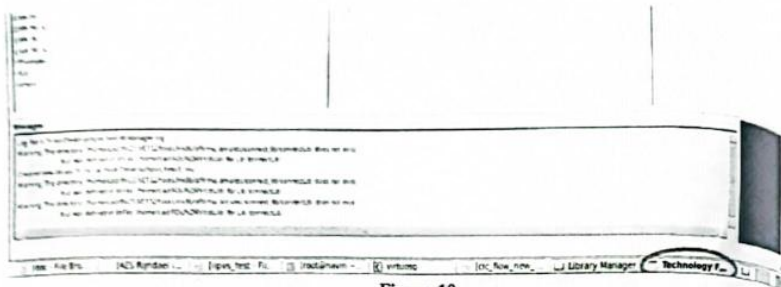


Figure-10

We get a form "Technology File for New Library". Select the option "Attach to an existing technology library" as given in Figure-11.



Figure-11

Select the respective Technology Node from the list of libraries as given in Figure-12. For example, we have selected "gpdk045".

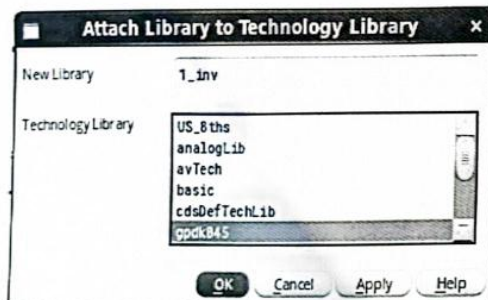


Figure-12

The created library is now available in the Library Manager under Library as given in Figure-13.



Figure-13

CREATE A CELL:

Before creating a cell, make sure that the created library is selected as given in Figure-13. Only then the created cell can be viewed under the respective library.

To create a Cell View, select **File -> New -> Cell View** as in Figure-14.



Figure-14

Now we get a "New File" from as given in Figure-15 and we can name the Cell and check the Library, View and Type of the respective Cell that is to be created.

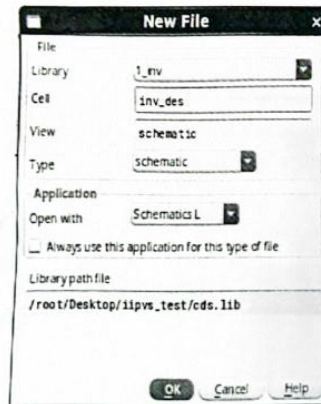


Figure-15

We get the Virtuoso Schematic Editor as in Figure-16.

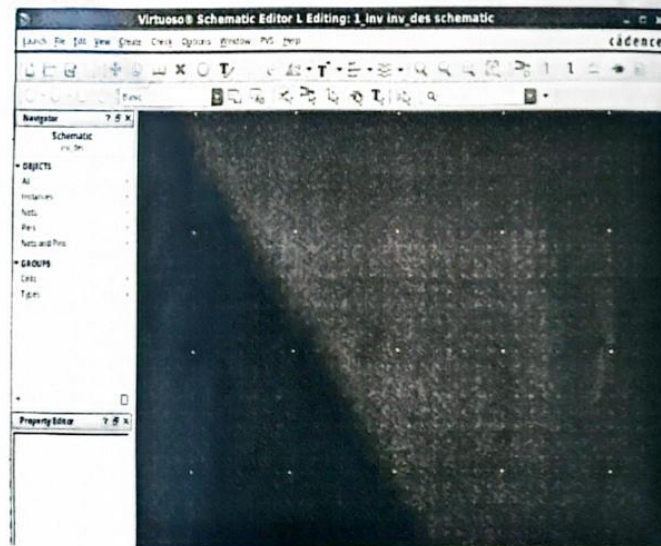


Figure-16

ADD AN INSTANCE:

To add an instance to the circuit, select "Create -> Instance" as in Figure-17. We can also use the bin key 'I' or the icon as in Figure-17(a).

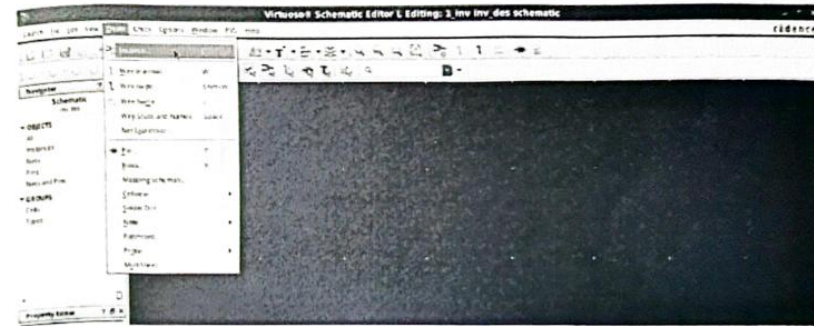


Figure-17



Figure-17(a)

We get the "Add Instance" form as given in Figure-18.



Figure-18

Select "Browse" option as seen in Figure-18, we get the "Library Manager" to choose the transistors required for the circuit and the type of view for the respective component from the respective Technology Library as selected in Figure-12.

After selecting the component, we get its properties like Length, Width, Multipliers, etc., as in Figure-19. Select "Hide" at the end of the form, we can have a view of the transistor in the Schematic Editor as in Figure-20.

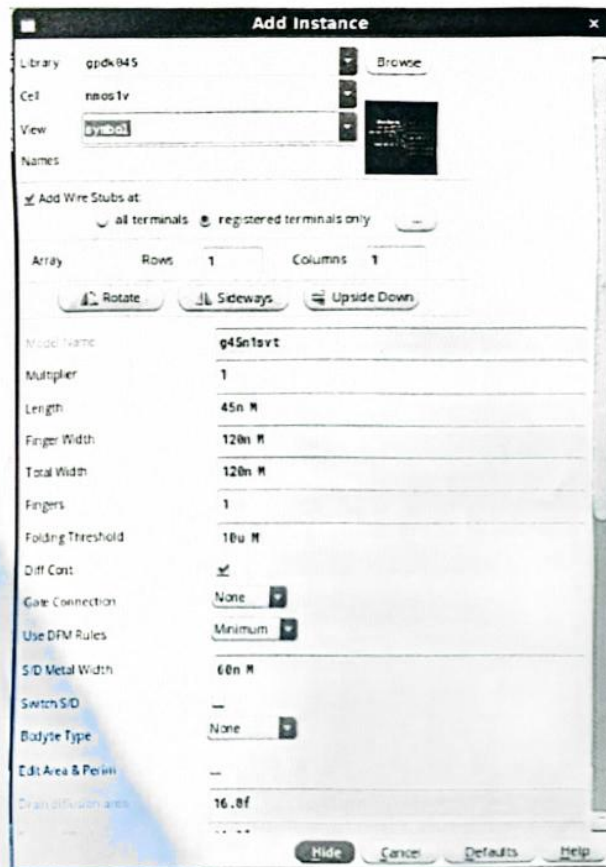


Figure-19



Figure-20

ADD PIN:

To include pins to the circuit, select "Create -> Pin" as in Figure-21. We get a form as in Figure-22. We can also use the bin key 'P' or the icon as in Figure-21(a).

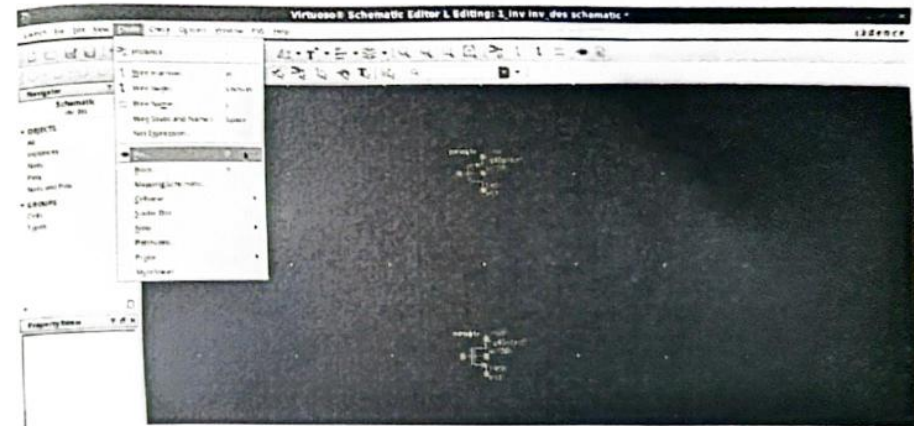


Figure-21



Figure-21(a)

We name the pins and select its direction as in Figure-22(a) and Figure-22(b).



Figure-22(a)



Figure-22(b)

Place the pins by a left click on the mouse and the circuit can be viewed as in Figure-23.

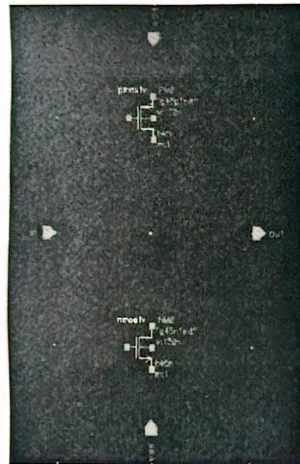


Figure-23

ADD WIRE:

Connect the pins and the components with the help of wire. Select "Create -> Wire" as in Figure-24. We can also use the bin key 'W' or the icon as in Figure-24(a).

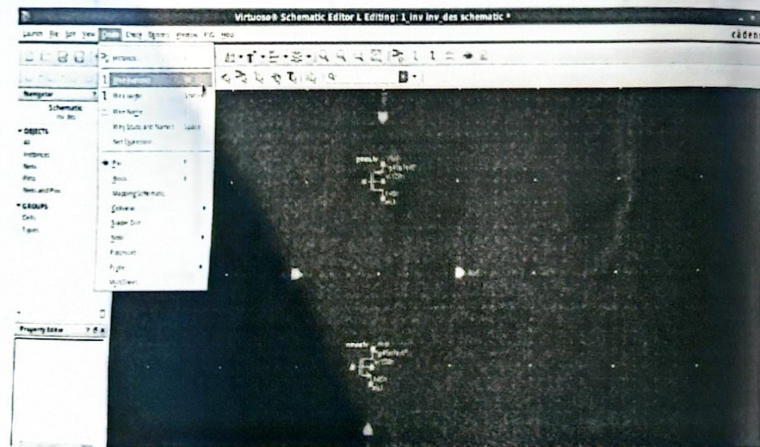


Figure-24

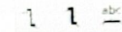


Figure-24(a)

Once wiring is completed, the circuit can be viewed as in Figure-25.

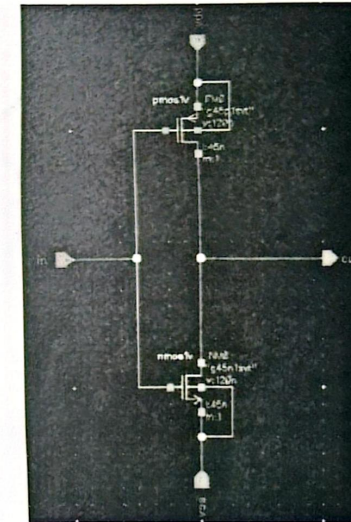


Figure-25

SAVE THE DESIGN:

After completing the design, it is mandatory to save the design before we move ahead to Simulation. We have two options, "Save" and "Check and Save" as in Figure-26.



Figure-26

Save option saves your design as it is but Check and Save option checks for any discontinuities like floating net or terminal, provides the details of errors or warnings in the circuit and then saves the design. The results can be seen in the virtuoso tab as in Figure-27.

```
INFO (SCH-1179): Extracting "inv_des schematic"
INFO (SCH-1426): Schematic check completed with no errors.
INFO (SCH-1181): "1_inv inv_des schematic" saved.
```

Figure-27

SYMBOL CREATION:

To create a symbol for the circuit, select "Create -> Cellview -> From Cellview" as in Figure-28.

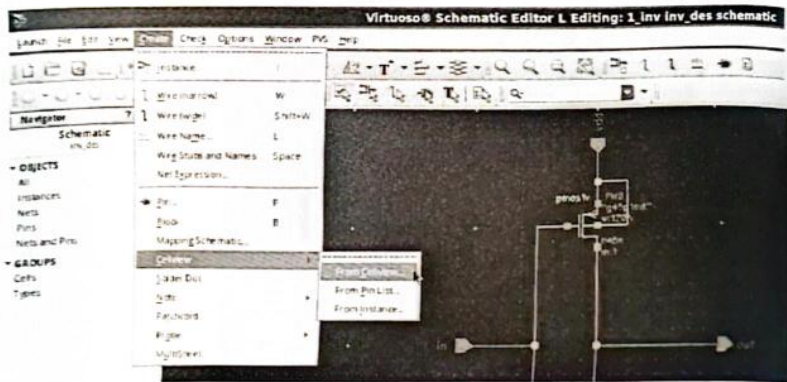


Figure-28

Check the Library Name, Cell Name, From View Name, To View Name, etc., as in Figure-29 and select OK.

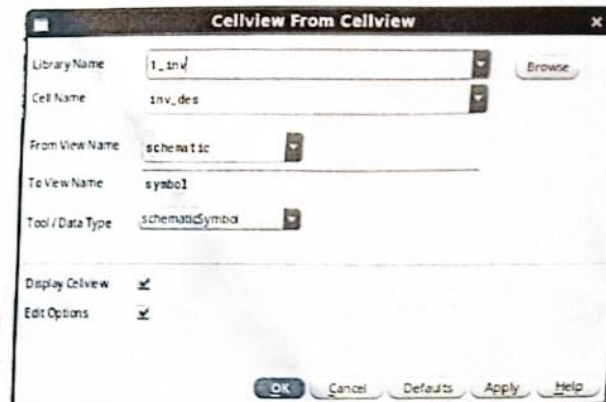


Figure-29

We get the "Symbol Generation Options" window as given in Figure-30. We are free to decide upon the pin locations with options like Left Pins, Right Pins, Top Pins and Bottom Pins. Note: Pin Names are case sensitive.



Figure-30

After naming the Left, Right, Top and Bottom Pins as in example shown below (Figure-31), select OK. The tool opens a "Virtuoso Symbol Editor" window which shows a temporary view of the symbol based on the pins assigned as given in Figure-32.

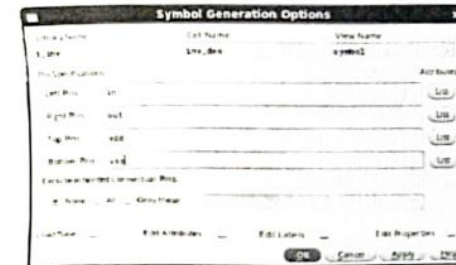


Figure-31

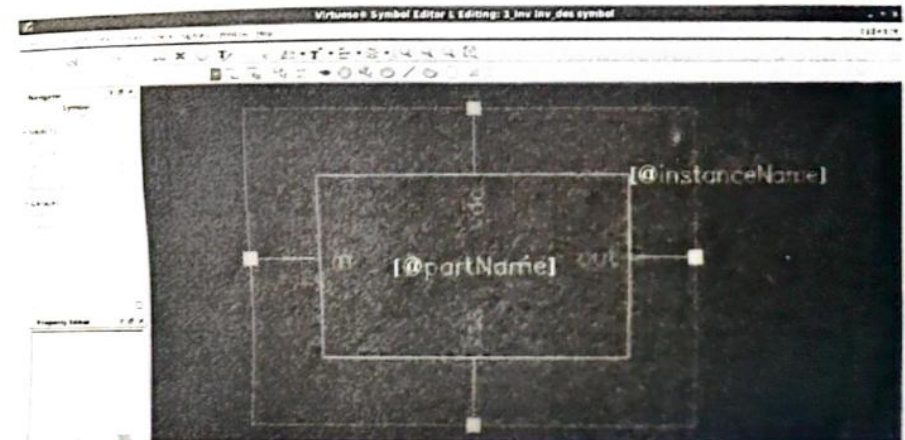


Figure-32

The symbol shown in Figure-32 can be customized with the help of drawing tools as given in Figure-33.



Figure-33

To create custom symbol, remove the inner rectangle (green) and then using drawing tools, custom symbol can be created. To remove the inner rectangle (green), place the mouse pointer within the rectangle so that the entire rectangle gets highlighted as in Figure-34(a), make a left click so that the entire rectangle gets selected as in Figure-34(b) and click on delete in the keyboard to remove the rectangle so that it is removed as in Figure-34(c).

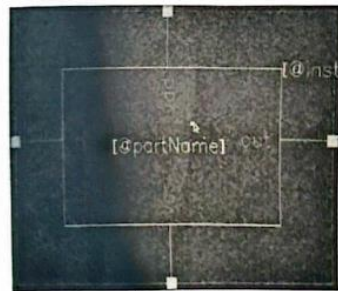


Figure-34(a)

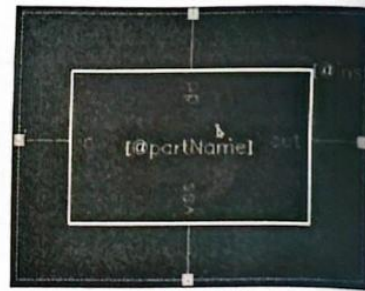


Figure-34(b)

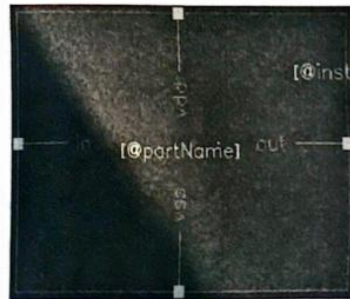


Figure-34(c)

Since Inverter design is taken as an example in this manual, the focus is on creating its symbol. To create a triangle, use "Create Line" option as in Figure-35(a) among the drawing tools and the way triangle is created is as similar as how wiring is done in the schematic. To create a bubble, use "Create Circle" option as in Figure-35(b), decide a point, make a left click to expand the circle and again make a left click to fix its size as in Figure-35(c).



Figure-35(a)

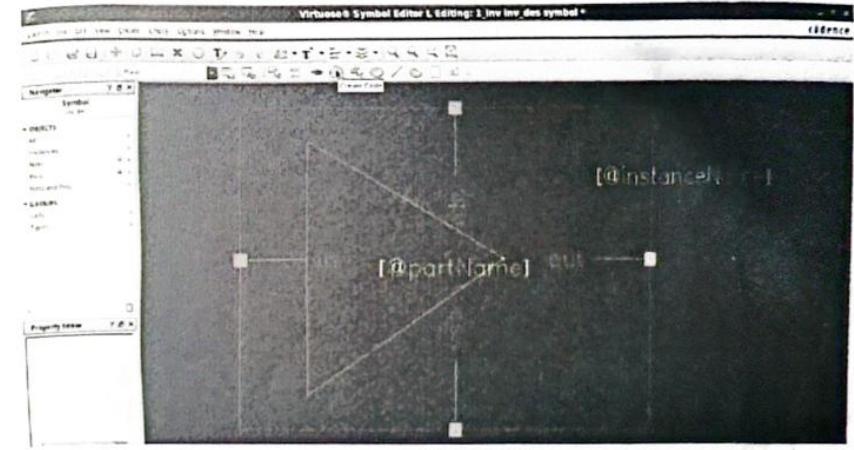


Figure-35(b)

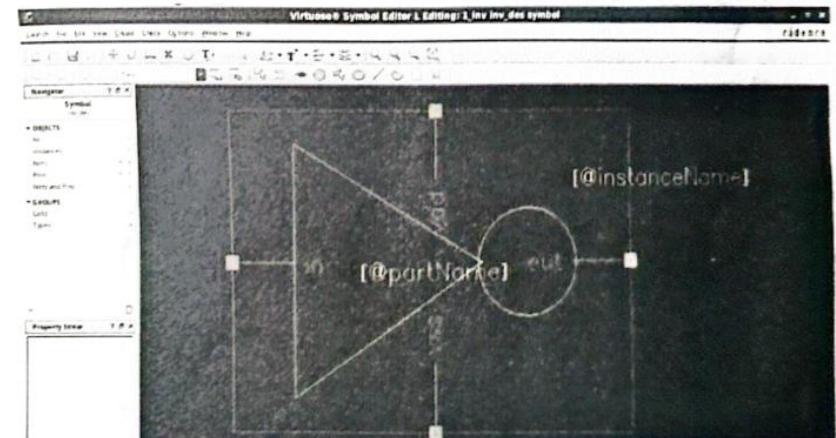


Figure-35(c)

After creating the symbol, save the symbol using the “Check & Save” option as in Figure-26.

TESTBENCH CIRCUIT FOR SIMULATION:

Instead of using transistor level design, testbench circuit can be created using its symbolic representation. To create a testbench circuit, a new Cellview with a different Cell Name should be created as in Figure-36.

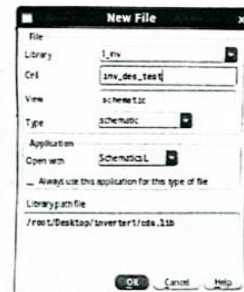


Figure-36

To include the created symbol, use the “Add Instance” option as in Figure-17 & 17(a) and select the respective library and cell as in Figure-37 using the browse option and then the symbol should appear in schematic editor with the mouse pointer.

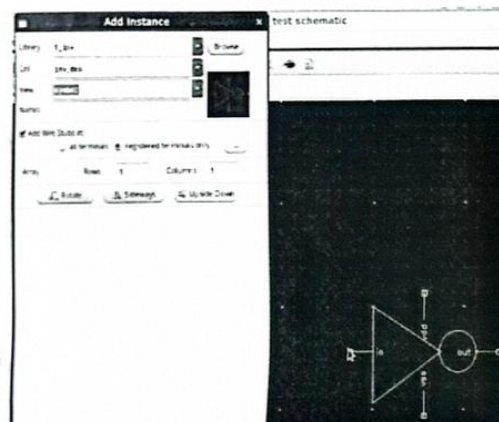


Figure-37

Create an output pin as in Figure-38(a) and 38(b).

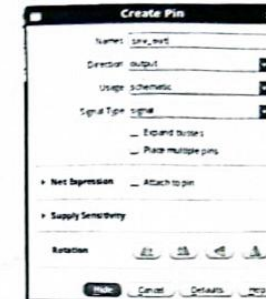


Figure-38(a)

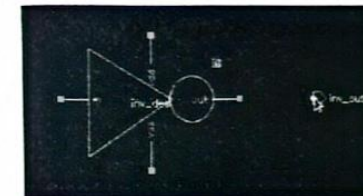


Figure-38(b)

To include supply voltage and input signal sources, “Add Instance” option shall be used. Browse to the “analogLib” library, select the Cell as “vdc”, View as “symbol” and click on “Tab” key in the keyboard to get the device properties as in Figure-39.



Figure-39

Mention the appropriate "DC voltage" based on the specification or technology node. Similarly, for an input source, (for example) "vpulse" is considered and parameters like Voltage 1, Voltage 2, Period, Delay time, Rise time, Fall time and Pulse width are mentioned as in Figure-40.



Figure-40

Similarly, "gnd" terminal shall be included to the circuit as in Figure-41 and the complete circuit is given in Figure-42.

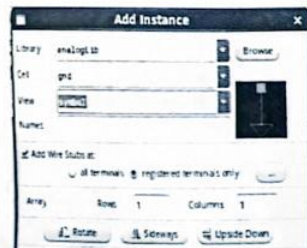


Figure-41

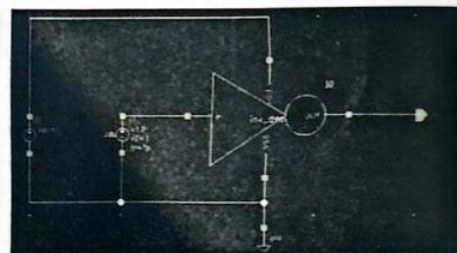


Figure-42

FUNCTIONAL SIMULATION:

To simulate the design, Launch "ADE L" as in Figure-43(a).

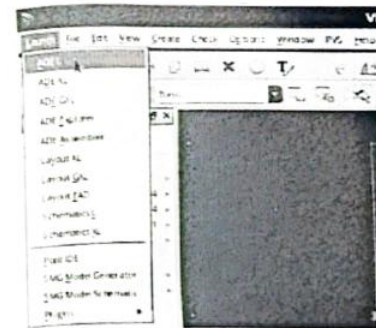


Figure-43(a)



Figure-43(b)

We get a window as in Figure-43(b) and following options are to be verified:

- (1) **Simulator** – to make sure that Spectre is the simulator selected [Figure-44(a) & 44(b)]

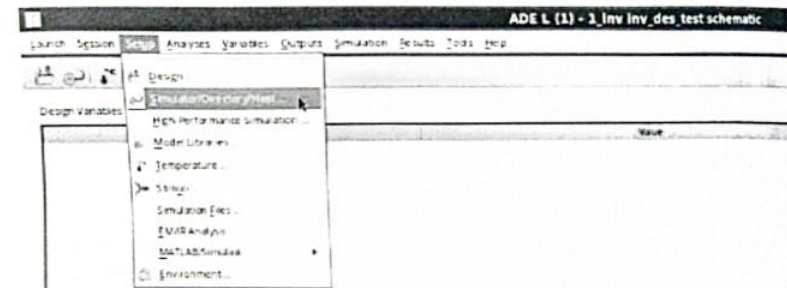


Figure-44(a)



Figure-44(b)

(2) Model Libraries & Process Corners – to make sure that “.scs” file of the respective technology node has been selected [Figure-45(a) & 45(b)]

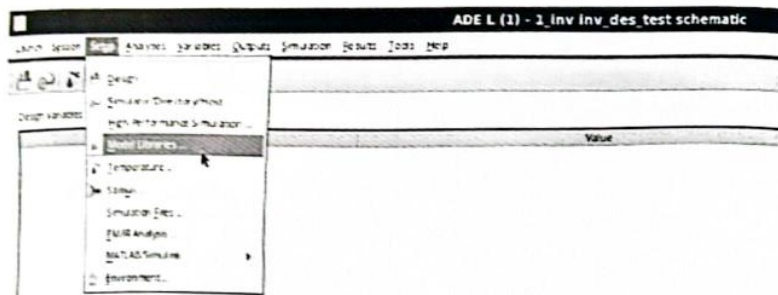


Figure-45(a)



Figure-45(b)

To analyze the circuit, select “Analyses” from the top menu in ADE L and under Analyses, select “Choose” as in Figure-46.

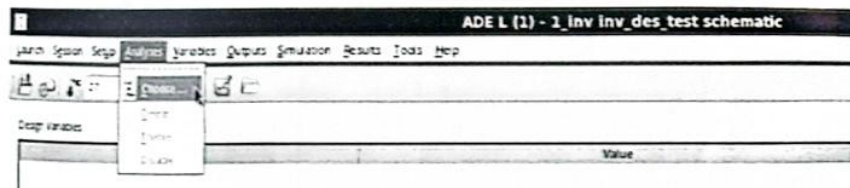


Figure-46

To set up a “Transient Analysis”, select “tran” under “Analysis” as in Figure-47(a), mention the “Stop Time” and “Accuracy Defaults” as in Figure-47(b) and select “OK”.

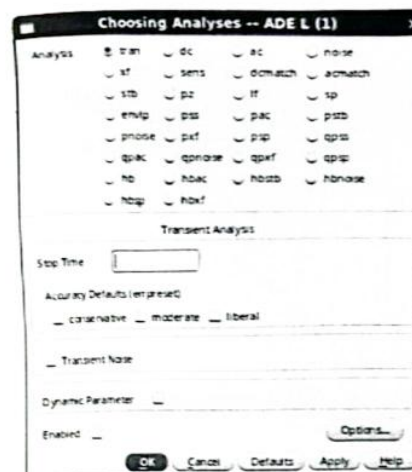


Figure-47(a)

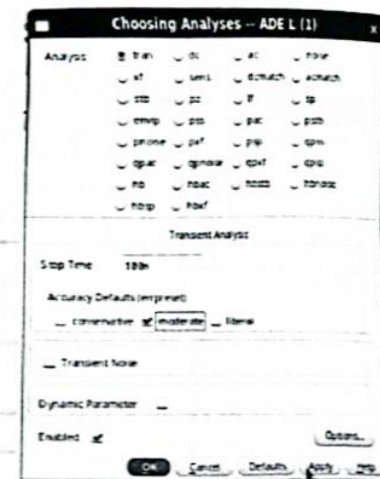


Figure-47(b)

The analyses chosen and the arguments that had been set up can be seen under “Analyses” tab in the ADE L window as in Figure-47(c). Similarly, rest of the analyses can be performed based on designer’s demands.

Analyses			
Type	Enable	Arguments	
1 tran	☒	0 100n	moderate

Figure-47(c)

To setup the simulation, select “Outputs” from the ADE L window and select “Setup” under Outputs option as in Figure-48.

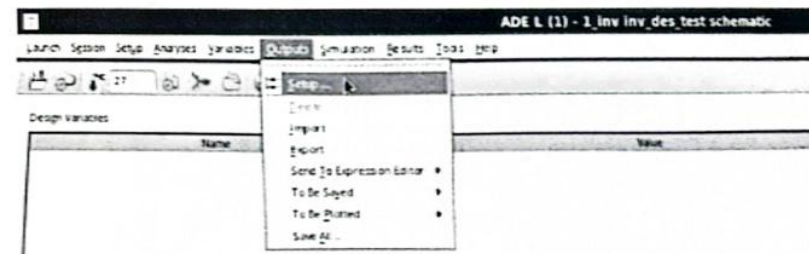


Figure-48

We get a “Setting Outputs” window as in Figure-49, select “From Design” which would bring back the testbench circuit created for the simulation process.

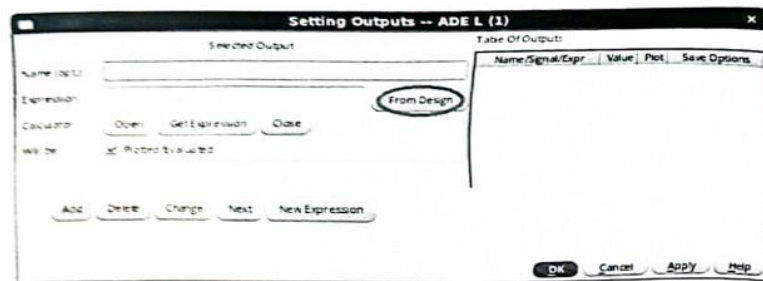
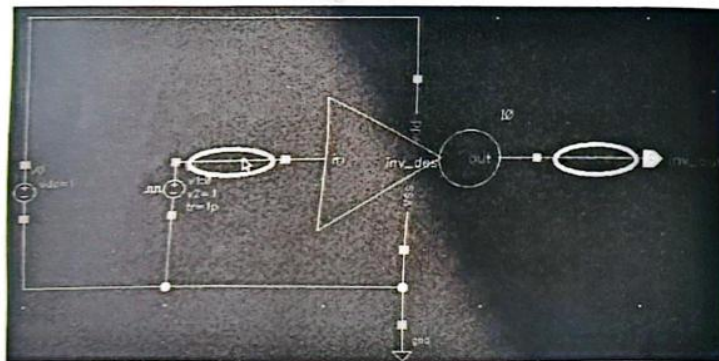


Figure-49

Select the input wire and output wire and in ADE L, they are supposed to be displayed under "Outputs" option as in Figure-50.



Outputs				
Name/Signal/Expr	Value	Plot	Save	Save Options
1 net5		☒	☐	allv
2 inv_out		☒	☐	allv

Figure-50

To run the simulation, select "Simulation" from ADE L and select the option "Netlist and Run" as in Figure-51. The design is simulated and waveforms are seen as in Figure-52.



Figure-51

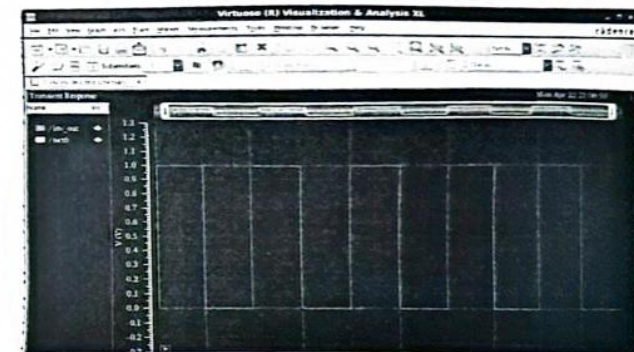


Figure-52

Input signals and Output signals can be seen separately by selecting "Graph -> Split All Strips" as in Figure-53.



Figure-53

To save the simulation state, select "Session" under ADE L, select "Save State" as in Figure-54, we get a "Saving State" window as in Figure-55. Select "Cellview" and select "OK" to save the state.

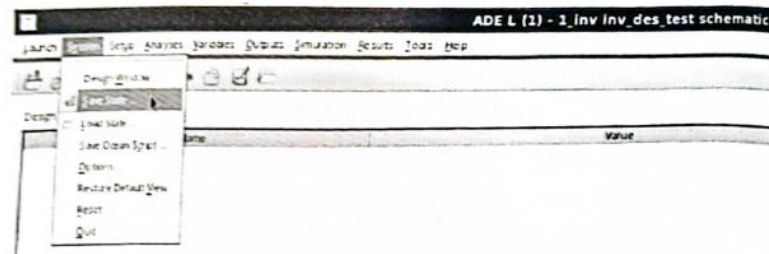


Figure-54

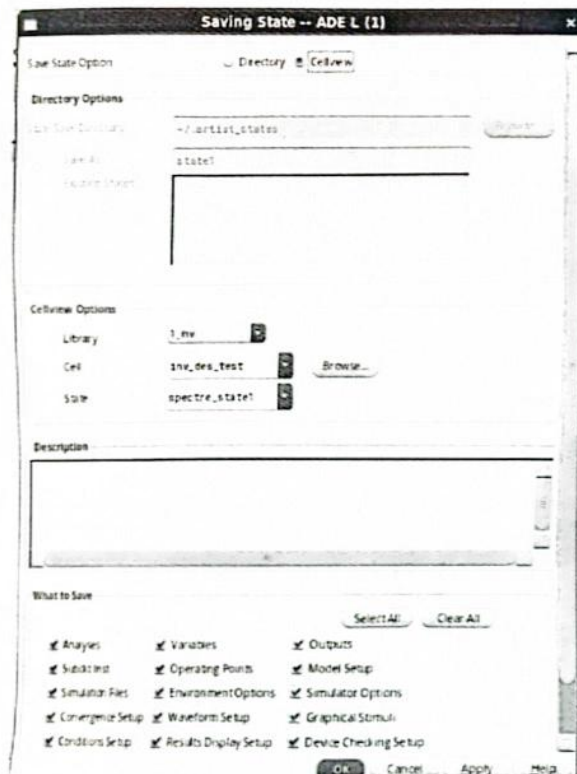


Figure-55

The simulation state and testbench circuit are viewed as in Figure-56.



Figure-56

LAYOUT:

Open the schematic as in Figure-25 through the Library Manager and select "Launch -> Layout XL" as in Figure-57.

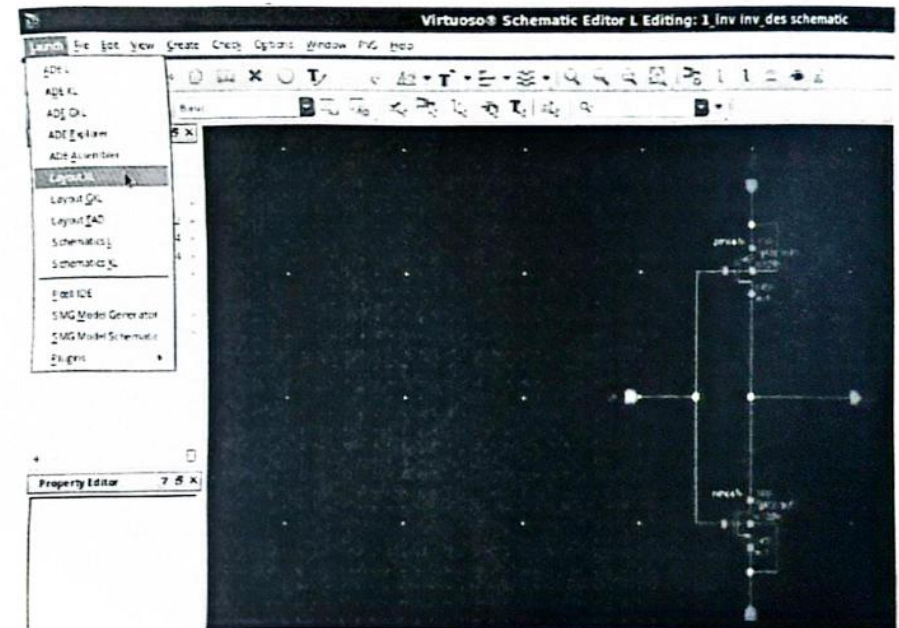


Figure-57

A "Startup Option" window pops up as in Figure-58. Select "Create New" under Layout tab, "Automatic" under Configuration tab and select OK.

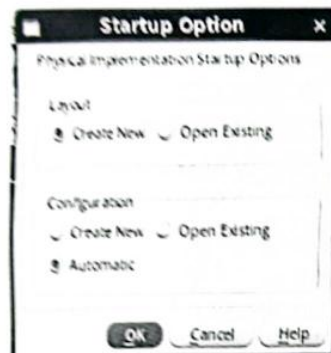


Figure-58

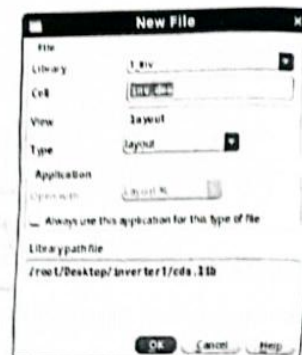


Figure-59

A "New File" window pops up as in Figure-59. No changes are to be made, so select OK. Virtuoso Layout Suite XL Editing window can be seen as in Figure-60.

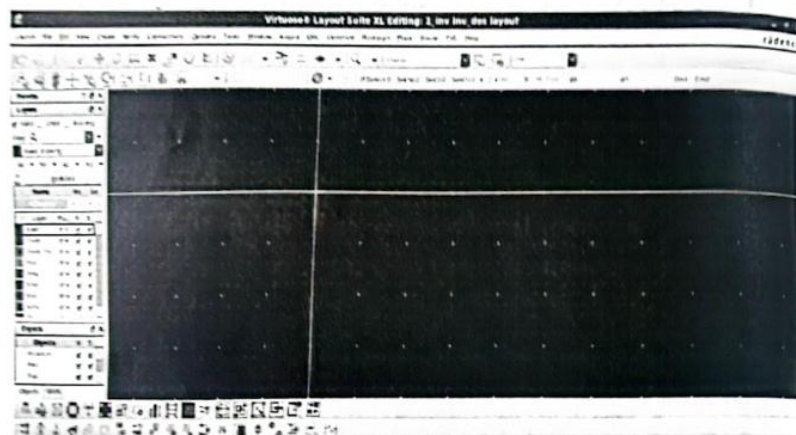


Figure-60

To import the circuit components, select "Connectivity -> Generate -> All From Source" as in Figure-61.

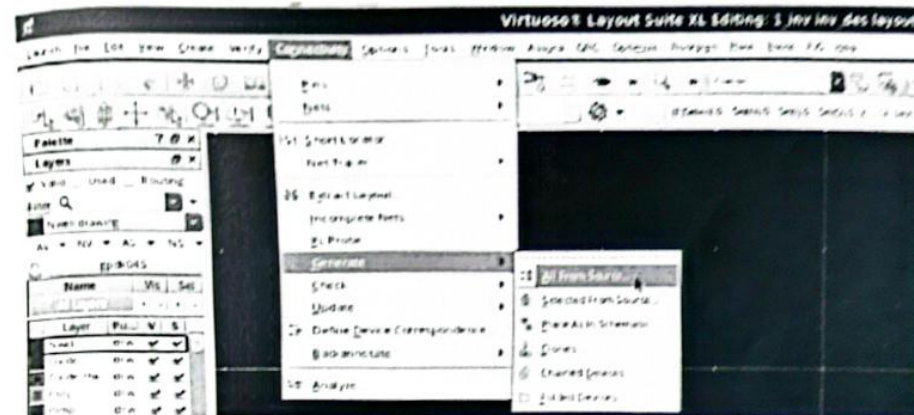


Figure-61

Select OK in the "Generate Layout" window as in Figure-62, and the components of the circuit can be seen as in Figure-63.

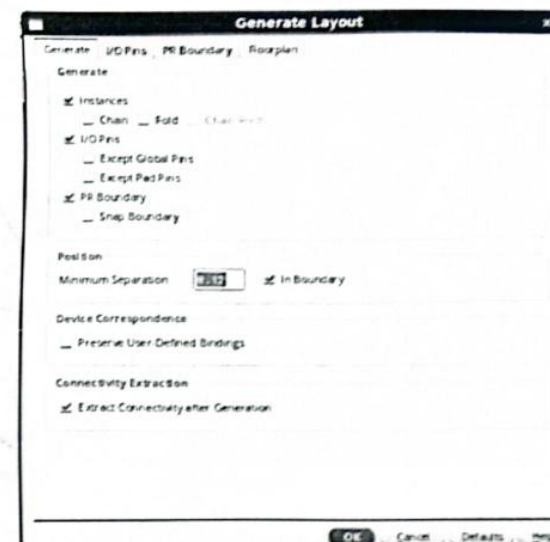


Figure-63

Select "Shift + F" to have a view of all the terminals of the transistors as in Figure-64.

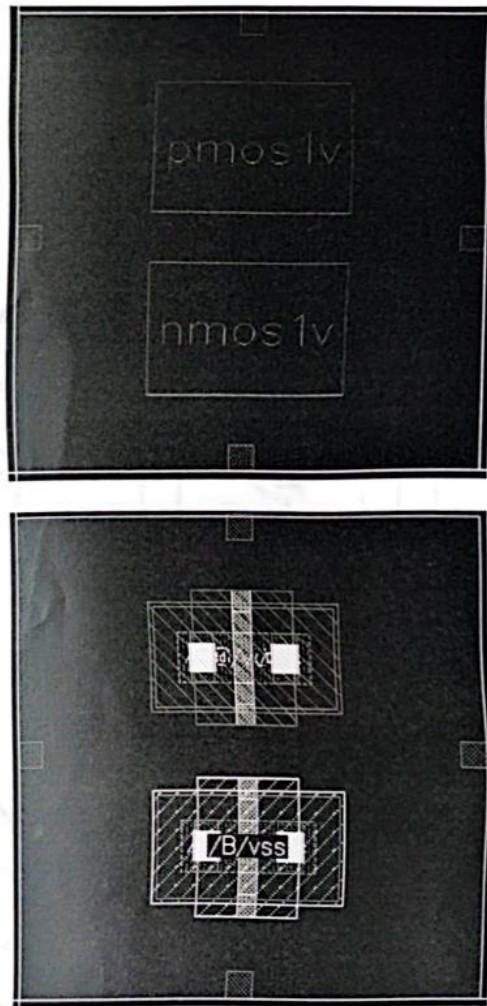


Figure-64

The blue colored box surrounding the circuit components is the **PR (Placement & Routing) Boundary**. PR-Boundary can be shifted using the shortcut "S". To shift the PR-Boundary, place the mouse pointer close to the respective boundary so that it gets highlighted as in Figure-65, make a left mouse click to select the boundary, use the mouse or arrow (up, down, left or right) keys to shift the boundary.

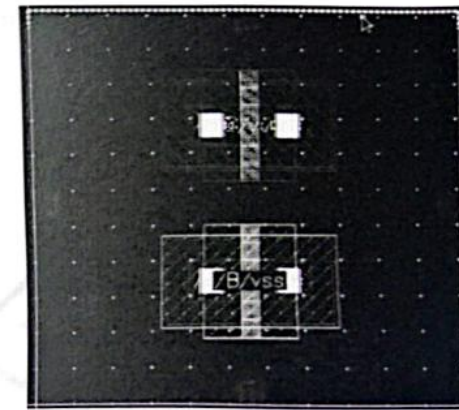


Figure-65

To attach "Bulk" terminal to the transistors, select the particular transistor by a left mouse click, make a right mouse click and select "Properties" as in Figure-66(a), select "Parameter -> Bodytie Type -> Integrated (or) Detached -> Left Tap (or) Right Tap" as in Figure-66(b) and select OK. Bulk terminal of that transistor can be seen as in Figure-66(c). Repeat the above steps for rest of the transistors to include the Bulk terminal.

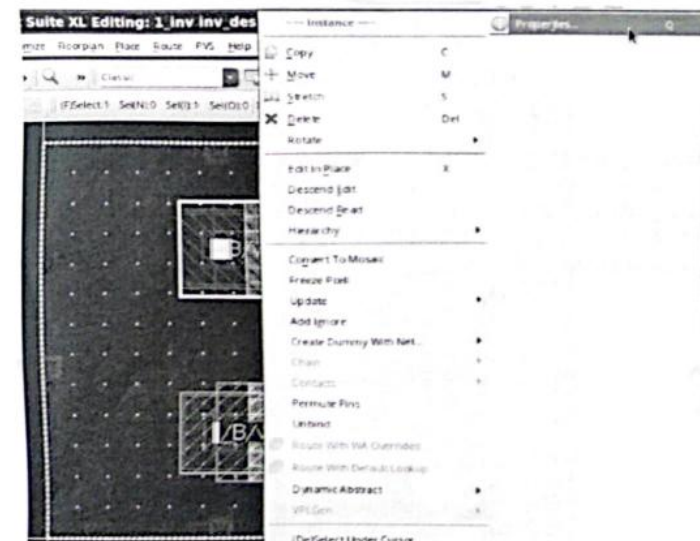


Figure-66(a)

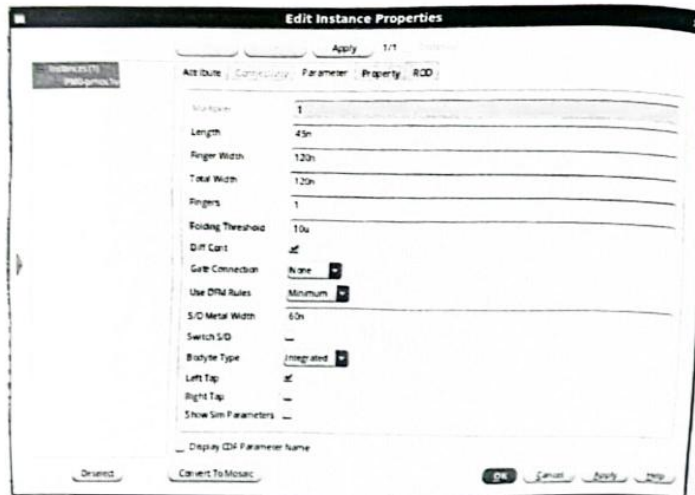


Figure-66(b)

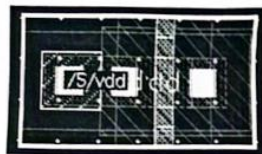


Figure-66(c)

By default, the background available in the Virtuoso Layout Editor is the P-Substrate. So, the PMOS transistors should be placed within an N-Well. This is mandatory for PMOS transistors in gpd090 and gpd045, but for PMOS transistors in gpd180, it has an N-Well by default. To create an N-Well, select the respective layer from the "Layers" tab on the left-hand side of the Virtuoso Layout Editor as in Figure-67.

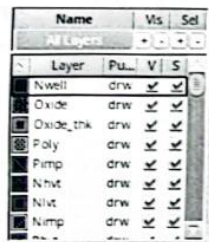


Figure-67

Select "R" to create N-Well in the shape of a rectangle surrounding the PMOS transistor as in Figure-68.

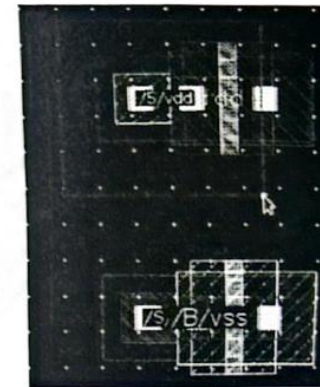


Figure-68

The PMOS transistor is placed within the N-Well as in Figure-69 and to complete routing between the components, we can use "P" which denotes "Path" in the layout and is meant for routing process.

When the mouse pointer is taken closer to any of the terminals of the components, the terminal gets highlighted with a yellow color bold line, make a left click to start the path segment from one terminal to another as in Figure-69.

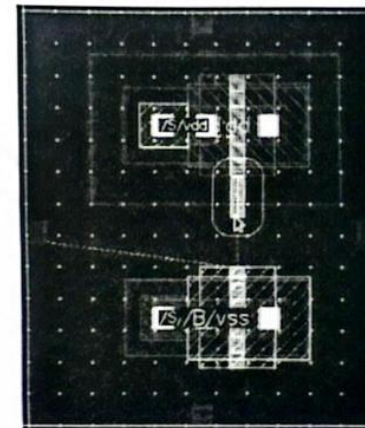


Figure-69

For routing between two different layers, usage of "Via" is mandatory. There are two options to place a via. Start routing from the layer of interest, make a right-click, select "Via Down To..." and select the layer of interest as in Figure-70(a) (or) select "Create" from the top menu, select "Via", select the respective via under "Via Definition", make a left mouse click to place the via as in Figure-70(b) & 70(c).

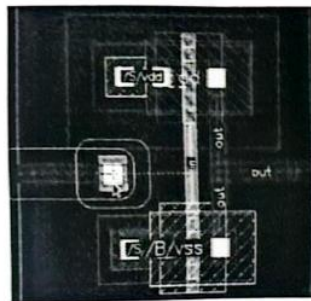
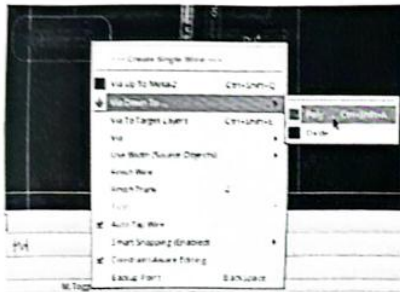


Figure-70(a)

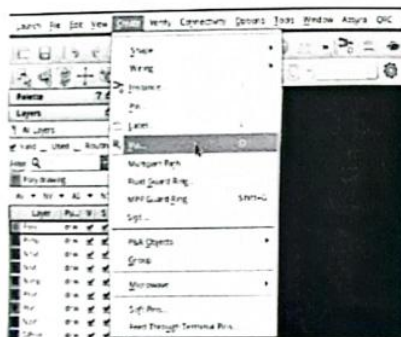


Figure-70(b)

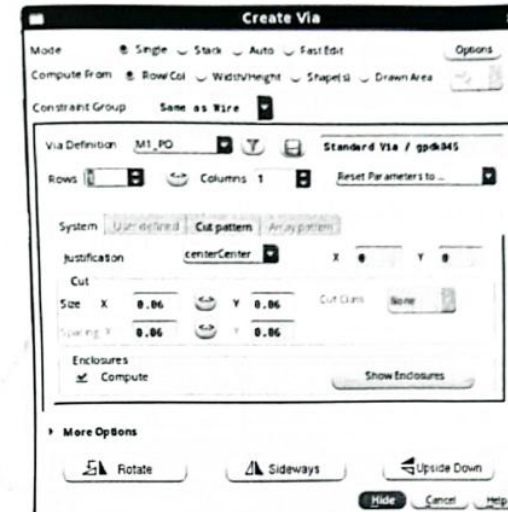


Figure-70(c)

For the example used here, "VDD" is at the top and "VSS" is at the bottom. Source & Bulk of PMOS/NMOS can be routed to VDD/VSS respectively as earlier between other pins and terminals (or) the entire top layer can be used for VDD and bottom layer for VSS using "Pin Placement" option.



Figure-71

Select "Place -> Pin Placement" as in Figure-71, a window pops up as in Figure-72. Under "Pin Planner" tab, "Pin Name, Attributes & Create" options can be seen. Select "VDD" under "Pin Name", select "Top" under "Attributes -> Edge", select "Hrail" under "Create" and select "Apply" under "Attributes" to place VDD at the top layer. Similarly, select "VSS" under "Pin Name", select "Bottom" under "Attributes -> Edge", select "Hrail" under "Create" and select "Apply" under "Attributes" to place VSS at the bottom layer as in Figure-73. After routing the terminals of PMOS/NMOS terminals with VDD/VSS respectively, save the layout.

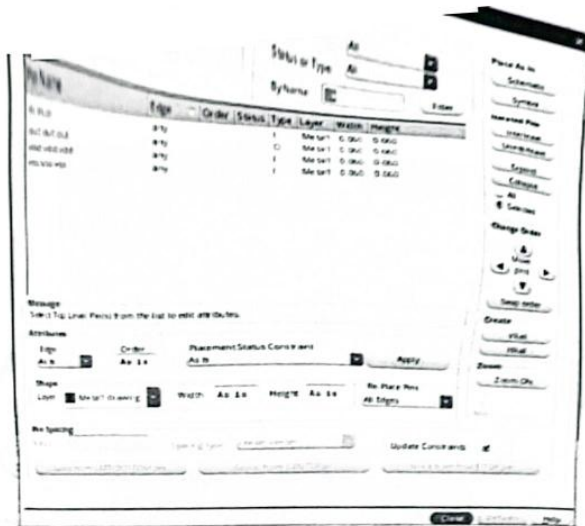


Figure-72

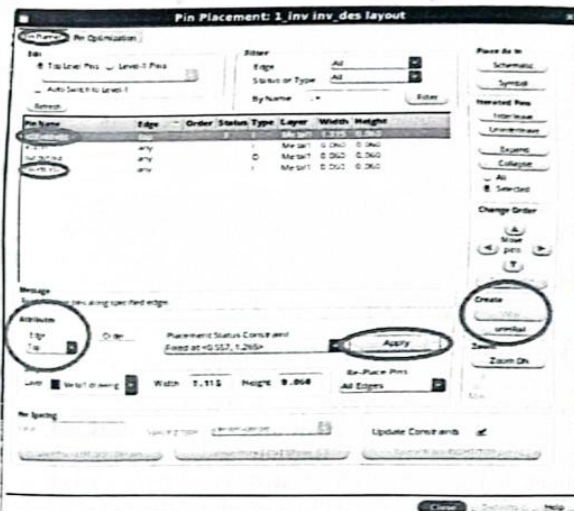


Figure-73

PHYSICAL VERIFICATION:

The tool used for Physical Verification is Assura where DRC, LVS of a design are verified and Parasitics of the design are extracted using QRC for technology nodes like gpd180, gpd190 and gpd405. PVS is another tool for Physical Verification on designs made using technology nodes like gpd405 and below.

PHYSICAL VERIFICATION WITH ASSURA:

The first step in Physical Verification is the Technology Library Mapping.

Technology Library Mapping:

Select "Assura -> Technology", an "Assura Technology Lib Select" window can be seen as in Figure-74.

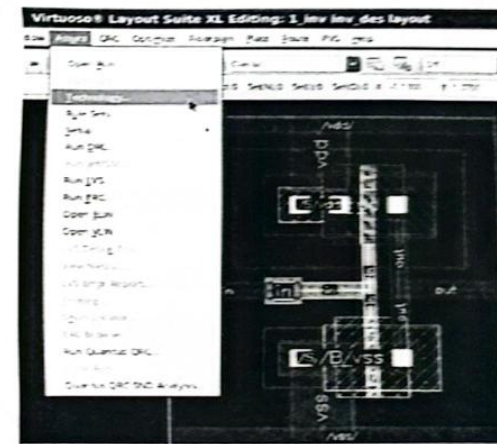


Figure-74

Select "Browse" so that the "File Selector" can be seen as in Figure-75.



Figure-75

Two columns, "Directories" and "Files" can be seen. The "Single Dot" under "Directories" indicates the present working directory, "Double Dot" just below the "Single Dot" is used to

go back to the previous/next directories and "I_inv" indicates the library under which the layout is saved. Use the "Double Dot" option to browse the library file "assura_tech.lib" as in Figure-76, based on the technology node of interest and select OK.



Figure-76

DRC:

To perform the DRC check, select "Assura -> Run DRC" as in Figure-77.

A "Run Assura DRC" window can be seen as in Figure-78. Verify the Library, Cell and View in the second row. Give any name under the "Run Name" option, click on the drop down next to "Technology" and select the "Technology Node" of interest.

The "Run Assura DRC" window after mentioning the options as given above can be seen as in Figure-79.

Select OK to run the DRC. A "Progress" tab can be seen at the right-side bottom corner of the screen as in Figure-80. To check the background process or log file, select "Watch Log File" option under the "Progress" tab. A run completion tab pops-up as in Figure-81(a). Select "Yes" and the DRC run result can be seen as in Figure-81(b).

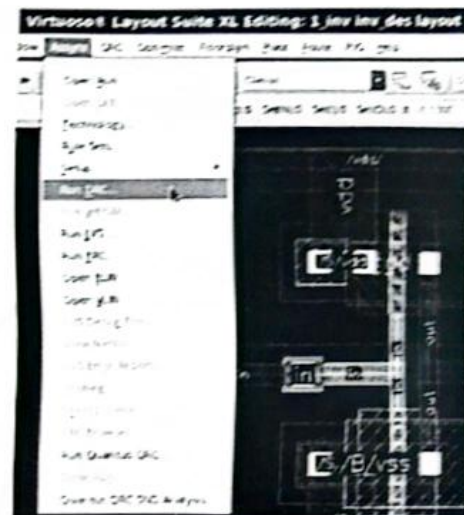


Figure-77

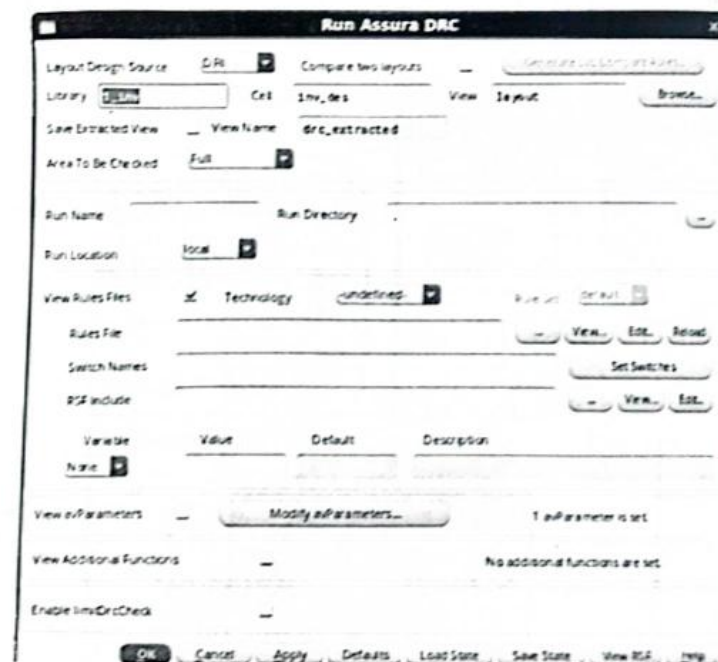


Figure-78

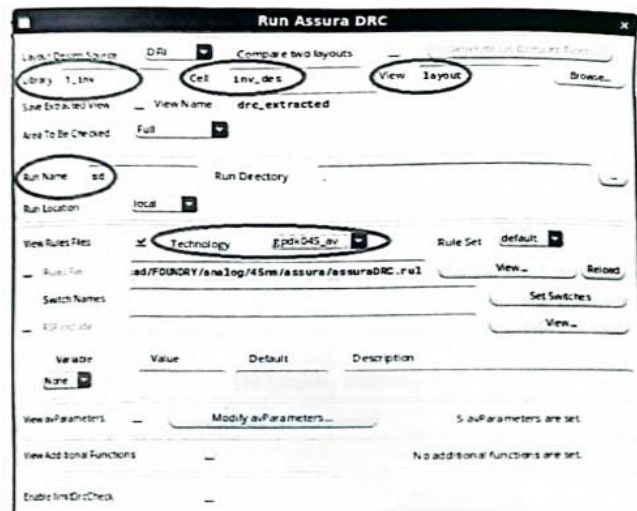


Figure-79

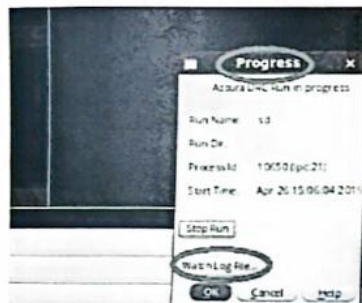


Figure-80



Figure-81(a)



Figure-81(b)

LVS:

To perform the LVS check, select "Assura -> Run LVS" as in Figure-82.



Figure-82

A "Run Assura LVS" window can be seen. Verify the Library, Cell and View for the Schematic Design Source and Layout Design Source. Give any name under the "Run Name" option, click on the drop down next to "Technology" and select the "Technology Node" of



Figure-83

interest and the "Run Assura LVS" window after mentioning the options as given above can be seen as in Figure-83.

Select OK to run the LVS. A "Progress" tab can be seen at the right-side bottom corner of the screen similar to Figure-80. To check the background process or log file, select "Watch Log File" option under the "Progress" tab.

A run completion tab pops-up as in Figure-84(a). Select "Yes" and the DRC run result can be seen as in Figure-84(b).

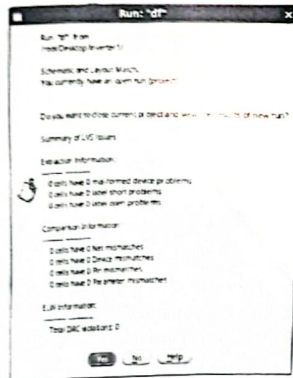


Figure-84(a)

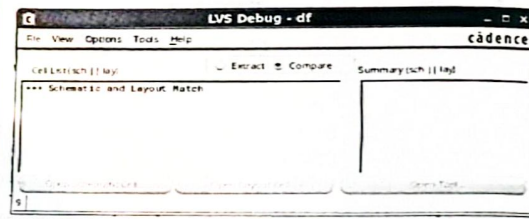


Figure-84(b)

QRC:

To extract the Parasitics, select "Assura -> Run Quantus QRC" as in Figure-85.

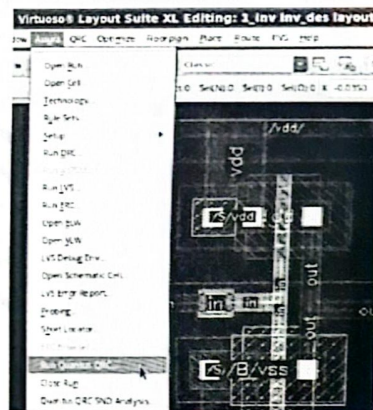


Figure-85

A "Quantus QRC (Assura) Parasitic Extraction Run Form" window can be seen as in Figure-86. Under the "Setup" tab, verify the "Technology", "Output -> Extracted View" and under "Extraction" tab, verify "Extraction Type -> RC", "Ref Node -> vdd! (or) gnd!" as in Figure-87(a) & 87(b). Select OK to extract the Parasitics.

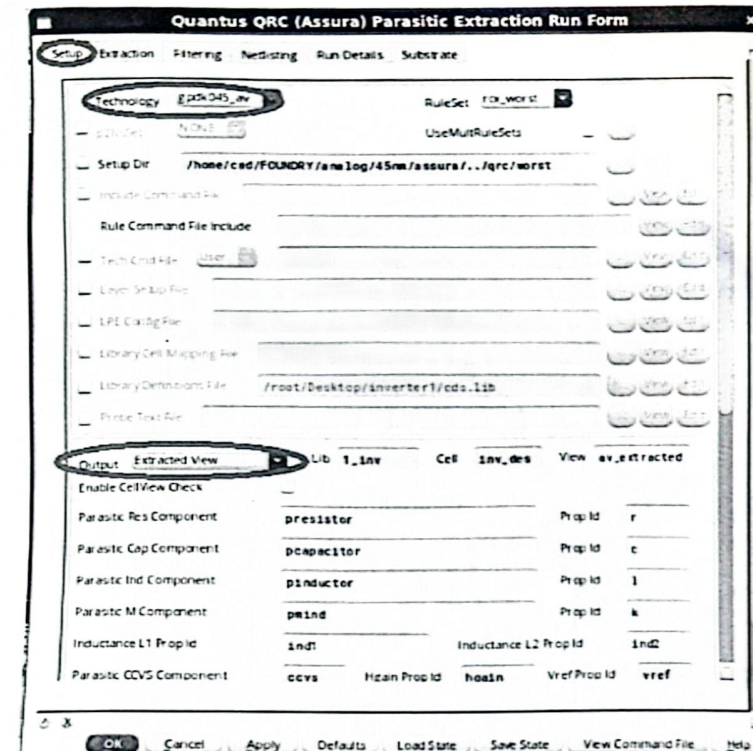


Figure-87(a)

A "Progress" tab can be seen at the right-side bottom corner of the screen as in Figure-80. To check the background process or log file, select "Watch Log File" option under the "Progress" tab.

A run completion tab pops-up as in Figure-88. Select "Close" and the "av_extracted" view can be seen in the Library Manager as in Figure-89.

Make a double click on the "av_extracted" view, a design can be seen as in Figure-90.

Select "Shift+F" on the keyboard and if zoomed into the design with the help of mouse scroller, the parasitic resistor, its value and parasitic capacitor, its value can be seen as in Figure-91.



Quantus QRC Run

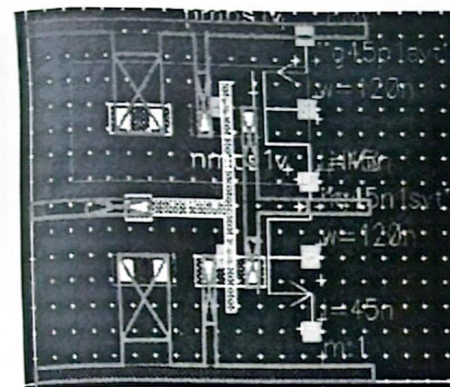
The Quantus QRC run IOT completed successfully.
The output is in:

Library: I:\nv
C:\nv_des
View: by extracted

Close

[illegible]

Copyright © CVR COLLEGE OF ENGINEERING



Copyright © CVR COLLEGE OF ENGINEERING

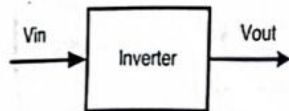
Experiment No: 01 Layout and DRC Verification of CMOS Inverter.

Aim: To design the layout of a CMOS inverter using 90nm technology and perform DRC verification to check for design rule errors.

Tools Required: Cadence 6.1.8

Entry Type	Name of the Tool
Schematic Entry	Composer Schematic
Symbol Creation	Composer Symbol
Testbench Creation	Composer Schematic
Simulation	Analog Design Environment
Layout	Virtuoso
DRC	Assura
LVS	Assura

Block Diagram

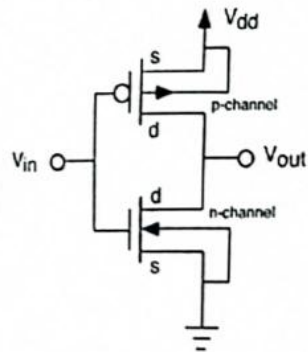


Truth table

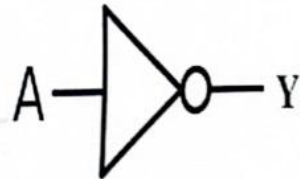
Vin	Vout
1	0
0	1

Boolean Expression: $V_{out} = \sim V_{in}$

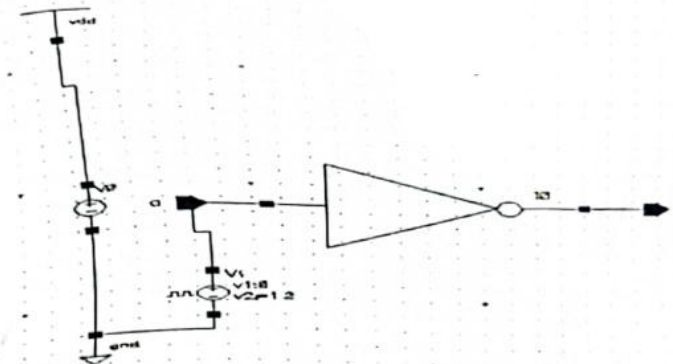
Schematic



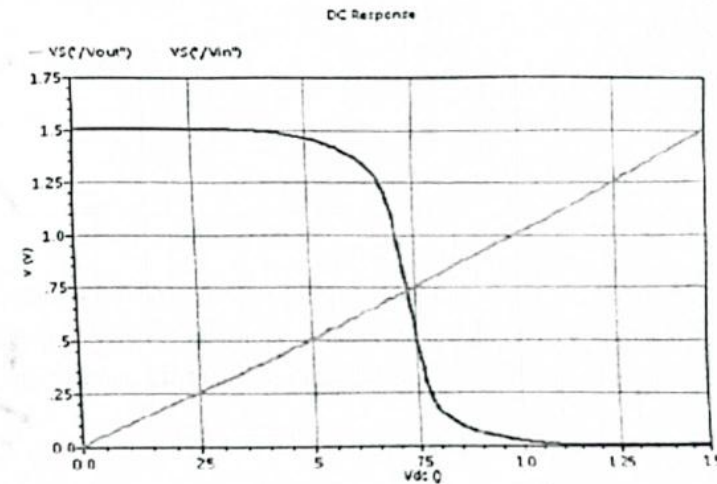
Symbol



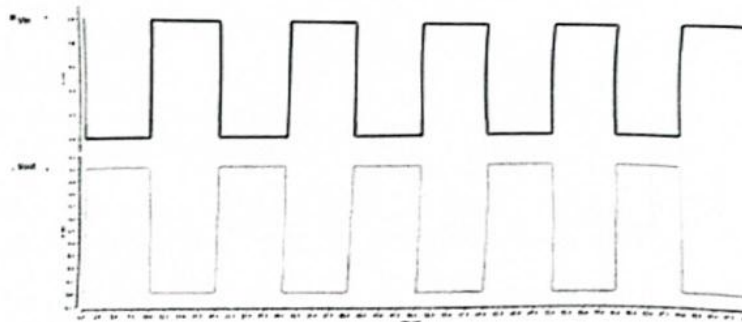
Testbench:



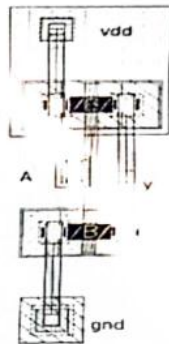
DC Analysis



Transient Analysis



Layout:



Result: The CMOS inverter designed using GPDK 90nm technology demonstrated proper switching behavior with a $V_{thV_{th}}$ of [value] V, propagation delay of [value] ps, dynamic power dissipation of [value] μ W, and successfully passed DRC and LVS checks.

Viva-voce Questions:

1. Define Noise Margin.
2. Explain CMOS Inverter DC characteristics.

Assignment:

1. Design NMOS depletion load inverter.
2. Design Pseudo NMOS inverter.
3. Perform I-V Characteristics of NMOS and PMOS transistor.

Design of CMOS NAND

Aim: To design a CMOS NAND gate using simulation and propagation delay through simulation and

Tools Required: Tools Required: Cadence 6.1.8

Entry Type	Name of the Tool
Schematic Entry	Composer Schematic
Symbol Creation	Composer Symbol
Testbench Creation	Composer Schematic
Simulation	Analog Design Environment

Block Diagram

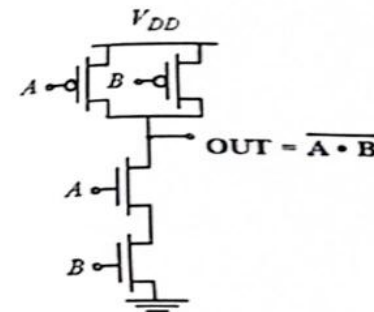


Truth table

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

Boolean Expression: $Y = \overline{A \cdot B}$

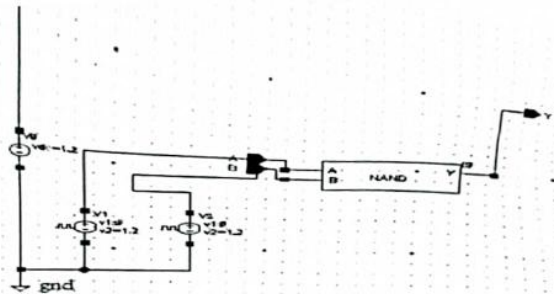
Schematic



2 input NAND gate (Static Logic)

Symbol





Result Analysis

The CMOS NAND gate was successfully designed using 90 nm CMOS technology and the area and propagation delay of the NAND gate were estimated through simulation.

Viva-voce Questions:

1. Draw the stick diagram of CMOS AND gate.
2. What is meant by Design Rule Checker?
3. Draw the stick diagram of CMOS OR gate.
4. What is meant by GPDK?

Assignment:

1. Design CMOS AND gate and perform its LVS.
2. Design CMOS OR gate and perform its LVS.

Experiment No: 03

Parasitic Extraction and Post-Layout Simulation of CMOS XOR Gate

Aim: To perform parasitic extraction and post-layout simulation of a CMOS XOR gate using 90 nm CMOS technology in order to analyze the effect of parasitic resistances and capacitances on circuit performance parameters such as delay and power consumption.

Tools Required: Cadence 6.1.8

Entry Type	Name of the Tool
Schematic Entry	Composer Schematic
Symbol Creation	Composer Symbol
Testbench Creation	Composer Schematic
Simulation	Analog Design Environment
Layout	Virtuoso
DRC	Assura
LVS	Assura
QRC	Assura

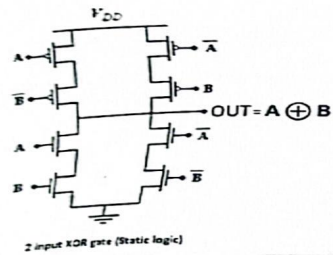
Block Diagram Truth table



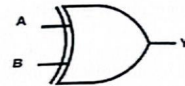
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

Boolean Expression: $Y = \overline{A}B + A\overline{B}$

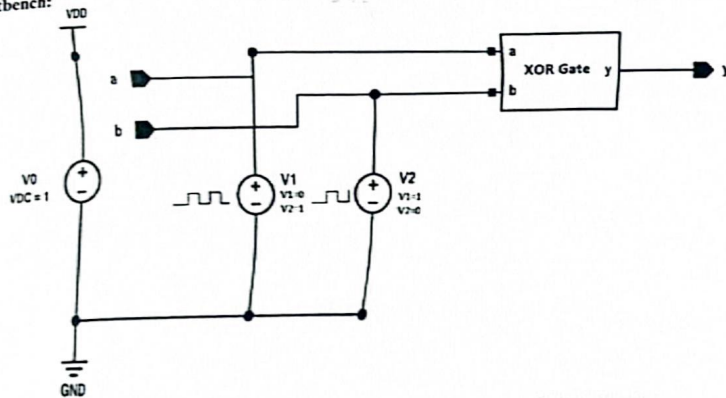
Schematic



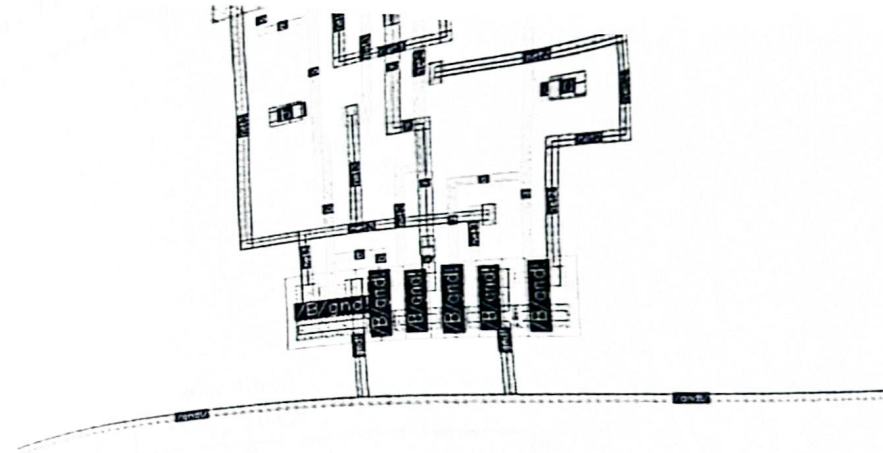
Symbol



Testbench:



Layout:



Result Analysis:- The CMOS XOR gate designed using GPDK 90nm technology exhibited correct logical behavior, with verified layouts passing DRC and LVS checks, and simulation results

Viva-voce Questions: -

- 1) What are the applications of parity generator?
- 2) What does no parity mean?
- 3) What is the parity value?
- 4) Why do we need a parity checker?
- 5) How can the parity bit detect a damaged data unit?

Assignment: -

1. Design 4-bit Parity generator.

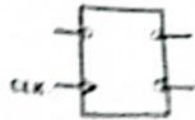
Experiment No: 04

Rule Check (DRC) and Layout vs. Schematic (LVS) for CMOS D Flip-Flop
 Aim: To design the layout of a CMOS D Flip-Flop and perform DRC and LVS verification to check for layout rule errors and schematic matching.

Tools Required: Tools Required: Cadence 6.1.2

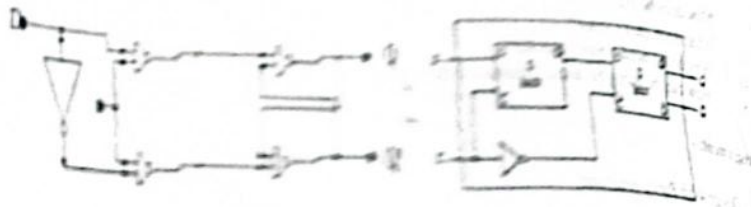
Entry Type	Name of the Tool
Schematic Entry	Cadence Schematic
Symbol Creation	Cadence Symbol
Testbench Creation	Cadence Testbench
Simulation	Verilog, VHDL, Behavioral
Layout	Layout
DRC	DRC
LVS	LVS

Block Diagram



Boolean Expression: $Q = D$

D-Latch Schematic



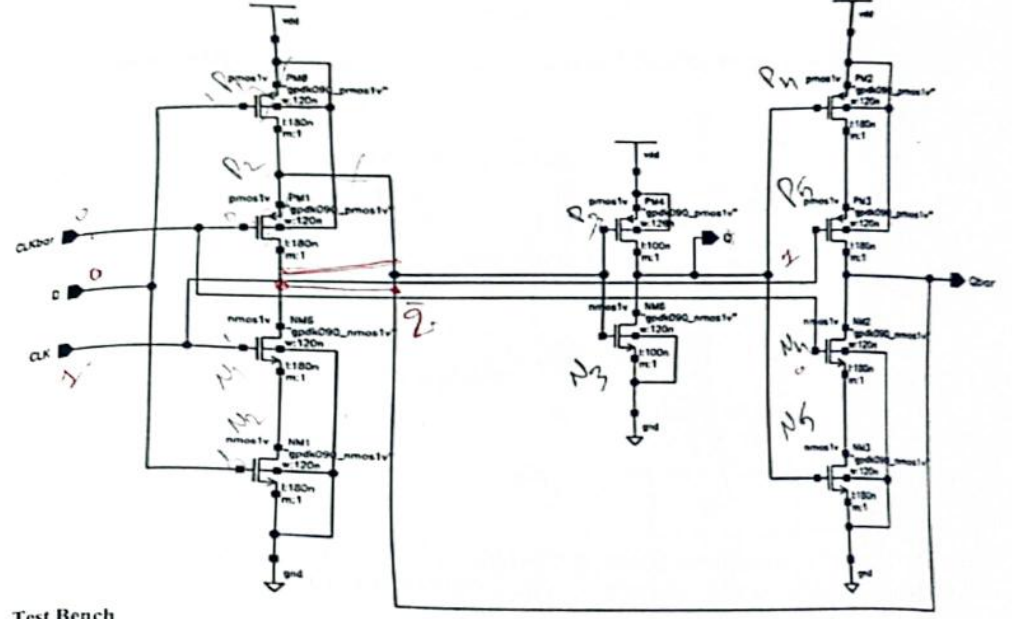
Truth table

CLK	D	Q	Q'
X	X	0	1
1	0	0	1
1	1	1	0

D Flip-Flop

Schematic

Copyright © CVR COLLEGE OF ENGINEERING



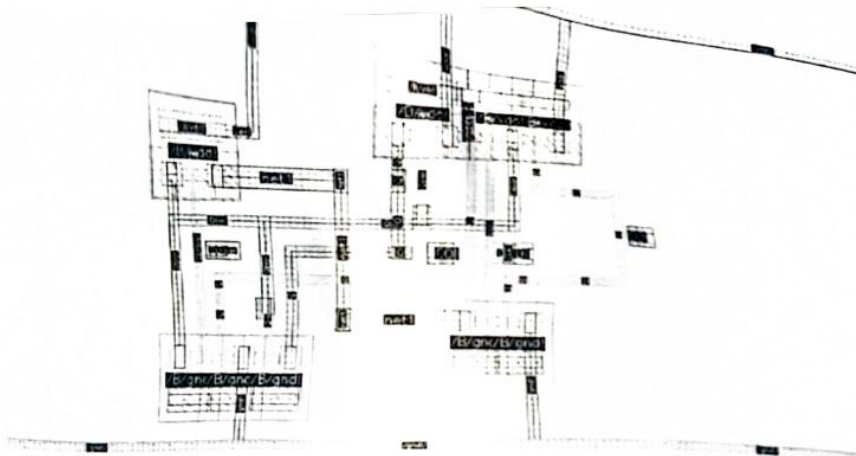
Experiment No: 04

Design Rule Check (DRC) and Layout vs. Schematic (LVS) for CMOS D Flip-Flop

Aim: To design the layout of a CMOS D Flip-Flop and perform DRC and LVS verification to check for layout rule errors and schematic matching.

Tools Required: Tools Required: Cadence 6.1.8

Entry Type	Name of the Tool
Schematic Entry	Composer Schematic
Symbol Creation	Composer Symbol
Testbench Creation	Composer Schematic
Simulation	Analog Design Environment
Layout	Virtuoso
DRC	Assura
LVS	Assura



Result Analysis: The layout of the CMOS D Flip-Flop was successfully designed, and the DRC and LVS verifications were completed without any errors.

Viva-voce Questions:

1. Draw the D-flipflop using nand gates.
2. Difference between flip-flop and latch.

Assignment:

Design and simulate 2-bit counter using D-Flip-flop

Experiment No: 05

Schematic design and simulation of CMOS 4:1 Multiplexer using 2:1 Muxes

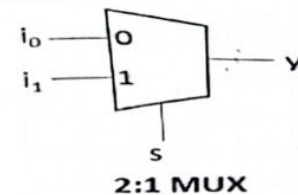
Aim: To design and simulate a CMOS 4:1 Multiplexer using 2:1 Multiplexers and verify its functionality through simulation.

Tools Required: Tools Required: Cadence 6.1.8

Entry Type	Name of the Tool
Schematic Entry	Composer Schematic
Symbol Creation	Composer Symbol
Testbench Creation	Composer Schematic
Simulation	Analog Design Environment

Block Diagram

2:1 Multiplexer

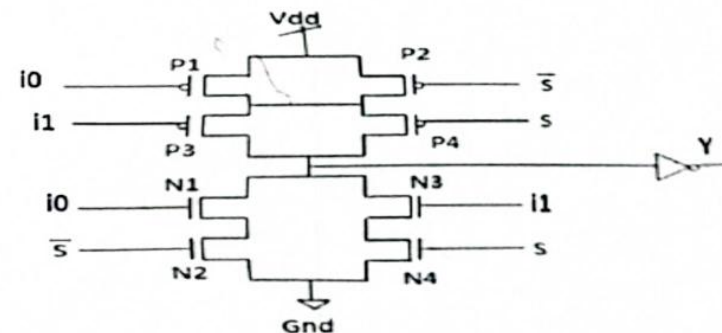


Truth table

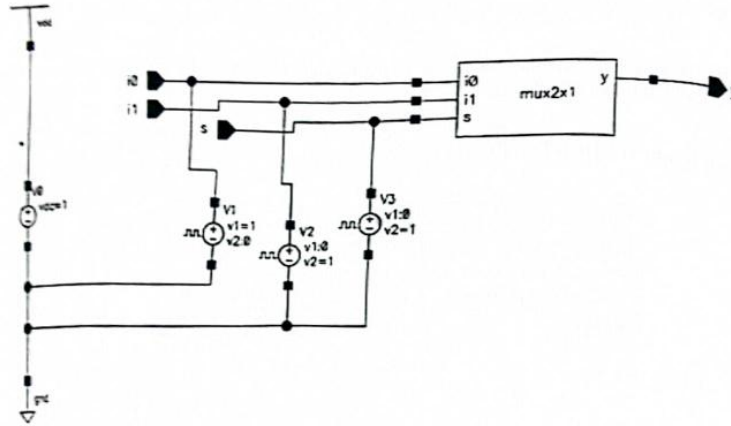
s	y
0	i ₀
1	i ₁

Boolean Expression: $Y = \bar{s}i_0 + si_1$

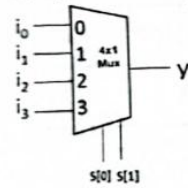
Schematic of 2:1 Multiplexer



Testbench



4:1 Multiplexer



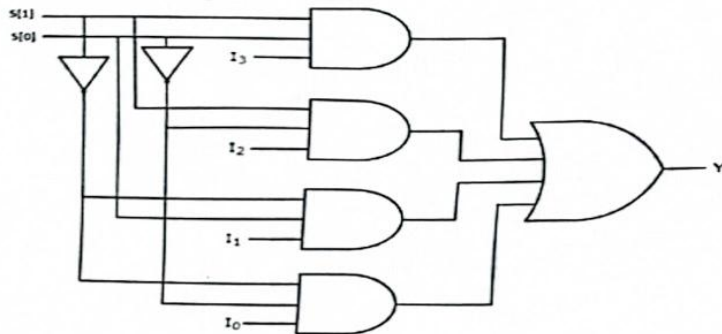
Truth table

s[0]	s[1]	Y
0	0	i ₀
0	1	i ₁
1	0	i ₂
1	1	i ₃

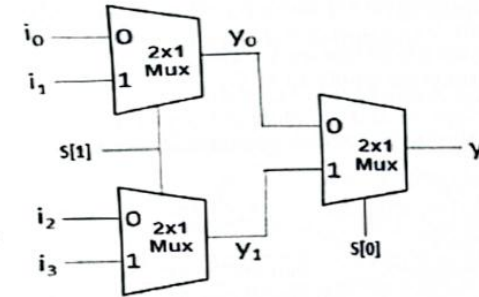
Boolean Expression:

$$Y = \overline{s_1} \overline{s_0} I_0 + \overline{s_1} s_0 I_1 + s_1 \overline{s_0} I_2 + s_1 s_0 I_3$$

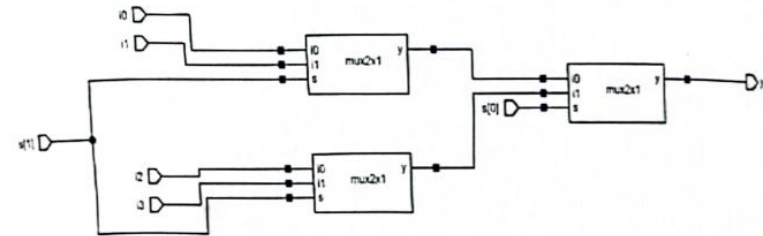
Gate Level Diagram



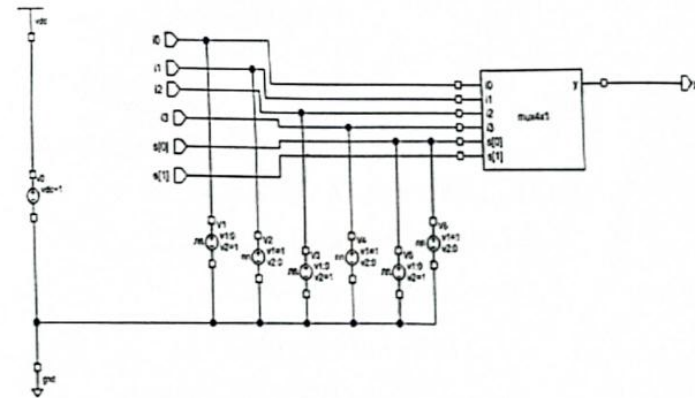
4:1 MUX using 2:1 MUXes



Schematic:



Testbench:



Result Analysis: The CMOS 4:1 Multiplexer was successfully designed using 2:1 Multiplexers and simulated. The simulation results verified the correct selection of inputs based on the select lines.

Viva-voce Questions:

1. What is the purpose of select lines in a MUX?
2. Derive the Boolean expression of a 4:1 MUX.

Assignment:

1. How can a 4:1 MUX be implemented using transmission gates?

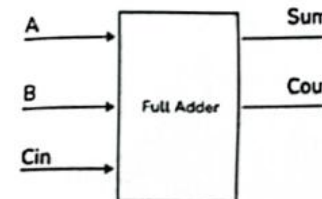
Design, DRC/LVS Verification, and Post-Layout Performance Analysis of a 1-Bit Full Adder Using CMOS Technology

Aim: To design and implement a 1-bit Full Adder using CMOS technology, perform Design Rule Check (DRC) and Layout Versus Schematic (LVS) verification to ensure layout correctness, and analyze the post-layout performance in terms of functionality, delay, power consumption, and area.

Tools Required: Tools Required: Cadence 6.1.8

Entry Type	Name of the Tool
Schematic Entry	Composer Schematic
Symbol Creation	Composer Symbol
Testbench Creation	Composer Schematic
Simulation	Analog Design Environment
Layout	Virtuoso
DRC	Assura
LVS	Assura
QRC	Assura

Block Diagram



Truth Table

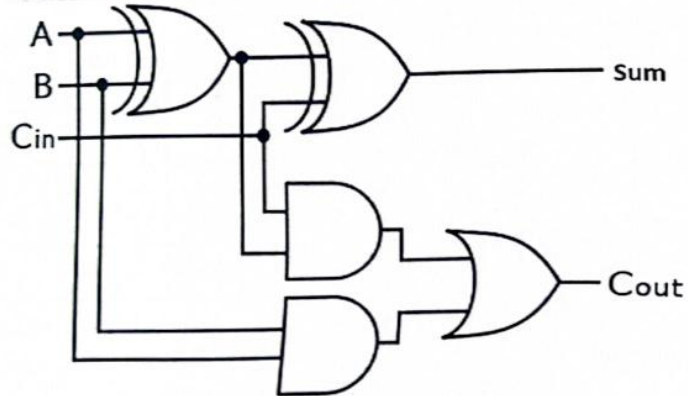
A	B	Cin	Sum	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Boolean Expressions

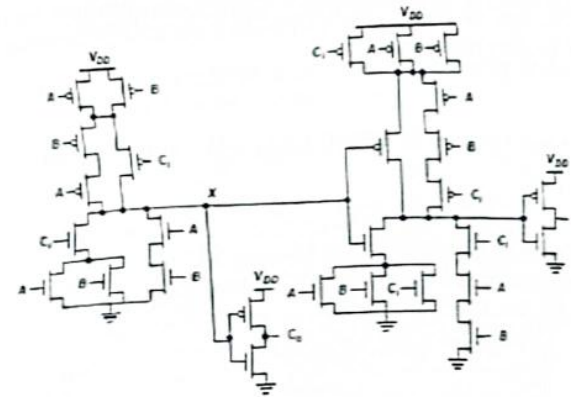
$$\begin{aligned}
 \text{Sum} &= \bar{A} \bar{B} C_{in} + \bar{A} B \bar{C}_{in} + A \bar{B} \bar{C}_{in} + A B C_{in} \\
 &= C_{in} (\bar{A} \bar{B} + AB) + \bar{C}_{in} (\bar{A} B + A \bar{B}) \\
 &= C_{in} (A \odot B) + \bar{C}_{in} (A \oplus B) \\
 &= C_{in} (\overline{A \oplus B}) + \bar{C}_{in} (A \oplus B) \\
 &= C_{in} \oplus (A \oplus B)
 \end{aligned}$$

$$\begin{aligned}
 C_{out} &= AB + C_{in} (A \bar{B} + \bar{A} B) \\
 &= AB + A \bar{B} C_{in} + \bar{A} B C_{in} \\
 &= AB (C_{in} + 1) + A \bar{B} C_{in} + \bar{A} B C_{in} \\
 &= AB C_{in} + AB + A \bar{B} C_{in} + \bar{A} B C_{in} \\
 &= AB + A C_{in} (B + \bar{B}) + \bar{A} B C_{in} \\
 &= AB + AC_{in} + \bar{A} B C_{in} \\
 &= AB (C_{in} + 1) + A C_{in} + \bar{A} B C_{in} \\
 &= ABC_{in} + AB + A C_{in} + \bar{A} B C_{in} \\
 &= AB + A C_{in} + BC_{in} (A + \bar{A}) \\
 &= AB + A C_{in} + BC_{in}
 \end{aligned}$$

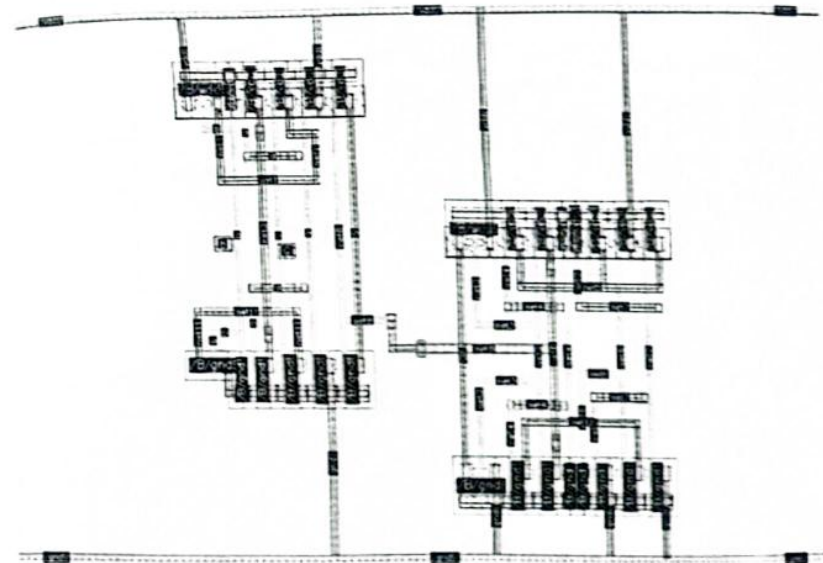
Gate Level Diagram



Schematic:



Layout



...igned using CMOS technology and its layout verification showed that the layout matched the schematic. Post-layout simulation verified the correct operation of the Full Adder, and its performance parameters such as propagation delay, power consumption, and area were successfully analyzed.

Viva-voce Questions:

1. Calculate the transistor count required for a CMOS Full Adder implementation..

Why is DRC performed?

Assignment:

1. Explain how Full Adders are used to construct ripple-carry adders.
2. Describe common layout design rules used in CMOS technology.

Experiment No: 07

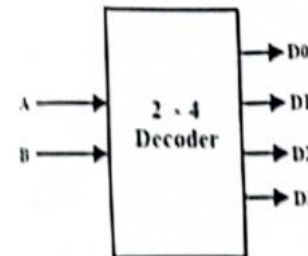
Design and Simulation of CMOS 2-to-4 Decoder

Aim: To design and simulate a CMOS 2-to-4 Decoder and verify its functionality by observing the output responses for all possible input combinations.

Tools Required: Tools Required: Cadence 6.1.7

Entry Type	Name of the Tool
Schematic Entry	Composer Schematic
Symbol Creation	Composer Symbol
Testbench Creation	Composer Schematic
Simulation	Analog Design Environment
Layout	Virtuoso
DRC	Assura
LVS	Assura

Block Diagram



Truth Table

A ₁	A ₀	D ₃	D ₂	D ₁	D ₀
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

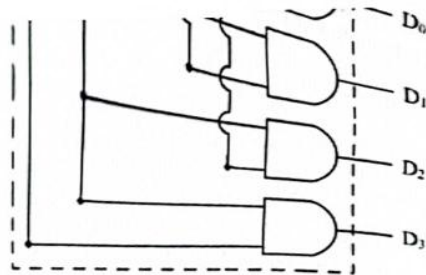
Boolean Expressions

$$D_0 = \bar{A}_1 \cdot \bar{A}_0$$

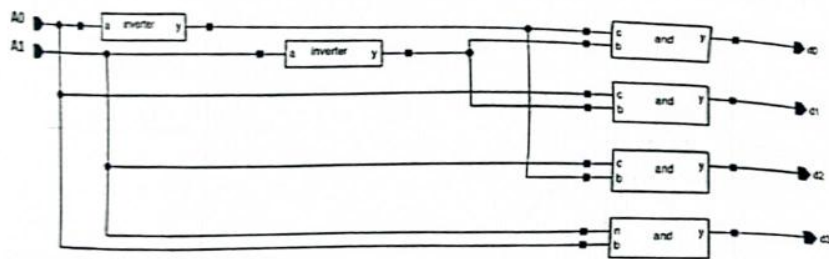
$$D_1 = \bar{A}_1 \cdot A_0$$

$$D_2 = A_1 \cdot \bar{A}_0$$

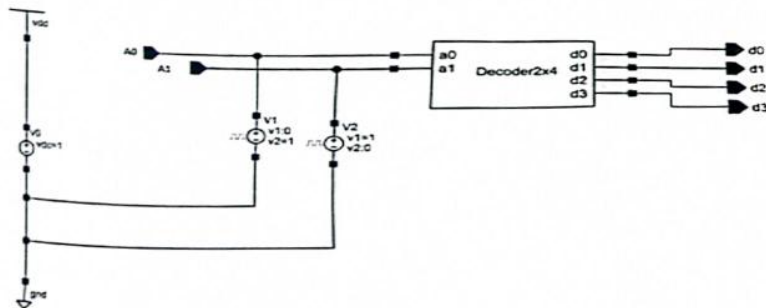
$$D_3 = A_1 \cdot A_0$$



Schematic:



Testbench:



Result Analysis: The CMOS 2-to-4 Decoder was successfully designed and simulated. The simulation results verified that the correct output line was activated for each input combination, confirming the proper operation of the decoder.

Viva-voce Questions:

1. What is a decoder in digital electronics?
2. Compare a decoder and an encoder.

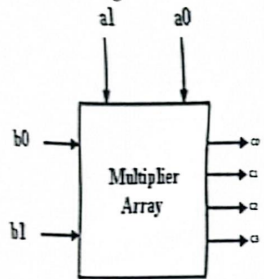
Assignment:

1. Design a CMOS implementation of a 2-to-4 decoder.
2. Design 3 to 8 decoder using 2 to 4 decoder

...y its functionality by

Schematic Entry	Name of the Tool
Symbol Creation	Composer Schematic
Testbench Creation	Composer Symbol
Simulation	Composer Schematic
Layout	Analog Design Environment
DRC	Virtuoso
LVS	Assura

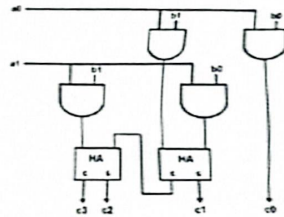
Block Diagram



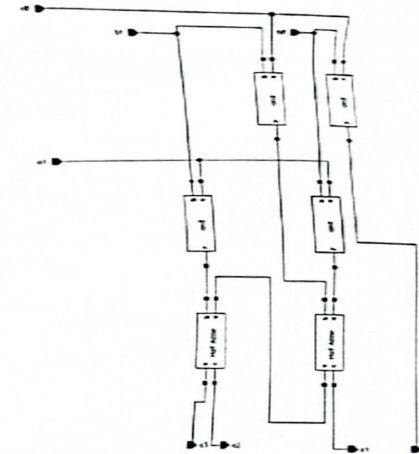
2bit array multiplication

$$\begin{array}{r}
 \begin{array}{cc}
 b1 & b0 \\
 a1 & a0 \\
 \hline
 a0b1 & a0b0 \\
 \hline
 a1b1 & a1b0 \\
 \hline
 c3 & c2 & c1 & c0
 \end{array}
 \end{array}$$

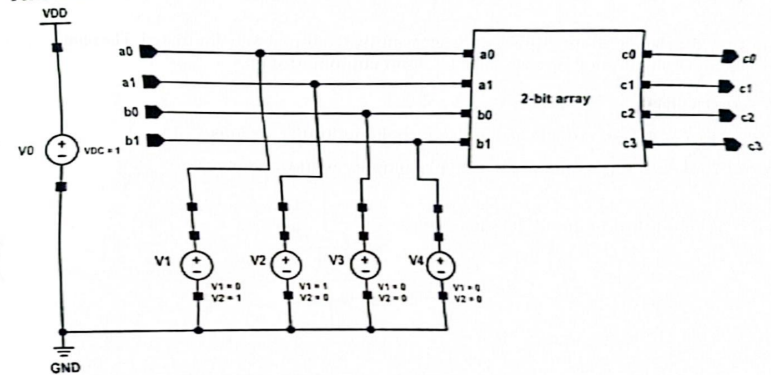
Architecture



Schematic:



Testbench:

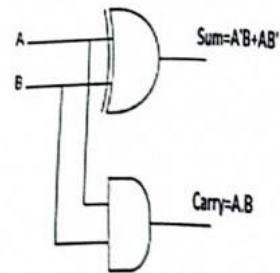


Half adder:

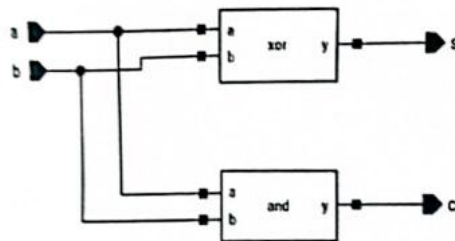
Truth Table

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Gate Level Diagram:



Schematic:



Result Analysis: The CMOS multiplier was successfully designed and simulated. The output waveforms verified correct multiplication operation for all input combinations.

Viva-voce Questions:

1. Explain the role of Half Adders and Full Adders in multiplier circuits.
2. Explain the simulation results of the CMOS multiplier with waveforms.

Assignment:

1. Design 4 bit multiplier using CMOS logic gates.

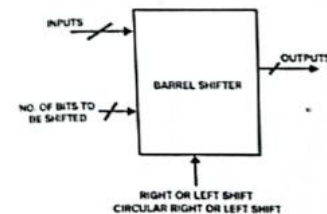
Experiment No: 9 Design and simulation of Shifters

Aim: To design and simulate a CMOS Barrel Shifter and verify its ability to shift binary data left or right by a specified number of bit positions using control signals.

Tools Required: Cadence 6.1.8

Entry Type	Name of the Tool
Schematic Entry	Composer Schematic
Symbol Creation	Composer Symbol
Testbench Creation	Composer Schematic
Simulation	Analog Design Environment

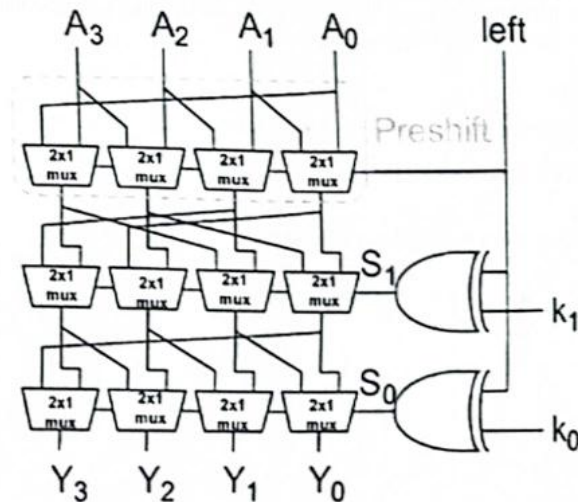
Block Diagram:

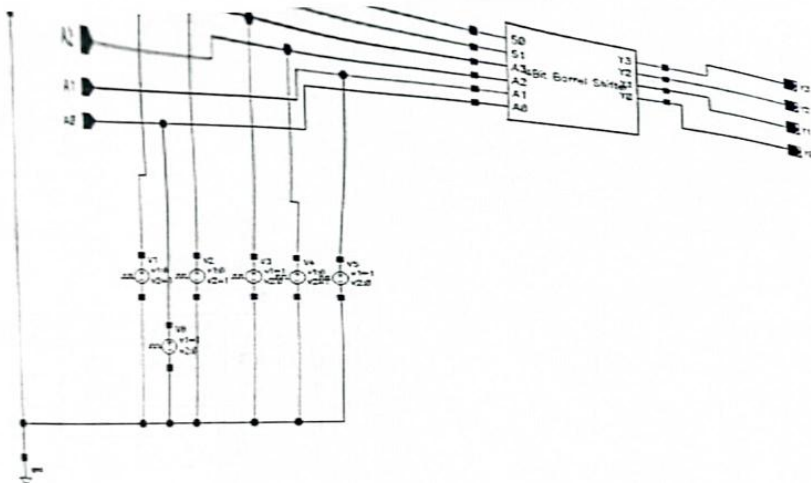


Truth table:

S1	S0	Y3	Y2	Y1	Y0	RL	RR
0	0	A3	A2	A1	A0	-	-
0	1	A2	A1	A0	A3	1	3
1	0	A1	A0	A3	A2	2	2
1	1	A0	A3	A2	A1	3	1

Schematic:





Result Analysis: The CMOS Barrel Shifter was successfully designed and simulated. The circuit correctly shifted the input binary data left or right according to the control signals, and the expected output was verified for different shift operations.

Viva-voce Questions:

1. What are the different types of shift operations?
2. Compare a barrel shifter with a shift register.

ASSIGNMENT:

1. Design a bidirectional barrel shifter and explain its working.

Experiment No: 10

Design of Fin-FET NAND and NOR gates

Aim: To design and simulate FinFET-based NAND and NOR gates and verify their logic operation and performance.

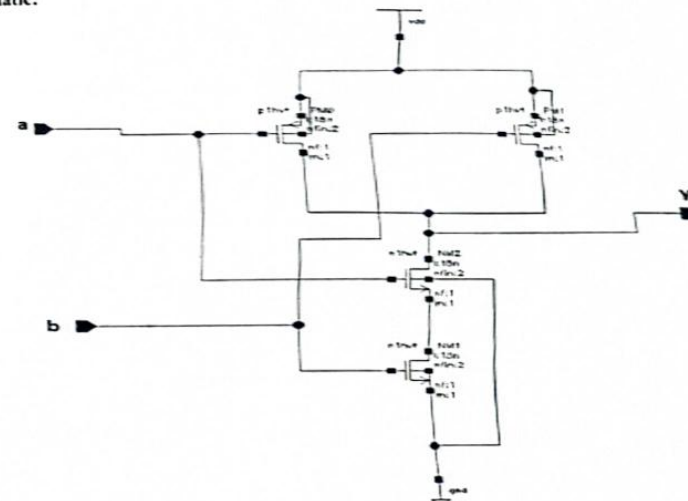
Tools Required: Cadence 6.1.8

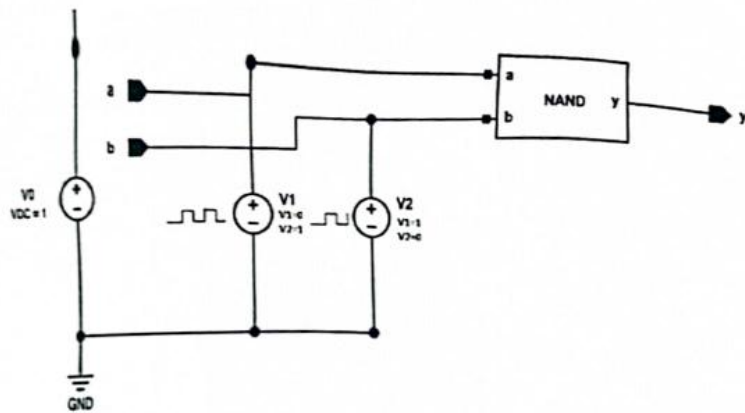
Entry Type	Name of the Tool
Schematic Entry	Composer Schematic
Symbol Creation	Composer Symbol
Testbench Creation	Composer Schematic
Simulation	Analog Design Environment

NAND Gate:
Truth table:

a	b	$y = \overline{ab}$
0	0	1
0	1	1
1	0	1
1	1	0

Schematic:

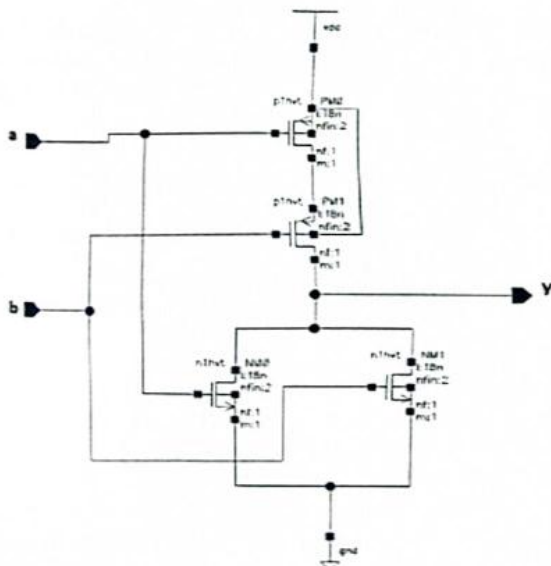




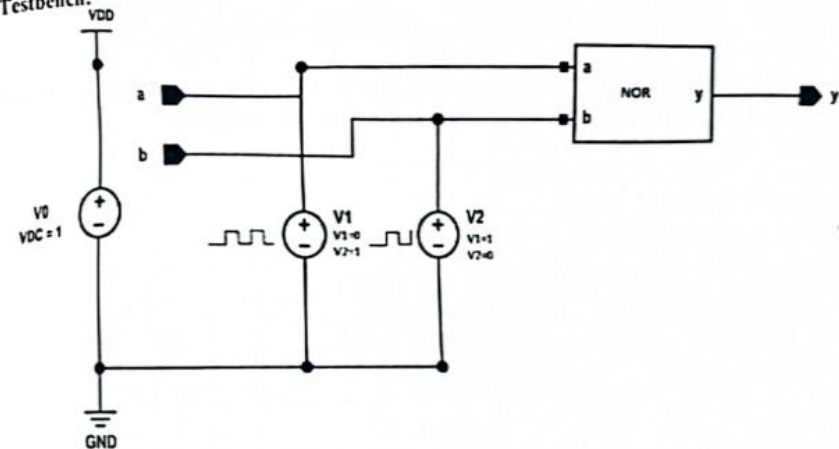
NOR Gate:
Truth table:

a	b	$y = \overline{a + b}$
0	0	1
0	1	0
1	0	0
1	1	0

Schematic:



Testbench:



Result Analysis: The FinFET-based NAND and NOR gates were successfully designed and simulated. The circuits produced the correct output for all input combinations, verifying their logic functionality. The simulation also demonstrated efficient operation with improved performance and reduced power consumption compared to conventional CMOS designs.

Viva-voce Questions:

1. How does a FinFET differ from a conventional MOSFET?
2. Why are NAND and NOR gates called universal gates?

Assignment:

1. Draw the schematic of a 2-input FinFET XOR gate and explain its operation.

Experiment No: 11

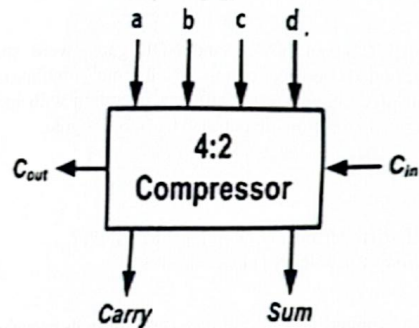
Design of 4:2 Compressors using Fin-FET

Aim: To design and simulate a 4:2 Compressor using FinFET technology and verify its functionality, speed, and power efficiency for high-performance arithmetic applications.

Tools Required: Cadence 6.1.8

Entry Type	Name of the Tool
Schematic Entry	Composer Schematic
Symbol Creation	Composer Symbol
Testbench Creation	Composer Schematic
Simulation	Analog Design Environment

Block Diagram:



Boolean Expressions:

$$\text{sum} = a \oplus b \oplus c \oplus d \oplus \text{cin}$$

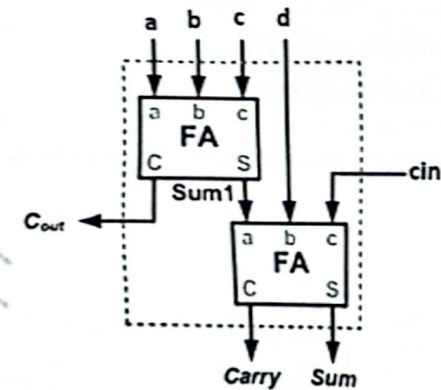
$$\text{carry} = (a \oplus b \oplus c)d + (a \oplus b \oplus c)\text{cin} + d\text{cin}$$

$$\text{cout} = ab + bc + ca$$

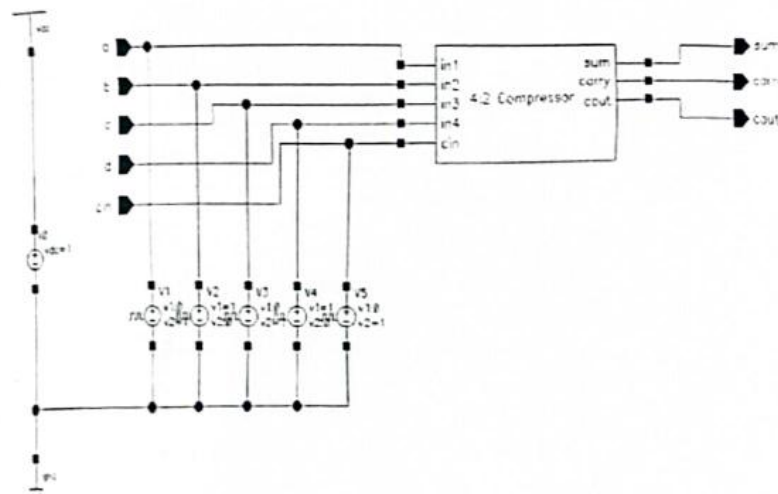
Truth table:

cin	a	b	c	d	Carry	Sum
0	0	0	0	0	0	0
0	0	0	0	1	0	1
0	0	0	1	0	0	1
0	0	0	1	1	0	0
0	0	1	0	0	0	1
0	0	1	0	1	0	0
0	0	1	1	0	1	0
0	0	1	1	1	1	1
0	1	0	0	0	0	1
0	1	0	0	1	0	0
0	1	0	1	0	1	0
0	1	0	1	1	1	1
0	1	1	0	0	1	0
0	1	1	0	1	1	1
0	1	1	1	0	1	0
0	1	1	1	1	1	1
1	0	0	0	0	0	1
1	0	0	0	1	0	0
1	0	0	1	0	0	0
1	0	0	1	1	0	1
1	0	1	0	0	0	0
1	0	1	0	1	0	1
1	0	1	1	0	1	1
1	0	1	1	1	1	0
1	1	0	0	0	1	0
1	1	0	0	1	1	1
1	1	0	1	0	1	0
1	1	0	1	1	1	1
1	1	1	0	0	1	1
1	1	1	0	1	1	0
1	1	1	1	0	1	1
1	1	1	1	1	1	0

Schematic:



Testbench:



Result Analysis: The 4:2 Compressor using FinFET technology was successfully designed and simulated. The circuit correctly compressed four input bits and one carry-in into a sum and carry output, verifying its functional correctness.

Viva-voce Questions:

1. What is the main application of a 4:2 Compressor?
2. What is the significance of compressors in multipliers?

Assignment:

1. Compare the performance of a FinFET-based 4:2 Compressor with a CMOS-based implementation.

Experiment No: 12

Layout of the Fin-FET based Inverter and NAND gate

Aim: To design the layout of FinFET-based Inverter and NAND gates and verify their correctness through layout design rules and functionality analysis.

Tools Required: Cadence 6.1.8

Entry Type	Name of the Tool
Schematic Entry	Composer Schematic
Symbol Creation	Composer Symbol
Testbench Creation	Composer Schematic
Simulation	Analog Design Environment
Layout	Virtuoso
DRC	Assura
LVS	Assura

Inverter:

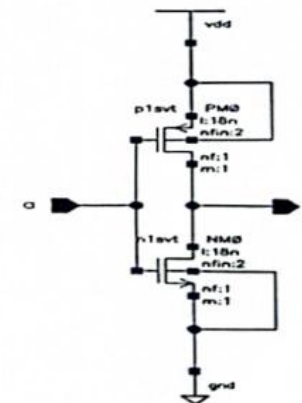
Logic diagram:



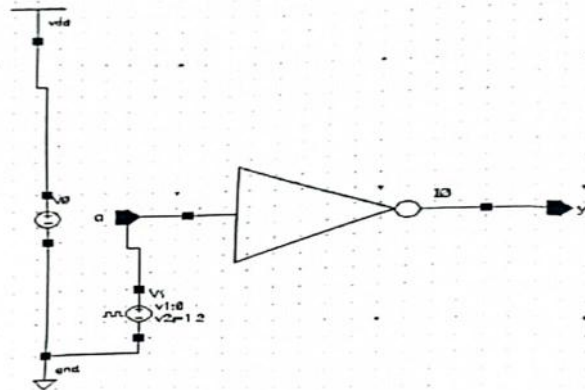
Truth table

a	y
1	0
0	1

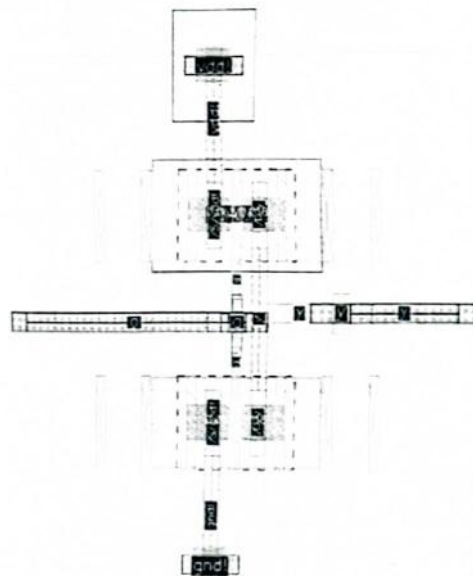
Schematic:



Testbench:

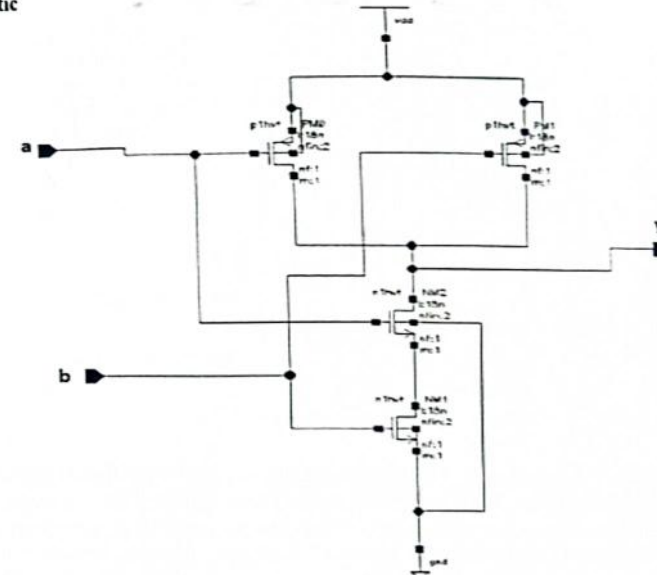


Layout:

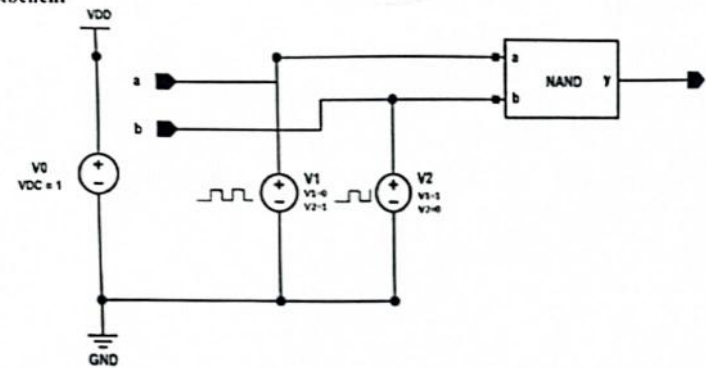
NAND gate:
Truth table:

a	b	$y = \overline{ab}$
0	0	1
0	1	1
1	0	1
1	1	0

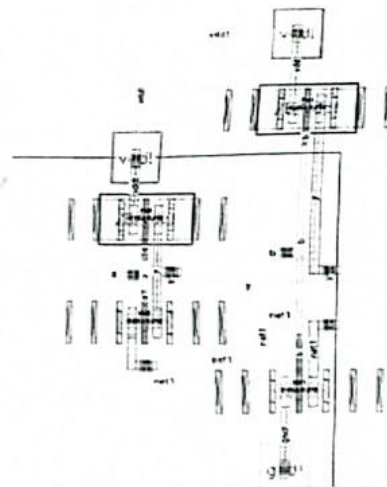
Schematic



Testbench:



Layout:



Result Analysis: The layouts of the FinFET-based Inverter and FinFET-based NAND gate were successfully designed and verified. Design Rule Check (DRC) confirmed that the layouts were free from rule violations. Functional analysis showed that the inverter produced the correct complemented output, and the NAND gate generated the expected NAND logic output for all input combinations. The results validate the correct operation and layout implementation of the FinFET-based logic gates.

Viva-voce Questions:

1. What is the purpose of layout verification?
2. What are the advantages of FinFET technology?
3. Explain the structure and working principle of a FinFET transistor.

Assignment:

1. Draw the layout of a 2-input FinFET XNOR gate and explain its operation.

Experiment No: 13

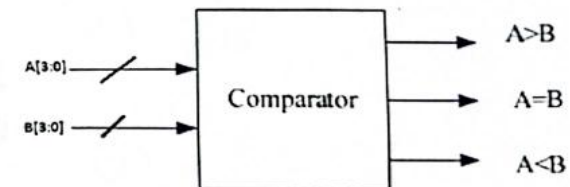
Design and simulation of Fin-FET-based digital magnitude comparator

Aim: To design and simulate a FinFET-based digital magnitude comparator and verify its ability to compare two binary inputs and determine whether one input is greater than, less than, or equal to the other.

Tools Required: Cadence 6.1.7

Entry Type	Name of the Tool
Schematic Entry	Composer Schematic
Symbol Creation	Composer Symbol
Testbench Creation	Composer Schematic
Simulation	Analog Design Environment

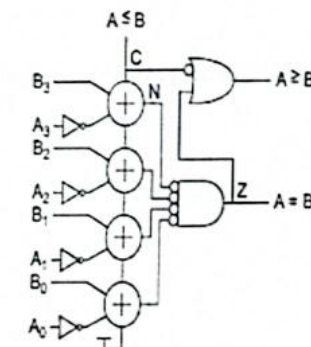
Block Diagram:



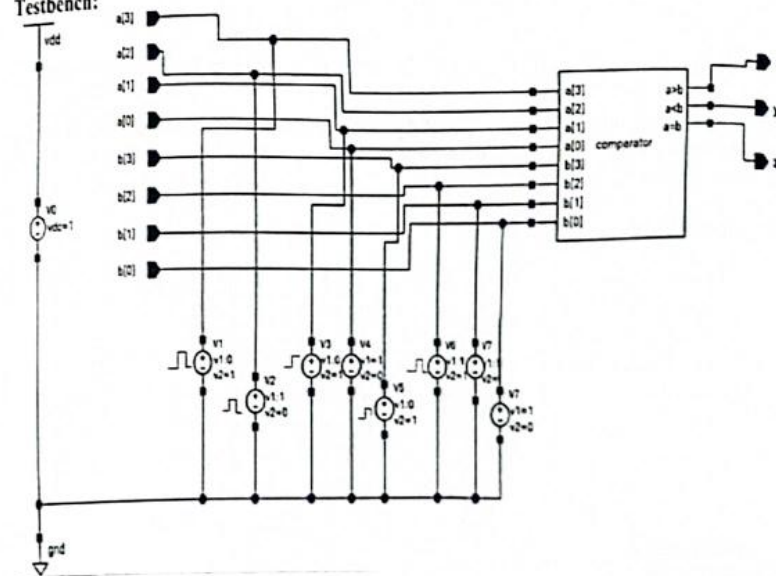
Truth table:

A[3]	A[2]	A[1]	A[0]	B[3]	B[2]	B[1]	B[0]	A > B	A < B	A = B
1	0	0	0	0	1	0	1	1	0	0
0	0	1	0	1	1	0	0	0	1	0
0	1	1	0	0	1	1	0	0	0	1

Schematic:



Testbench:



Result Analysis: The FinFET-based digital magnitude comparator was successfully designed and simulated. The circuit correctly compared two binary input values and generated the appropriate outputs indicating whether $A > B$, $A < B$, or $A = B$.

Viva-voce Questions:

1. Is a magnitude comparator a combinational or sequential circuit?
2. Explain how a FinFET-based comparator reduces power consumption.
3. Discuss the applications of digital magnitude comparators in digital systems.

Assignment:

1. Design a 8-bit digital magnitude comparator and verify its truth table.

Experiment No: 14

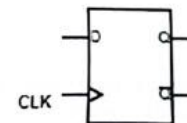
Simulation of Fin-FET-Based D Flip-Flop

Aim: To design and simulate a FinFET-based D Flip-Flop and verify its functionality by observing the storage and transfer of input data to the output on the active edge of the clock signal.

Tools Required: Cadence 6.1.7

Entry Type	Name of the Tool
Schematic Entry	Composer Schematic
Symbol Creation	Composer Symbol
Testbench Creation	Composer Schematic
Simulation	Analog Design Environment

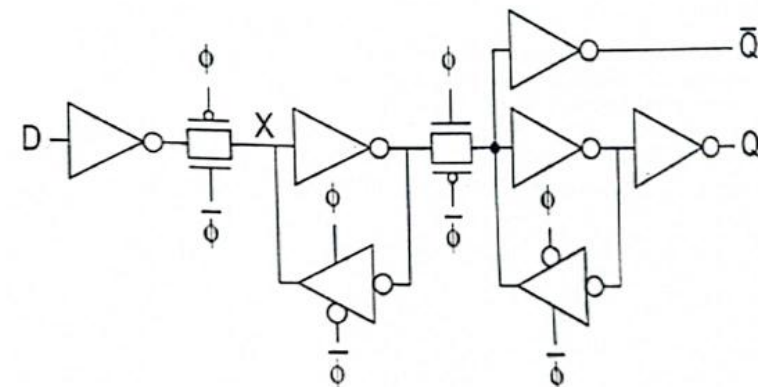
Block Diagram



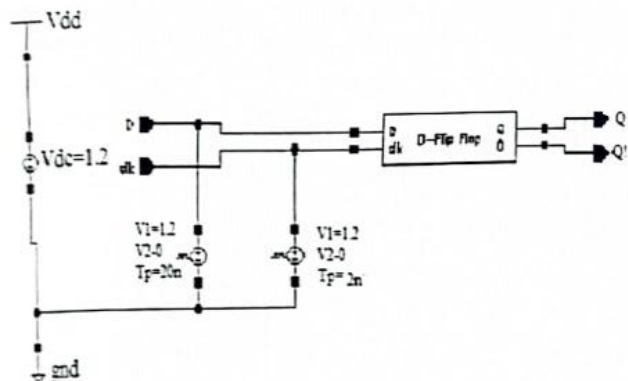
Truth table

CLK	D	Q	Q'
X	X	0	1
↑	0	0	1
↑	1	1	0

Schematic:



Testbench:



Result Analysis: The FinFET-based D Flip-Flop was successfully designed and simulated. The simulation results verified that the flip-flop correctly stored and transferred the input data (D) to the output (Q) on the active edge of the clock signal while maintaining the previous state during inactive clock periods.

Viva-voce Questions:

1. What is the characteristic equation of a D Flip-Flop?
2. What is the difference between a latch and a flip-flop?
3. What is meant by edge-triggering?

ASSIGNMENT:

1. Design a 4 bit binary counter using FinFET-based D Flip-Flop

Experiment No: 15

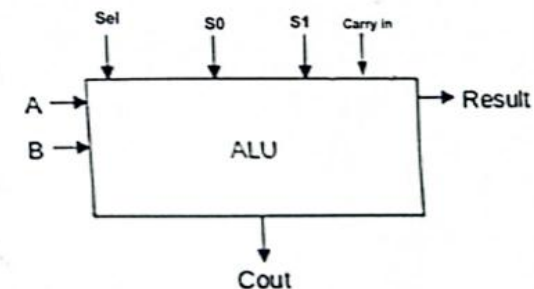
Implementation and Power-Delay Evaluation of Fin-FET-Based Arithmetic Logic Unit (ALU)

Aim: To design, implement, and simulate a FinFET-based Arithmetic Logic Unit (ALU) and evaluate its performance in terms of power consumption and delay while verifying the correct execution of arithmetic and logical operations using FinFET technology.

Tools Required: Cadence 6.1.8

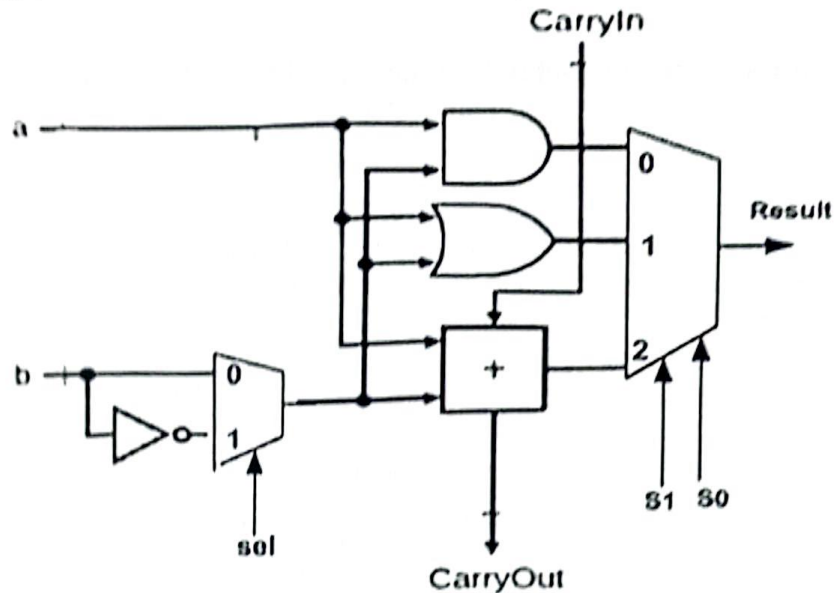
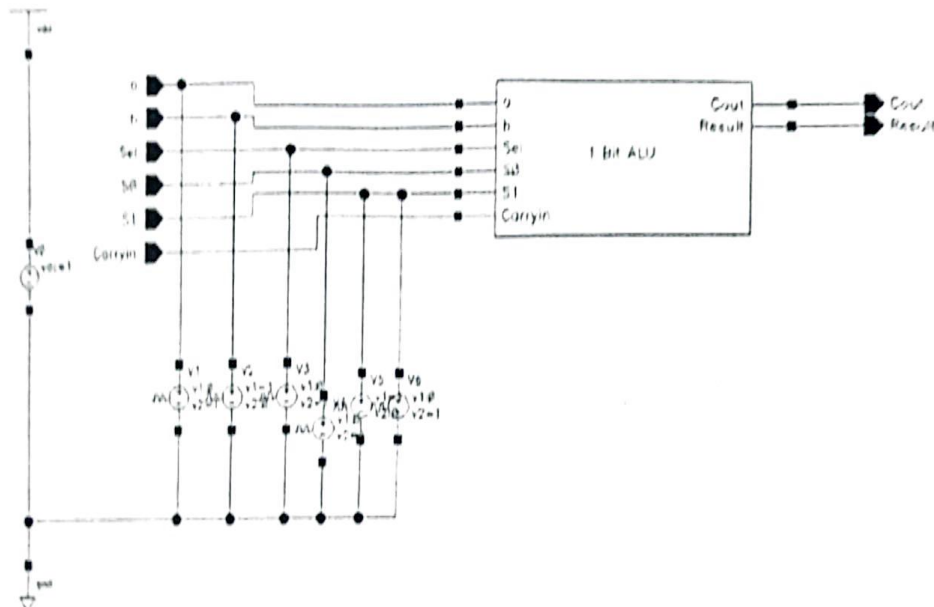
Entry Type	Name of the Tool
Schematic Entry	Composer Schematic
Symbol Creation	Composer Symbol
Testbench Creation	Composer Schematic
Simulation	Analog Design Environment
Layout	Virtuoso
DRC	Assura
LVS	Assura

Block Diagram:



Truth table:

Sel	Operation	Function
0	0	AND
0	1	OR
0	2	ADD
1	2	SUBTRACT
1	0	A AND B'
1	1	A OR B'

Logic Diagram:**Testbench:**

Result Analysis: The FinFET-based Arithmetic Logic Unit (ALU) was successfully designed, implemented, and simulated. The ALU correctly performed the required arithmetic operations (such as addition and subtraction) and logical operations (such as AND and OR) according to the applied control signals.

Viva-voce Questions:

1. What are the basic operations performed by an ALU?
2. What is the function of control signals in an ALU?
3. What is meant by edge-triggering?

Assignment:

1. Compare FinFET and CMOS technologies for ALU implementation.
2. Discuss the applications of ALUs in microprocessors, DSPs, and embedded systems.

