VLSI Testing and Fabrication
Course Project
Design of 3-bit Ripple Carry Adder in Cadence Virtuoso

By- Aadi Juvekar 211010201
& Anmol Agrawal 211010215

AIM: To make a 3-bit Ripple Carry Adder in Cadence Virtuoso using gpdk90 technology.

Tools Used: Cadence Virtuoso.

We construct the 3-bit Ripple Carry Adder in the following order:

1. Construction of NAND gate using CMOS technology.
2. Construction of XOR gate using the previously made NAND gate.
3. Construction of Full Adder using NAND & XOR gates.
4. Construction of a 3-bit Ripple Carry Adder using the previously constructed Full Adder.

1. Construction of NAND GATE

THEORY:

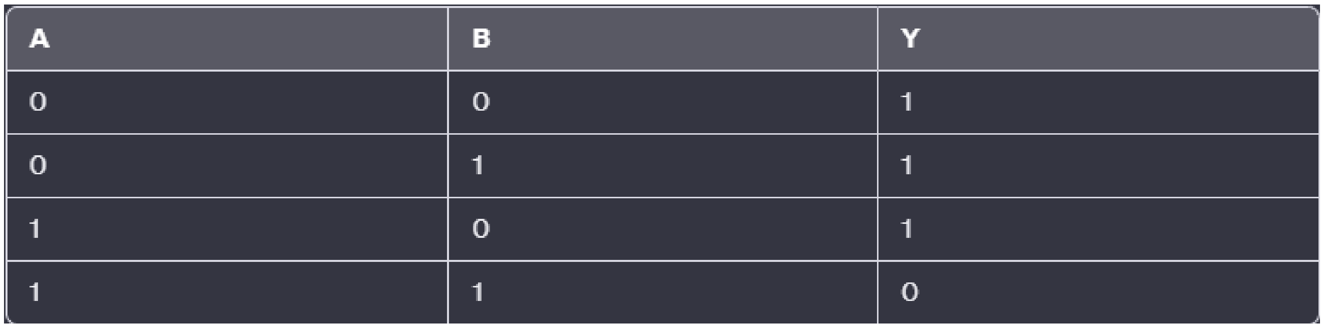
A NAND gate, short for NOT-AND gate, is a fundamental digital logic gate that performs the logical AND operation followed by a logical NOT operation. It is a building block in digital circuit design and is often used to create other logic gates. The basic theory of a NAND gate can be summarized as follows:

1. Symbol and Representation:

   The symbol for a NAND gate is a triangle with a curved arrow, representing the logical AND operation, followed by a small circle, indicating the logical NOT operation.

2. Truth Table:

   The truth table for a NAND gate has two inputs (A and B) and one output (Y). The output (Y) is 0 (LOW) only when both inputs (A and B) are 1 (HIGH). Otherwise, the output is 1 (HIGH).



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

3. Boolean Expression:

   The Boolean expression for the output (Y) of a NAND gate is *Y=(A*B)'* where *A'* represents the NOT of *A*, and (*A*B)* represents the logical AND of A and B.

4. Functional Description:

   A NAND gate produces a HIGH output only when both inputs are LOW. In all other cases, the output is LOW. It is effectively an inverted AND gate.

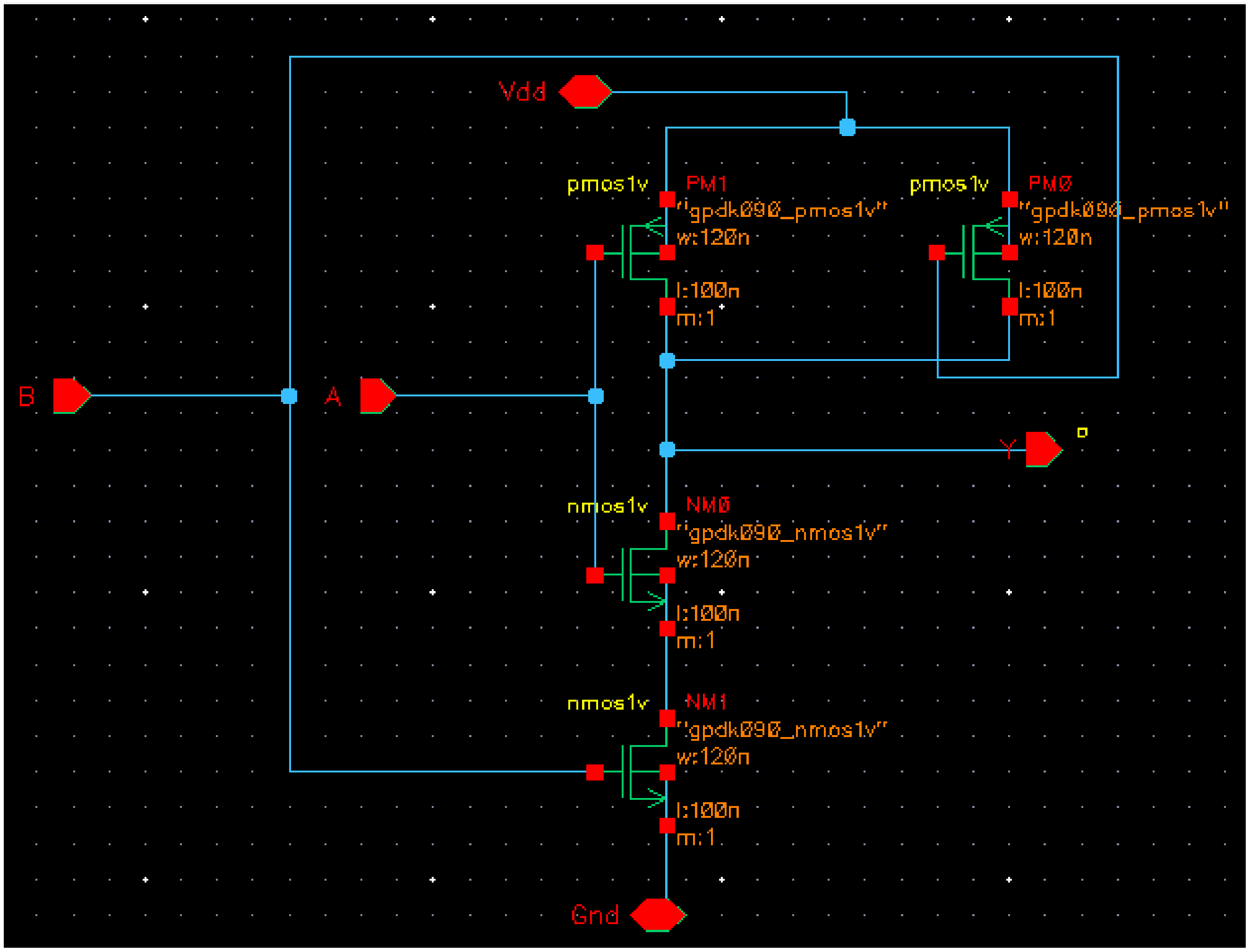
PROCEDURE:

SCHEMATIC:

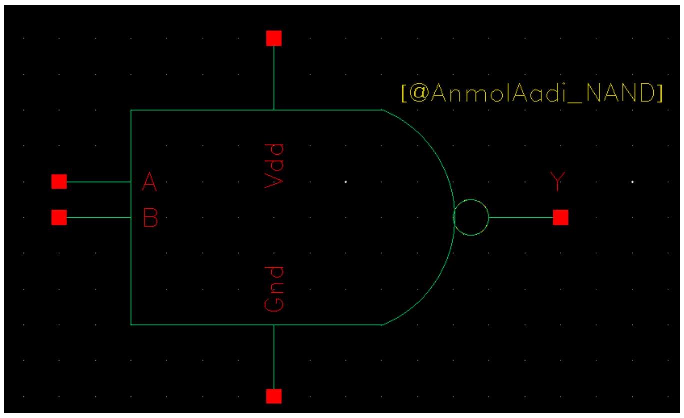
1. Setting Up the Cadence Virtuoso Environment:
   a. Open Cadence Virtuoso and create a new library for your project.
   b. Set up the library to use the gpdk90 technology file.

2. Schematic Design:
   a. Create a new schematic cell view within your library.
   b. Add a PMOS (pMOS) and an NMOS (nMOS) transistor from the gpdk90 library.

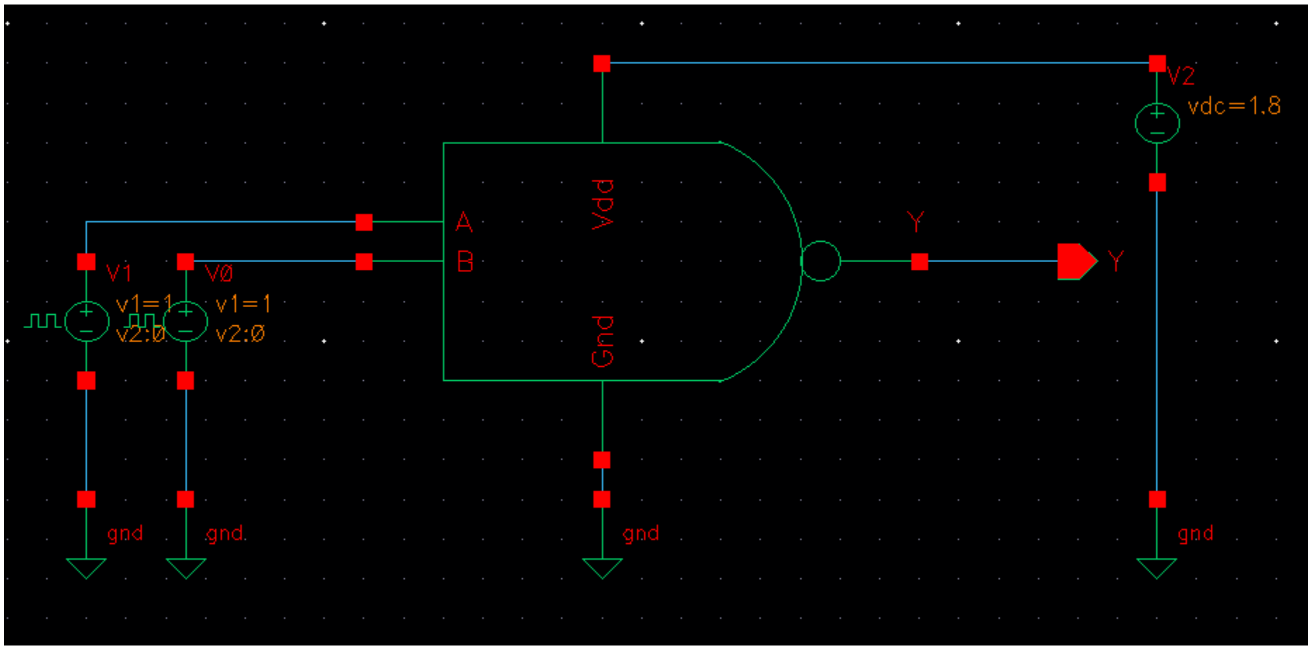
c. Connect the transistors to form a NAND gate according to the logic diagram.
d. Add power supply (VDD and GND) connections.
e. Use the appropriate transistor sizes based on the gpdk90 technology specifications.



3. Symbol Creation:
a. Create a symbol view for the NAND gate.
b. Assign pins to the symbol for input (A and B) and output (Y).
c. Associate the pins with the corresponding nodes in the schematic.

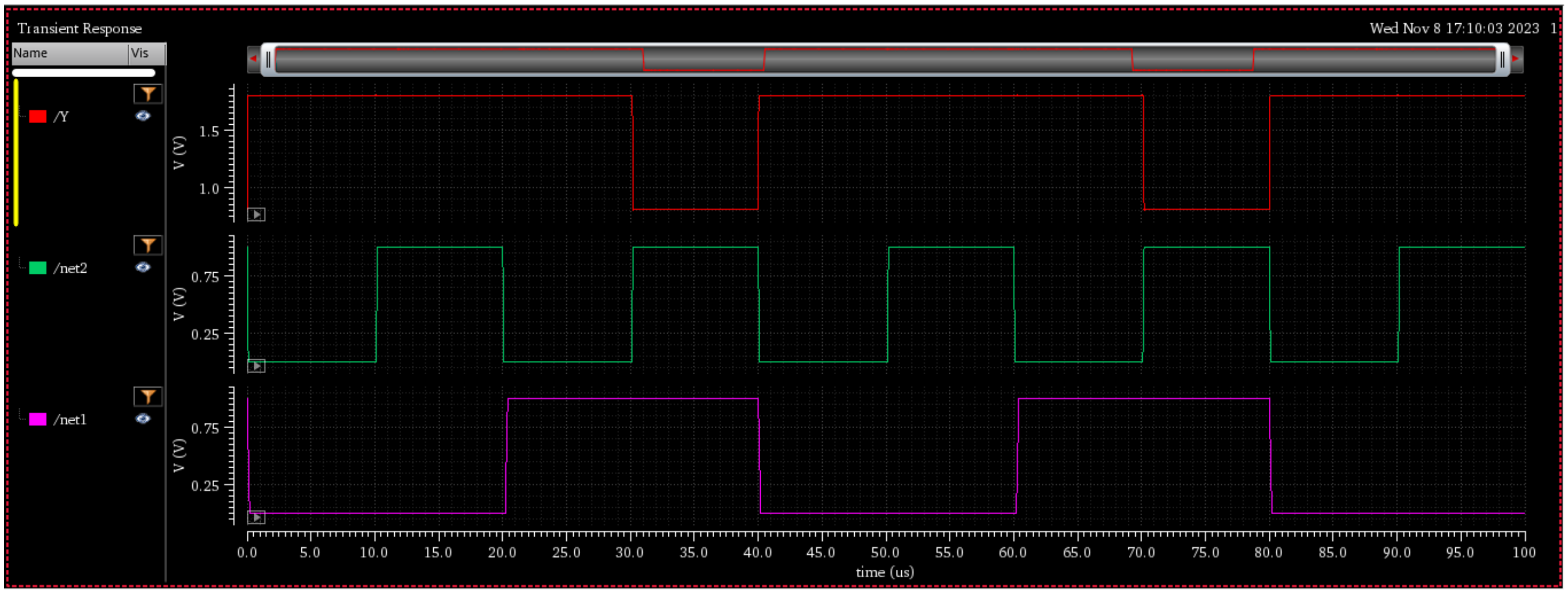


4. Simulation Setup:
    a. Create a new simulation cell view.
    b. Add a test bench with a voltage pulse (vpulse) source for input testing.
    c. Connect the vpulse source to the input pins of the NAND gate.
    d. Add a voltage probe to monitor the output.



5. Simulation:
    a. Run a transient simulation to observe the behavior of the NAND gate.
    b. Adjust simulation settings (time duration, time step) as needed.

c. Verify that the output responds correctly to changes in input.



LAYOUT:

1. Open the Schematic Window:
   a. Open Cadence Virtuoso and load your existing project.
   b. Open the schematic of your NAND gate.

2. Create a New Layout Cell View:
   a. In the schematic window, select the "Launch Virtuoso Layout" option to open the layout environment.
   b. Create a new layout cell view for your NAND gate.

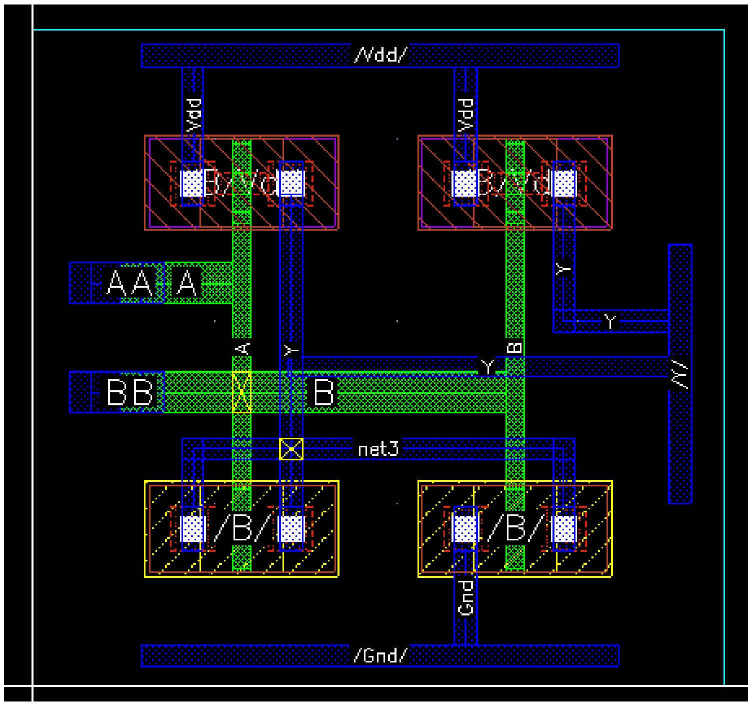
3. Place Transistors:
   a. Manually arrange and place the nMOS and pMOS transistors on the layout according to your schematic. Ensure that the transistors are oriented correctly (source, drain, and gate connections).

4. Connect Transistors:
   a. Use metal layers to connect the source and drain terminals of the transistors, creating the desired connections.

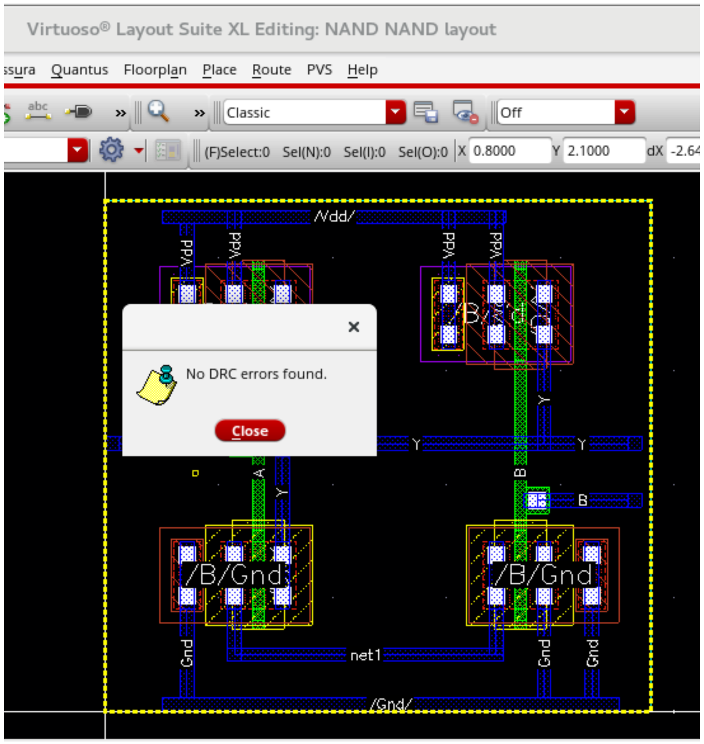
5. Add Contacts and Vias:
    a. Place contacts to connect metal layers vertically (between transistors) and horizontally (to connect different layers).
    b. Add vias to connect metal layers where needed.

6. Route Metal Layers:
    a. Use the routing tools to create metal traces between transistors and connect various components.

7. Add Power and Ground Connections:
    a. Connect the power (Vdd) and ground (Gnd) to your NAND gate layout.

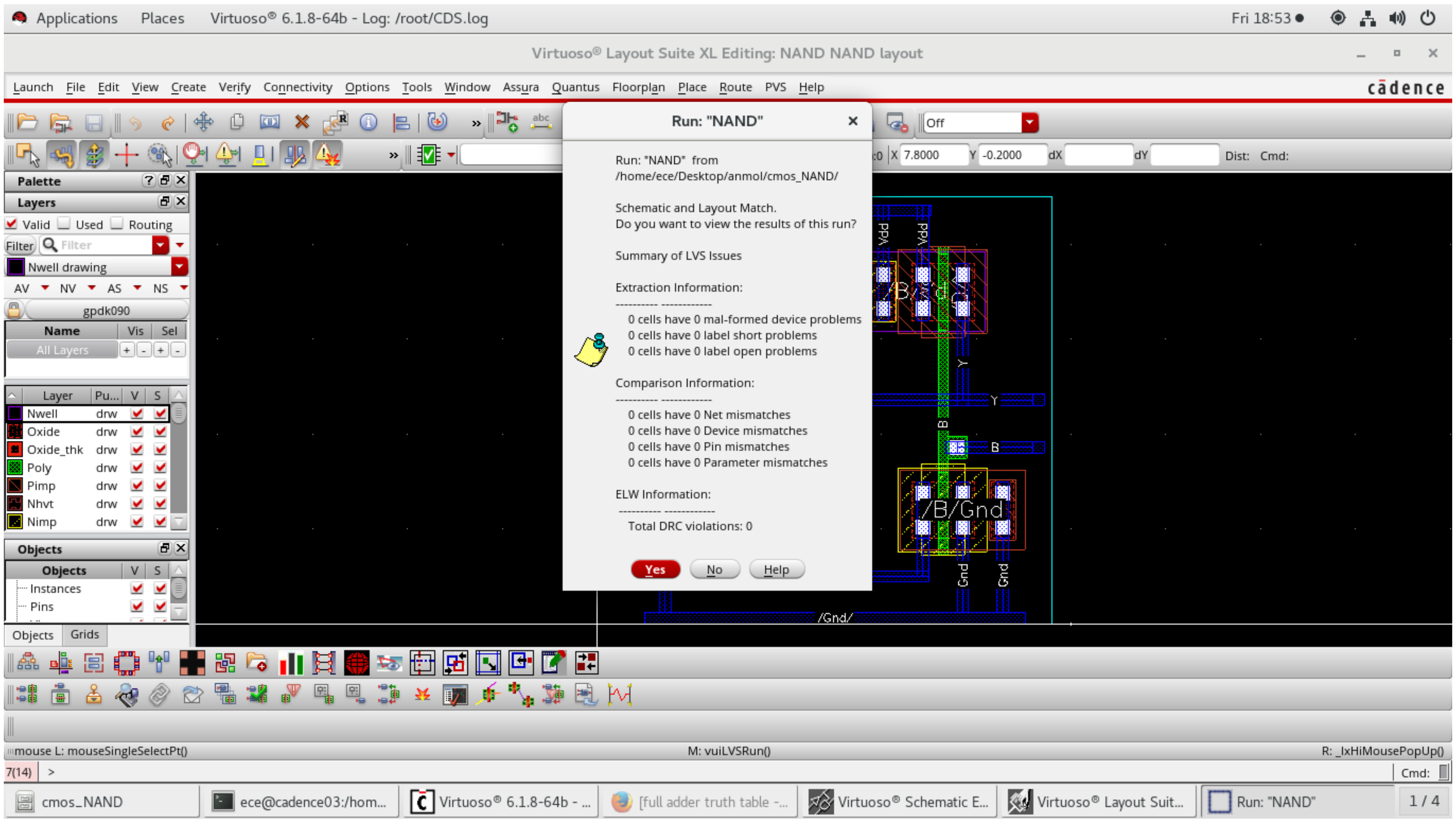


8. Design Rule Check (DRC):
    a. Run a Design Rule Check (DRC) to ensure your layout adheres to the manufacturing rules and constraints.


Virtuoso® Layout Suite XL Editing: NAND NAND layout
ssura Quantus Floorplan Place Route PVS Help
Classic
Off
(F)Select:0 Sel(N):0 Sel(I):0 Sel(O):0 X 0.8000 Y 2.1000 dX -2.64
/Vdd/
Vdd
Y
A
B
/B/Gnd
net1
Gnd
/Gnd/
Close

No DRC errors found.

9. Layout vs. Schematic (LVS) Check:
    a. Perform a Layout vs. Schematic (LVS) check to verify that the layout matches the schematic.


Applications Places Virtuoso® 6.1.8-64b - Log: /root/CDS.log Fri 18:53
Virtuoso® Layout Suite XL Editing: NAND NAND layout
Launch File Edit View Create Verify Connectivity Options Tools Window Assura Quantus Floorplan Place Route PVS Help
cādence
Off
X 7.8000 Y -0.2000 dX dY Dist: Cmd:
Palette
Layers
Valid Used Routing
Nwell drawing
gpdk090
Name Vis Sel
All Layers
Layer Pu... V S
Nwell drw
Oxide drw
Oxide_thk drw
Poly drw
Pimp drw
Nhvt drw
Nimp drw
Objects
Instances
Pins
Objects Grids
Vdd
Y
B
/B/Gnd
Gnd
/Gnd/
mouse L: mouseSingleSelectPt() M: vuiLVSRun() R: _lxHiMousePopUp()
7(14) > Cmd:
cmos_NAND
ece@cadence03:/hom...
Virtuoso® 6.1.8-64b - ...
[full adder truth table -...
Virtuoso® Schematic E...
Virtuoso® Layout Suit...
Run: "NAND"
1 / 4

Run: "NAND"

```
Run: "NAND" from
/home/ece/Desktop/anmol/cmos_NAND/

Schematic and Layout Match.
Do you want to view the results of this run?

Summary of LVS Issues

Extraction Information:
---------- ------------
  0 cells have 0 mal-formed device problems
  0 cells have 0 label short problems
  0 cells have 0 label open problems

Comparison Information:
---------- ------------
  0 cells have 0 Net mismatches
  0 cells have 0 Device mismatches
  0 cells have 0 Pin mismatches
  0 cells have 0 Parameter mismatches

ELW Information:
----------- ------------
  Total DRC violations: 0
```

Yes No Help

2. Construction of XOR gate using NAND gate

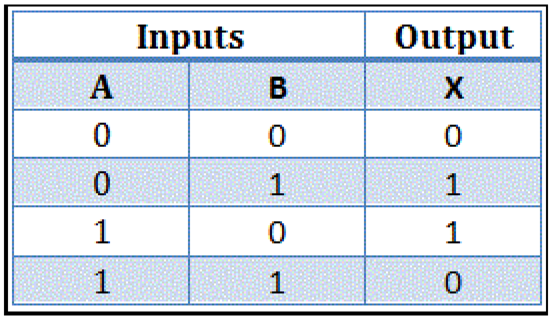
THEORY:

The XOR (exclusive OR) gate is a digital logic gate that performs an exclusive OR operation. In other words, the output of an XOR gate is true (1) only when an odd number of its inputs are true (1). If an even number of inputs are true, the output is false (0). The XOR gate has two inputs, typically A and B, and one output, often labeled as Q.

1. Symbol and Representation:

   The XOR (exclusive OR) gate is represented in circuit diagrams by a shape with two inputs (usually labeled A and B) and one output (typically labeled Q). Inside the shape, the symbol ⊕ or the label "XOR" may denote the operation.

2. Truth Table:



Inputs		Output
A	**B**	**X**
0	0	0
0	1	1
1	0	1
1	1	0

3. Boolean Expression:

   The Boolean expression for the XOR gate output (Q) in terms of inputs A and B is Q=A⊕B. Here, ⊕ it represents the XOR operation.

4. Functional Description:

   The XOR gate outputs a true (1) only when an odd number of its inputs are true. If both inputs are the same (both 0 or both 1), the output is false (0). It functions to detect differences between input values. XOR gates are fundamental in digital circuit design, serving roles in error detection, encryption, and arithmetic operations. They are crucial building blocks for constructing complex digital circuits.

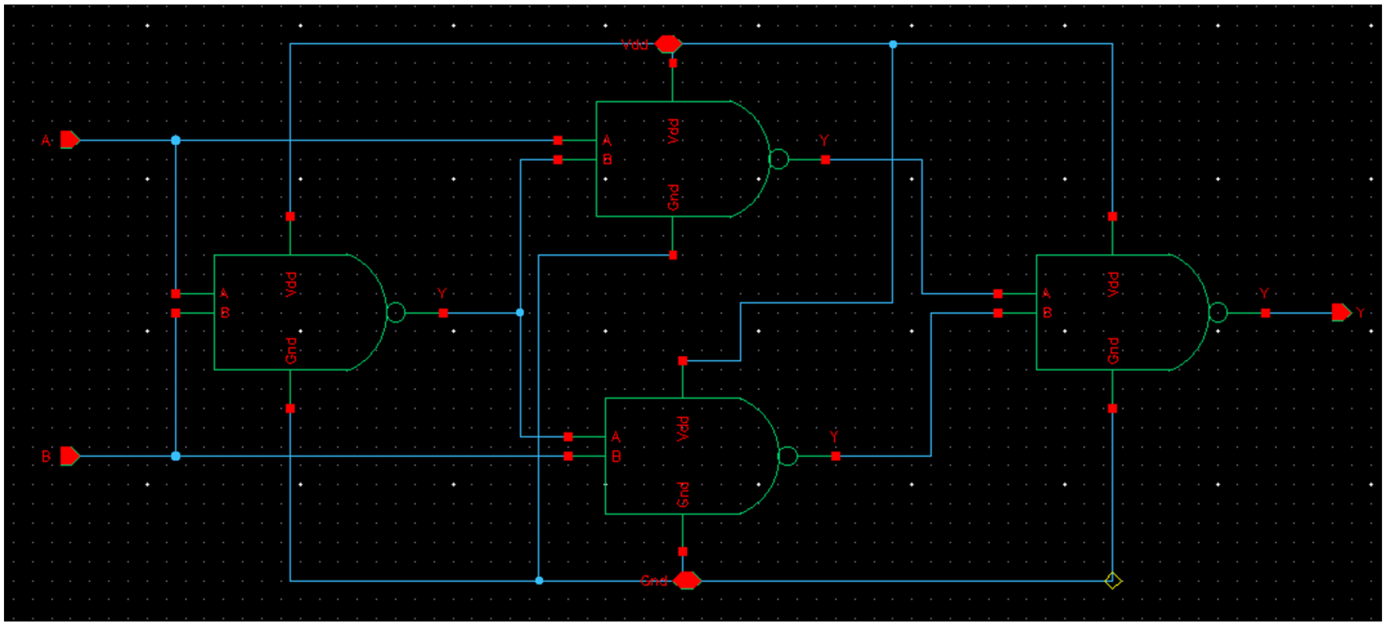
PROCEDURE:

SCHEMATIC:

1. Open the Schematic Window:
   a. Open Cadence Virtuoso and load your existing project.
   b. Create a new schematic cell view for the XOR gate.

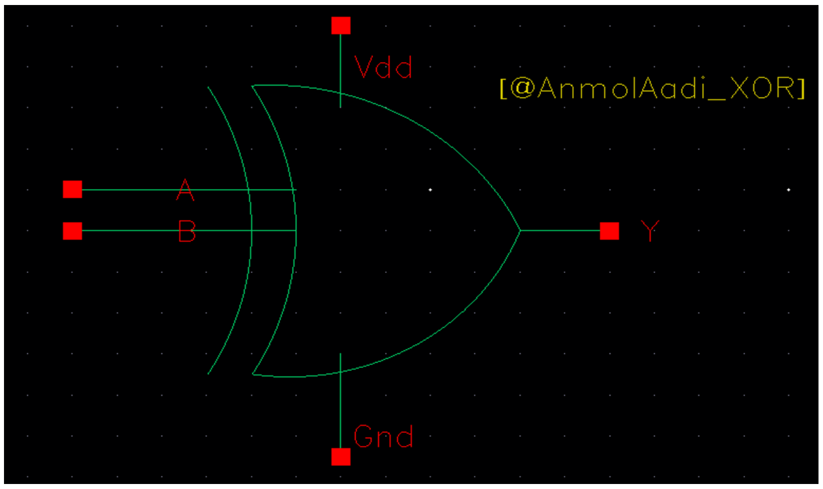
2. Place NAND Gates:

   Assuming you have already designed NAND gates, use them to create an XOR gate. The XOR gate can be implemented using four NAND gates. Connect the gates appropriately to achieve the XOR functionality.



3. Connect Inputs and Outputs:

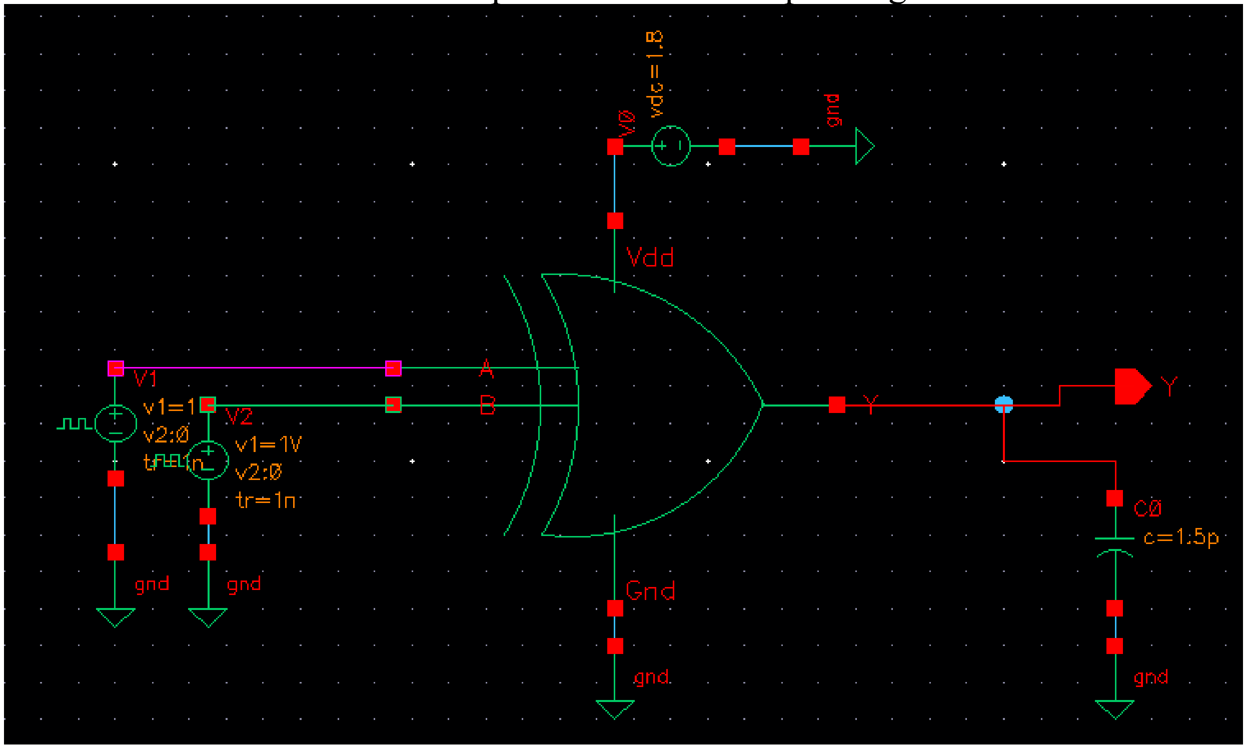
   Connect the inputs (A and B) and the output (Q) to the corresponding pins of the NAND gates.



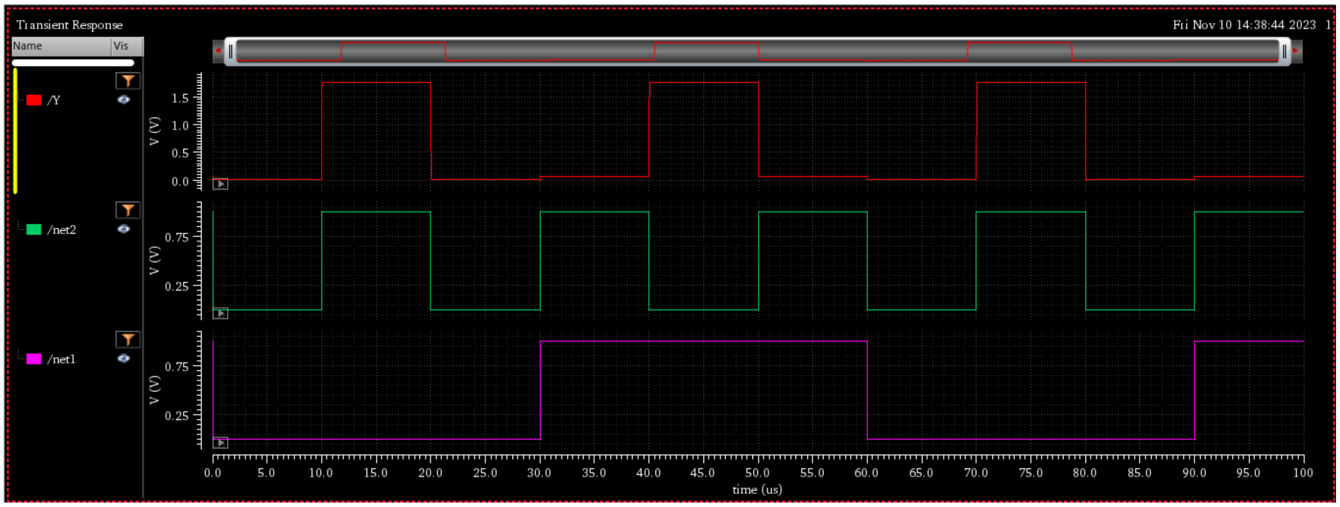
4. Add Power and Ground Connections:

   Connect the power (Vdd) and ground (Gnd) to your circuit.

5. Symbol Creation:
   a. Create a symbol view for the NAND gate.
   b. Assign pins to the symbol for input (A and B) and output (Y).
   c. Associate the pins with the corresponding nodes in the



6. Simulation:
    a. Create a new simulation cell view.
    b. Add a "vpulse" source for each input (A and B) with appropriate parameters (voltage levels, delay, etc.).
    c. Add a voltage probe to the output (Q).
    d. Run a transient simulation to observe the XOR gate's behavior.



LAYOUT:

1. Open the NAND Gate Layout:
    a. Open Cadence Virtuoso and load your existing project.
    b. Open the layout view of the NAND gate that you have already designed.

2. Create a New Layout Cell View for XOR:
    a. In the layout window, create a new layout cell view for the XOR gate.

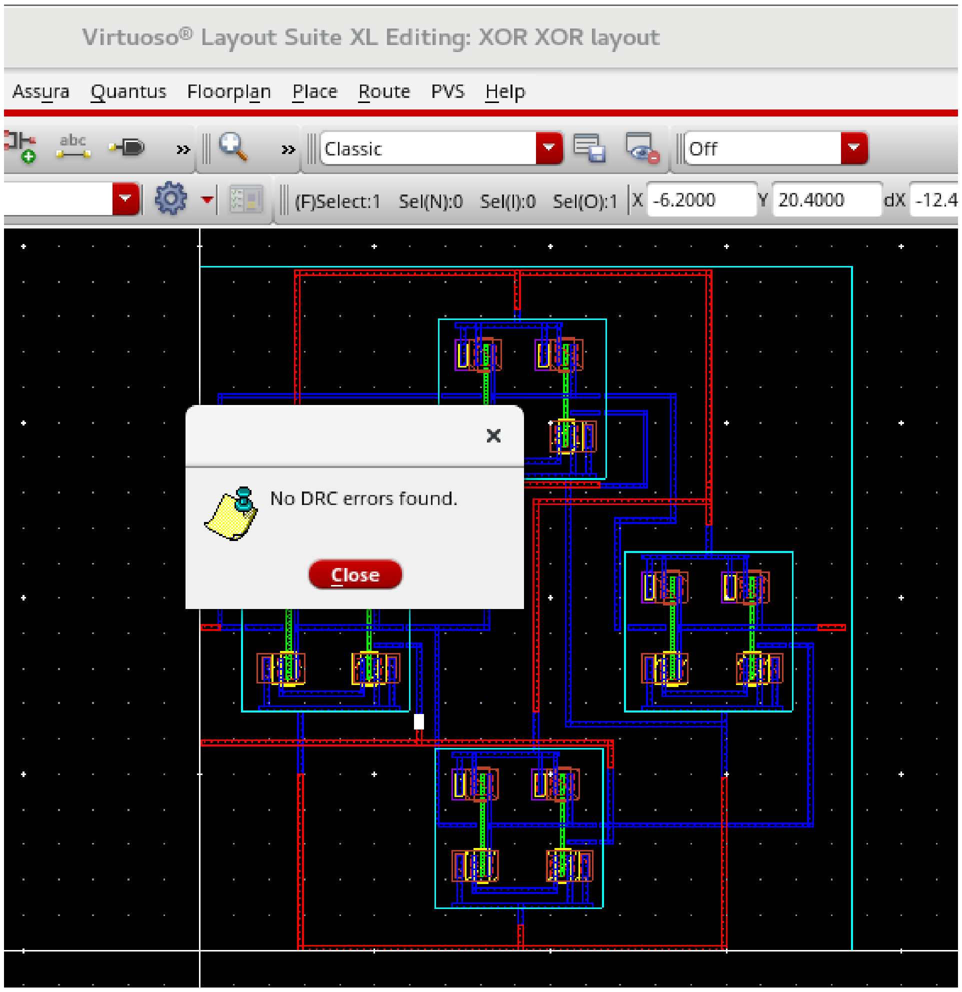
3. Copy and Place NAND Gates:
    a. Copy and paste the NAND gate layout instances from the NAND gate layout view to the XOR gate layout view.

4. Position NAND Gates for XOR Logic:

   a. Arrange the NAND gate instances to form the XOR gate logic. Remember that an XOR gate can be implemented using three NAND gates.
   b. Connect the output of the NAND gates appropriately to form the XOR logic.

5. Add Power and Ground Connections:
   a. Connect the power (Vdd) and ground (Gnd) to your XOR gate layout.

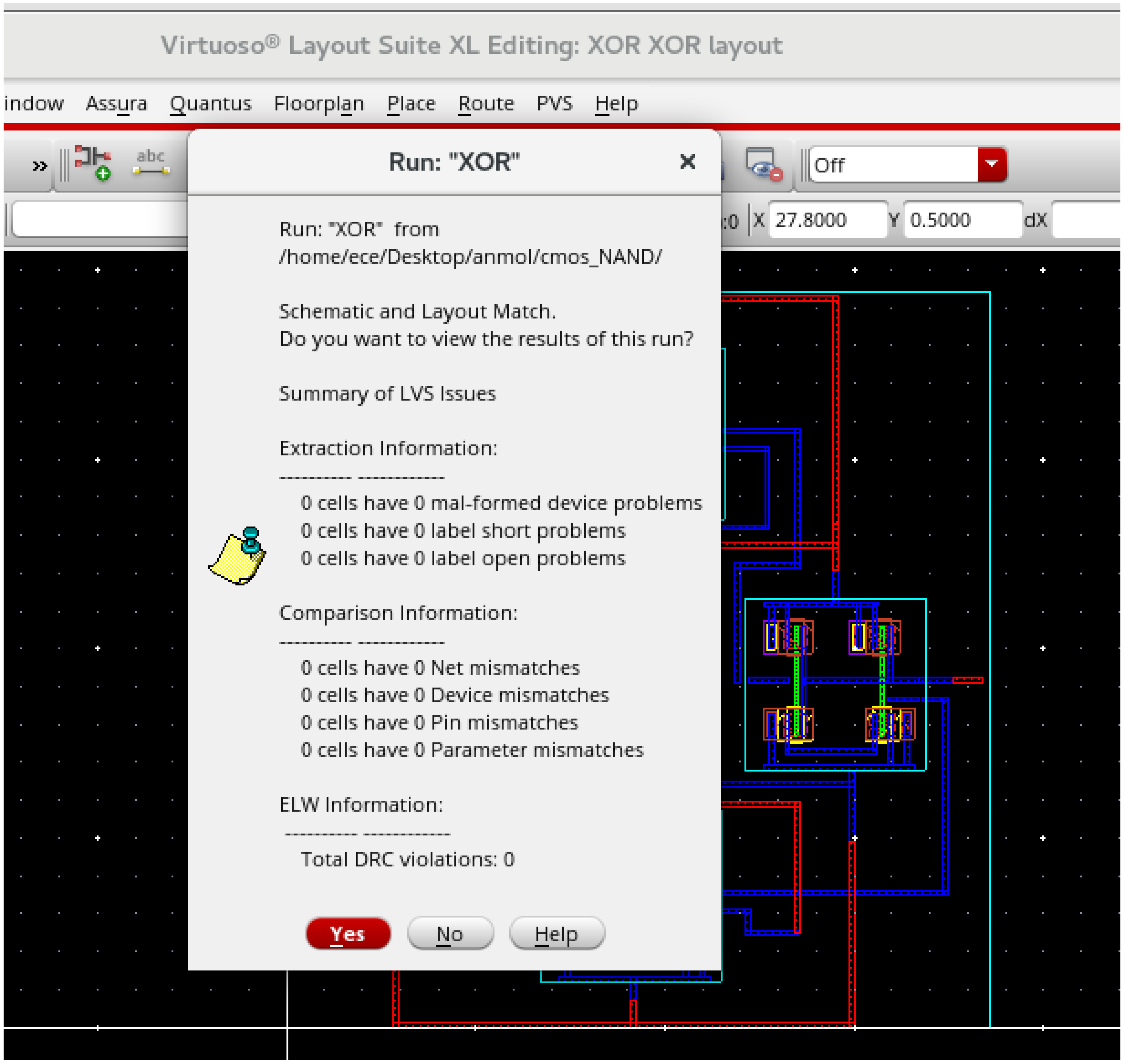
6. Design Rule Check (DRC):
   a. Run a Design Rule Check (DRC) to ensure that your XOR gate layout adheres to the manufacturing rules and constraints.


Virtuoso® Layout Suite XL Editing: XOR XOR layout

(F)Select:1 Sel(N):0 Sel(I):0 Sel(O):1 X -6.2000 Y 20.4000 dX -12.4

No DRC errors found.

7. Layout vs. Schematic (LVS) Check:
   a. Perform a Layout vs. Schematic (LVS) check to verify that the XOR gate layout matches the intended schematic.


Virtuoso® Layout Suite XL Editing: XOR XOR layout
indow Assura Quantus Floorplan Place Route PVS Help
Off
X 27.8000 Y 0.5000 dX
Run: "XOR"
Yes
No
Help

Run: "XOR" from
/home/ece/Desktop/anmol/cmos_NAND/

Schematic and Layout Match.
Do you want to view the results of this run?

Summary of LVS Issues

Extraction Information:
---------- ------------
  0 cells have 0 mal-formed device problems
  0 cells have 0 label short problems
  0 cells have 0 label open problems

Comparison Information:
---------- ------------
  0 cells have 0 Net mismatches
  0 cells have 0 Device mismatches
  0 cells have 0 Pin mismatches
  0 cells have 0 Parameter mismatches

ELW Information:
---------- ------------
  Total DRC violations: 0

3. Construction of Full Adder:

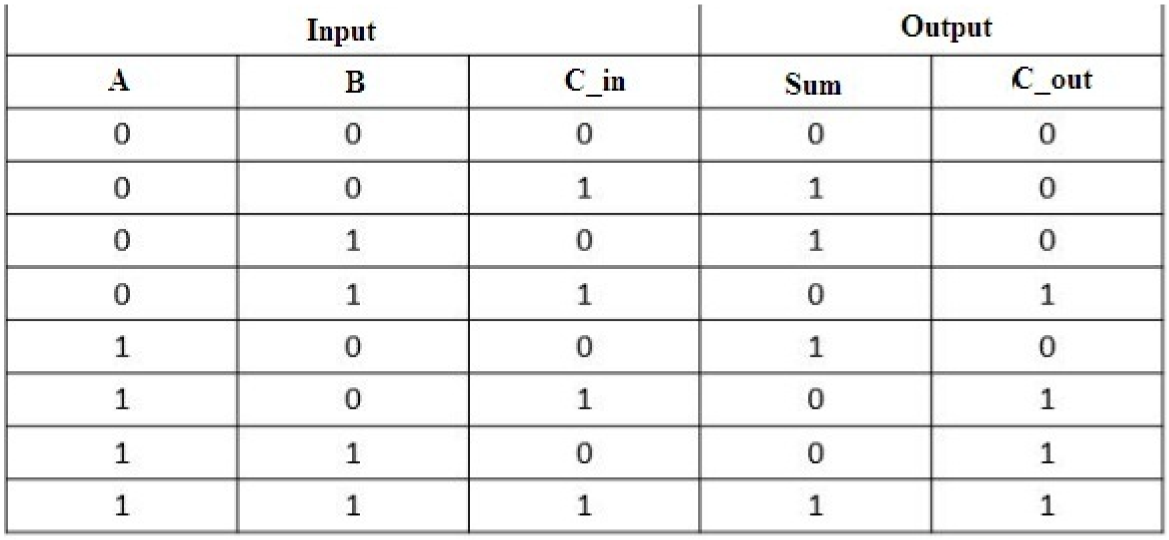
THEORY:

1. Symbol and Representation:

   The Full Adder is a digital circuit that adds three binary digits - two inputs (A and B) and a carry input (Cin). It has two outputs, the sum (S) and the carry out (Cout). In schematic diagrams, the Full Adder is often represented by a block with three inputs and two outputs. The symbols used may vary, but a common representation includes two XOR gates, two AND gates, and an OR gate.

2. Truth Table:

   The truth table for a Full Adder is more extensive than that of a Half Adder, as it accounts for the carry input. The table is as follows:



Input			Output	
A	B	C_in	Sum	C_out
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

3. Boolean Expression:

   The Boolean expressions for the outputs of a Full Adder are as follows:

   $S=A\oplus B\oplus Cin$

   $Cout=(A\wedge B)\vee(Cin\wedge(A\oplus B))$

   Here,

   $\oplus$ Represents the XOR operation, $\wedge$ represents the AND operation, and $\vee$ represents the OR operation.

4. Functional Description:

    The Full Adder performs the addition of three binary digits, considering both the current inputs (A and B) and the carry input (Cin) from the previous stage. The Sum (S) output represents the least significant bit of the addition, and the Carry Out (Cout) represents the carry-over to the next higher bit. The Full Adder is a fundamental building block in arithmetic circuits and is used in the construction of more complex digital systems such as CPUs and microcontrollers.

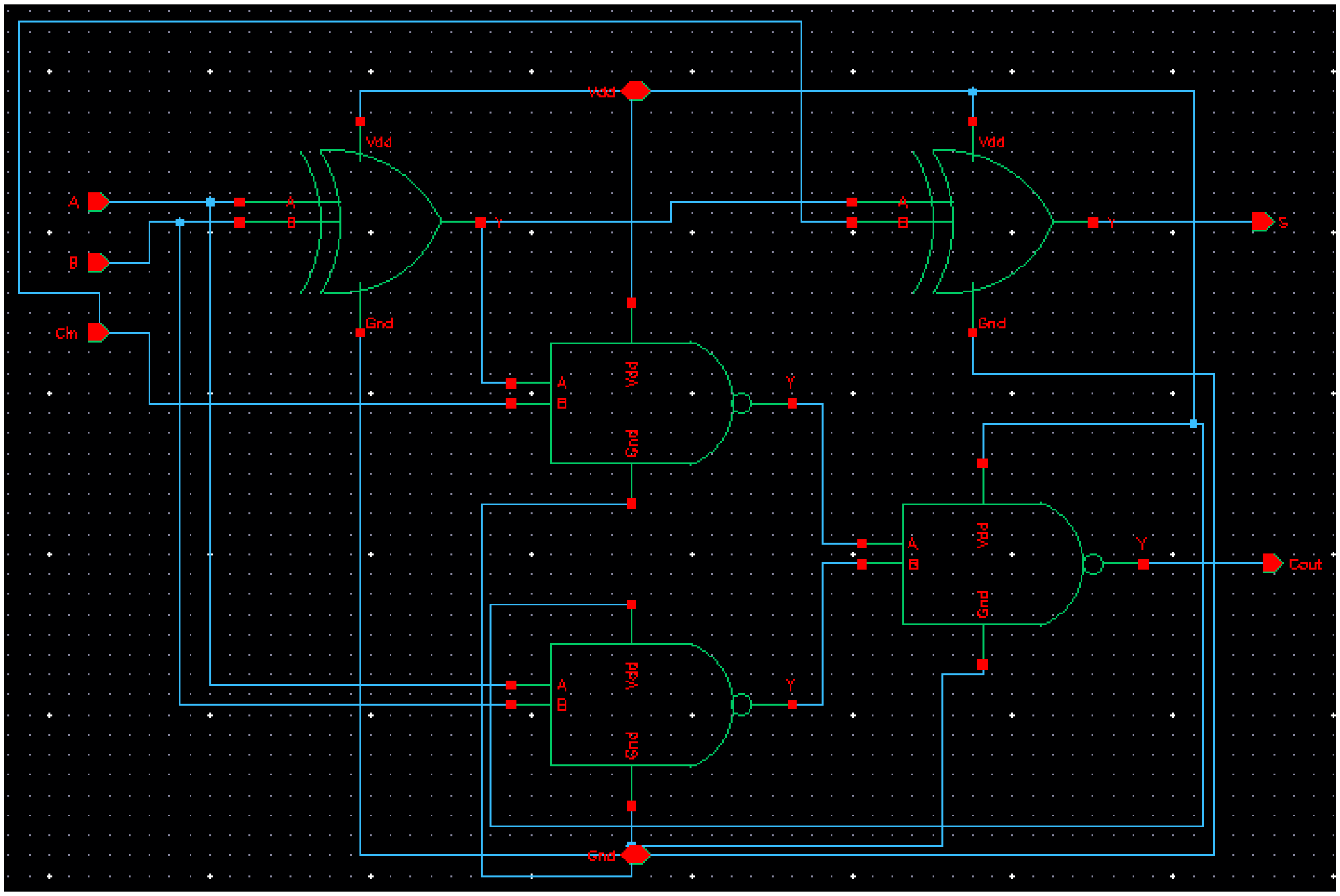
PROCEDURE:

SCHEMATIC:

1. Open the Schematic Window:
    a. Open Cadence Virtuoso and load your existing project.
    b. Create a new schematic cell view for the Full Adder.

2. Place NAND and XOR Gates:

    Assuming you have already designed NAND gates and XOR gates, use them to create a Full Adder. A Full Adder typically requires two XOR gates and three NAND gates. Connect them appropriately to achieve the Full Adder functionality.



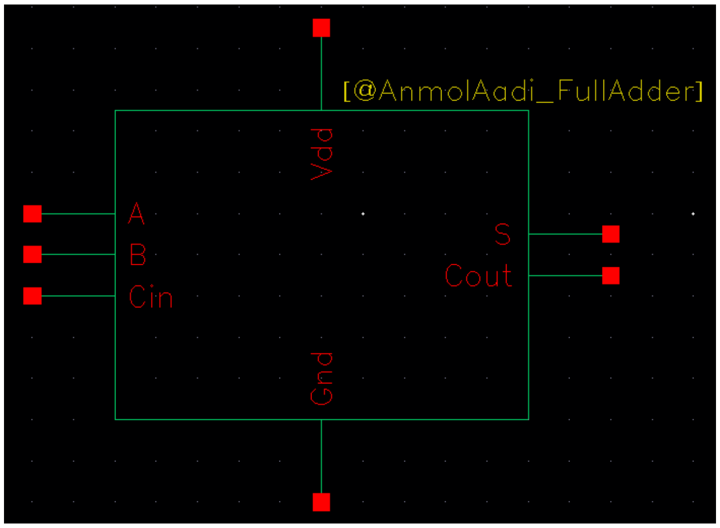
3. Connect Inputs and Outputs:

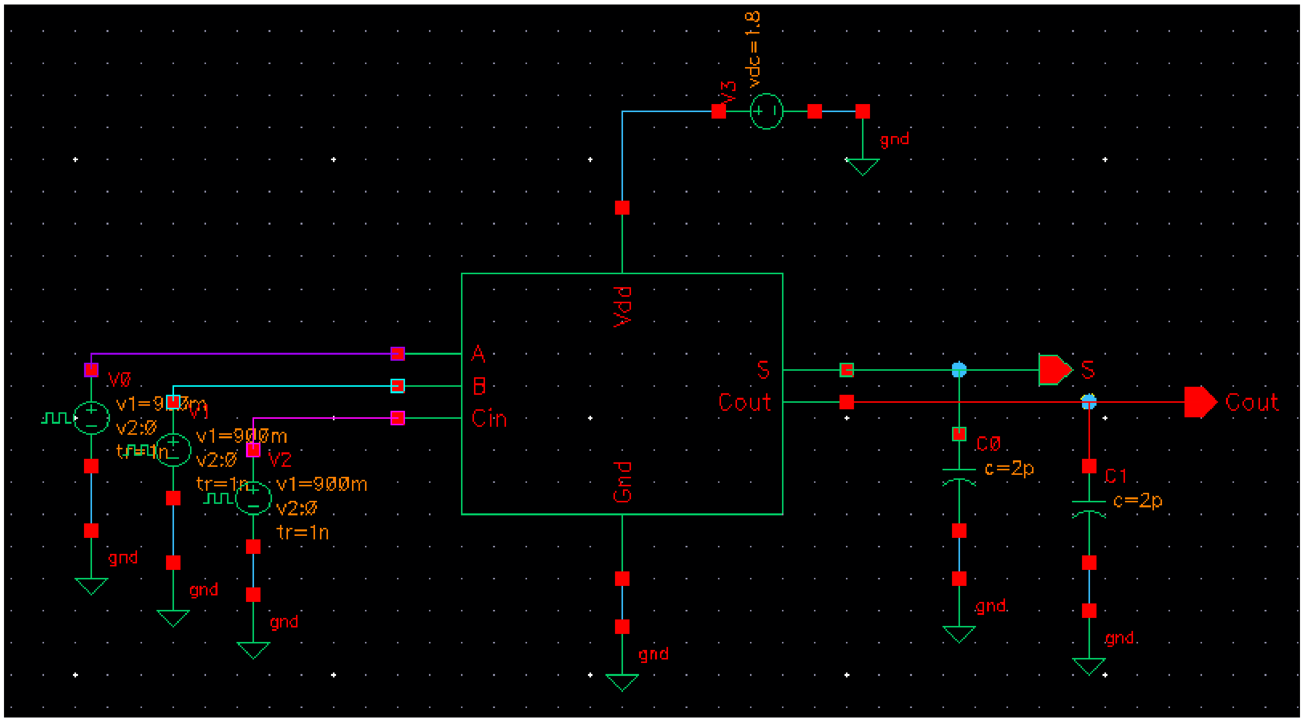
   Connect the inputs (A, B, Cin) and the outputs (Sum, Cout) to the corresponding pins of the NAND and XOR gates.

4. Add Power and Ground Connections:

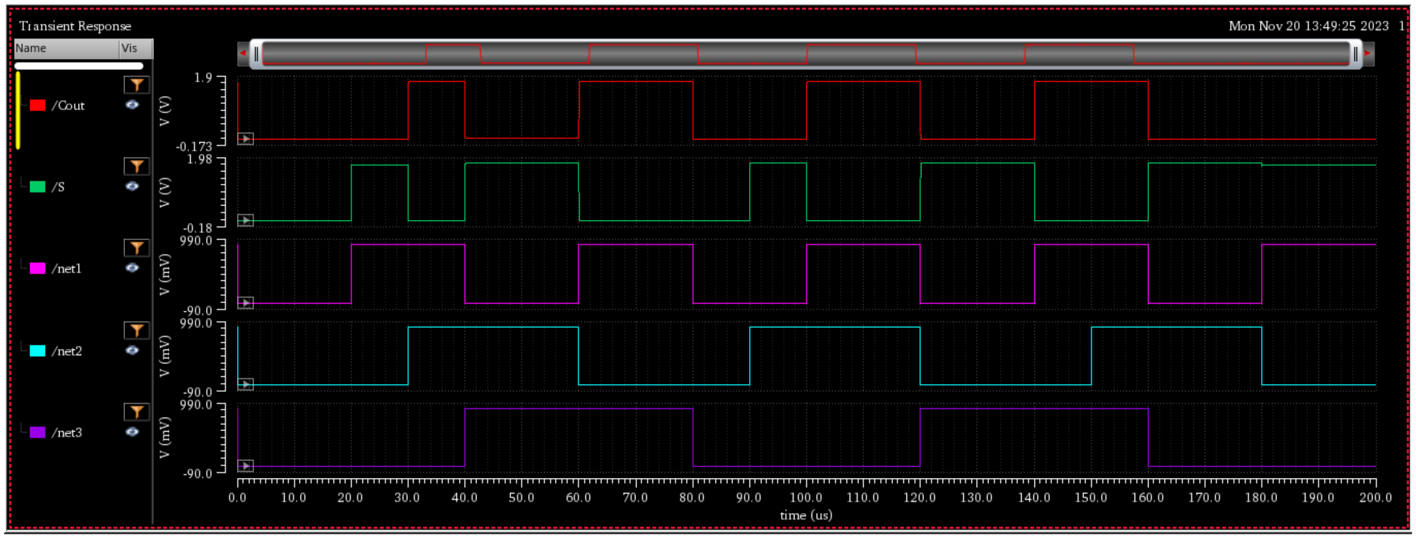
   Connect the power (Vdd) and ground (Gnd) to your circuit.

5. Symbol Creation:
   a. Create a symbol view for the NAND gate.
   b. Assign pins to the symbol for input (A and B) and output (Y).
   c. Associate the pins with the corresponding nodes in the schematic.





5. Simulation:
    a. Create a new simulation cell view.
    b. Add "vpulse" sources for each input (A, B, Cin) with appropriate parameters (voltage levels, delay, etc.).
    c. Add voltage probes to the outputs (Sum, Cout).
    d. Run a transient simulation to observe the Full Adder's behavior.



LAYOUT:

1. Open NAND and XOR Gate Layouts:
    a. Open Cadence Virtuoso and load your existing project.
    b. Open the layout views of the NAND gate and XOR gate that you have already designed.

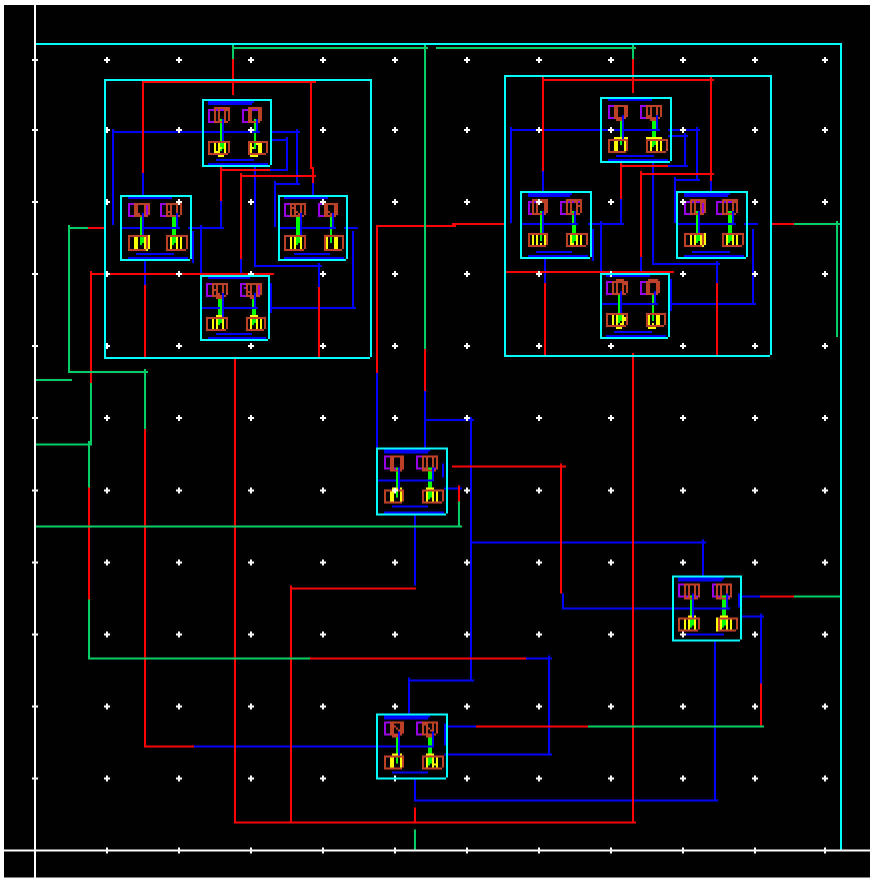
2. Create a New Layout Cell View for Full Adder:
    a. In the layout window, create a new layout cell view for the Full Adder.

3. Copy and Place NAND and XOR Gates:
    a. Copy and paste the NAND gate and XOR gate layout instances from their respective layout views to the Full Adder layout view.

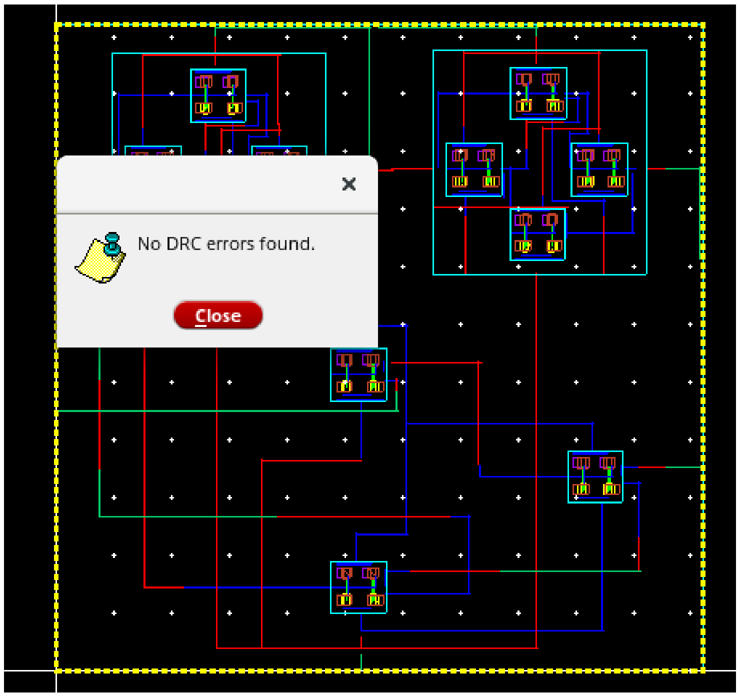
4. Position Gates for Full Adder Logic:

    a. Arrange the NAND and XOR gate instances to form the Full Adder logic. Recall that a Full Adder consists of XOR gates, AND gates, and additional logic for sum and carry-out generation.
    b. Connect the output of the XOR gates and AND gates appropriately to form the Full Adder logic.

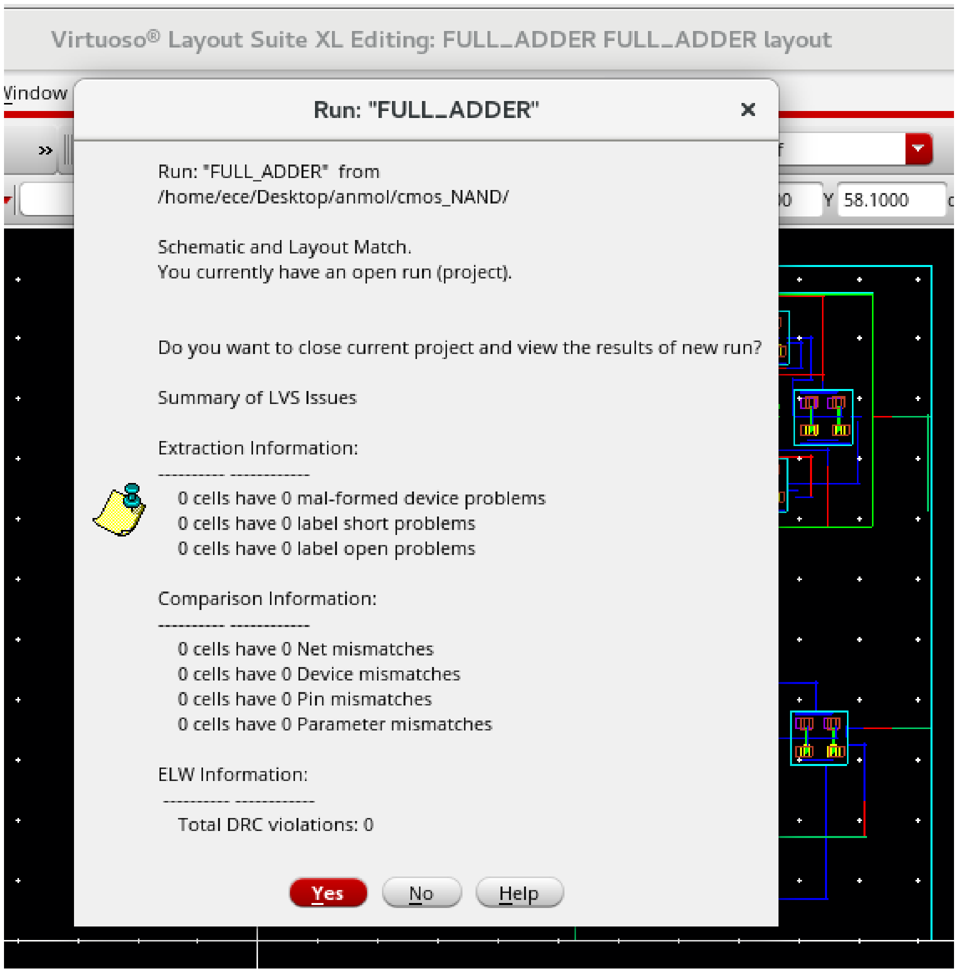
5. Add Power and Ground Connections:
    a. Connect the power (Vdd) and ground (Gnd) to your Full Adder layout.



6. Design Rule Check (DRC):
    a. Run a Design Rule Check (DRC) to ensure that your Full Adder layout adheres to the manufacturing rules and constraints.


No DRC errors found.

7. Layout vs. Schematic (LVS) Check:
    a. Perform a Layout vs. Schematic (LVS) check to verify that the Full Adder layout matches the intended schematic.


Virtuoso® Layout Suite XL Editing: FULL_ADDER FULL_ADDER layout

Run: "FULL_ADDER"

Run: "FULL_ADDER" from
/home/ece/Desktop/anmol/cmos_NAND/

Schematic and Layout Match.
You currently have an open run (project).

Do you want to close current project and view the results of new run?

Summary of LVS Issues

Extraction Information:
---------- ------------
0 cells have 0 mal-formed device problems
0 cells have 0 label short problems
0 cells have 0 label open problems

Comparison Information:
---------- ------------
0 cells have 0 Net mismatches
0 cells have 0 Device mismatches
0 cells have 0 Pin mismatches
0 cells have 0 Parameter mismatches

ELW Information:
---------- ------------
Total DRC violations: 0

4. Construction of 3-bit Ripple Carry Adder:

THEORY:

1. Symbol and Representation:

   The 3-bit ripple carry adder is a digital circuit designed to add three sets of binary digits. It consists of three full adders connected in series, where the carry-out from each full adder serves as the carry-in for the next. The symbol for a 3-bit ripple carry adder typically shows three inputs (A[2:0], B[2:0]), three outputs (Sum[2:0]), and a carry-out (Cout). The carry-in (Cin) is assumed to be zero for the first adder.

2. Truth Table:

   The truth table for a 3-bit ripple carry adder is a comprehensive representation of all possible combinations of input bits (A[2:0], B[2:0]) and the corresponding outputs (Sum[2:0], Cout). The carry-out from each stage becomes the carry-in for the next.

3. Boolean Expression:

   The Boolean expressions for the outputs of each bit position in a 3-bit ripple carry adder are similar to those of a single full adder but extended to accommodate the three bits. For instance, for bit position 0:

   $S0=A0\oplus B0\oplus Cin$

   $Cout0=(A0\land B0)\lor(Cin\land(A0\oplus B0))$

   Similar expressions apply to bit positions 1 and 2.

4. Functional Description:

   The 3-bit ripple carry adder performs binary addition on three sets of input bits (A[2:0], B[2:0]). The addition is carried out bit-wise, with each full adder stage considering the current input bits and the carry-in from the previous stage. The outputs include the sum bits (Sum[2:0]) representing the least significant to most significant bits of the addition and the final carry-out (Cout) indicating the overflow to the next higher bit. Ripple carry adders are simple but suffer from longer propagation delays due to the sequential nature of the carry.

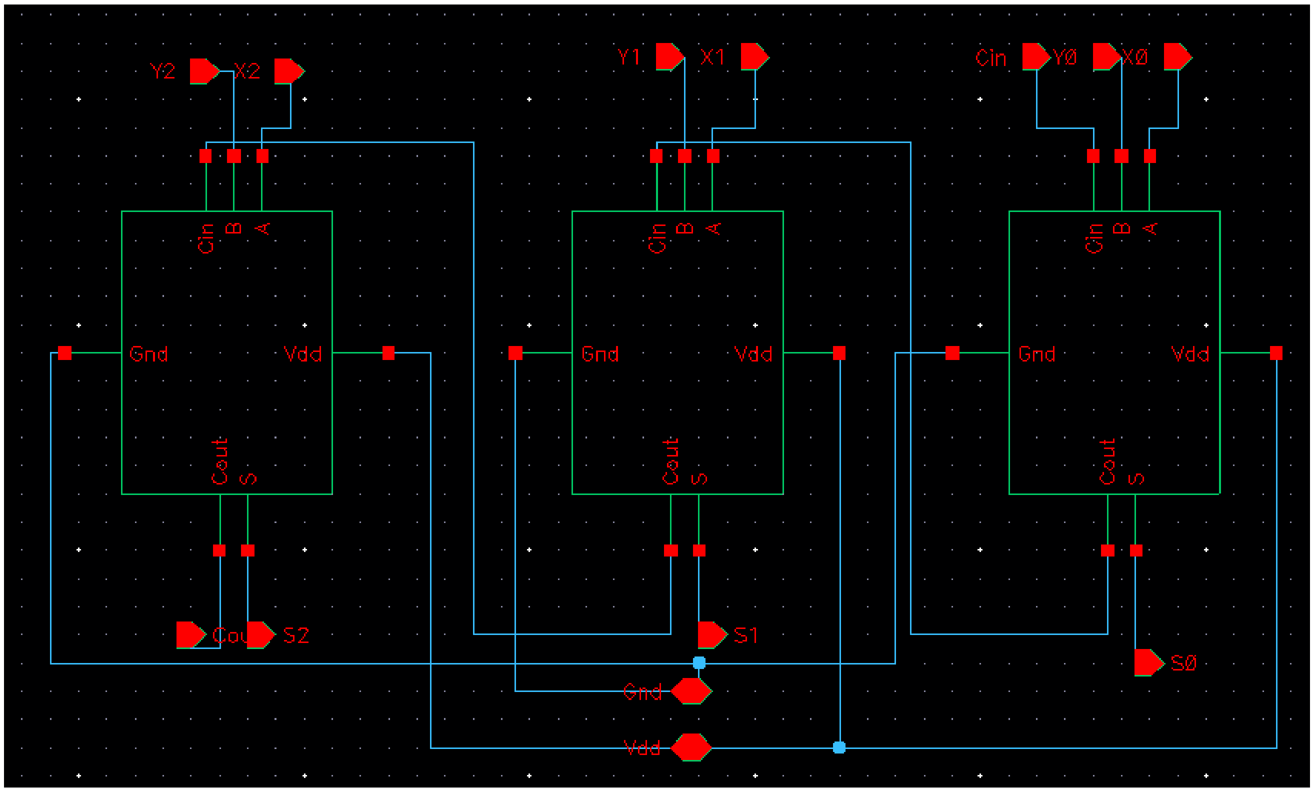
PROCEDURE:

SCHEMATIC:

1. Open the Schematic Window:
   a. Open Cadence Virtuoso and load your existing project.
   b. Create a new schematic cell view for the 3-bit full adder.

2. Place Full Adders:

   Place three instances of your pre-designed full adders. Connect them in series, where the carry-out (Cout) of each full adder is connected to the carry-in (Cin) of the next one.

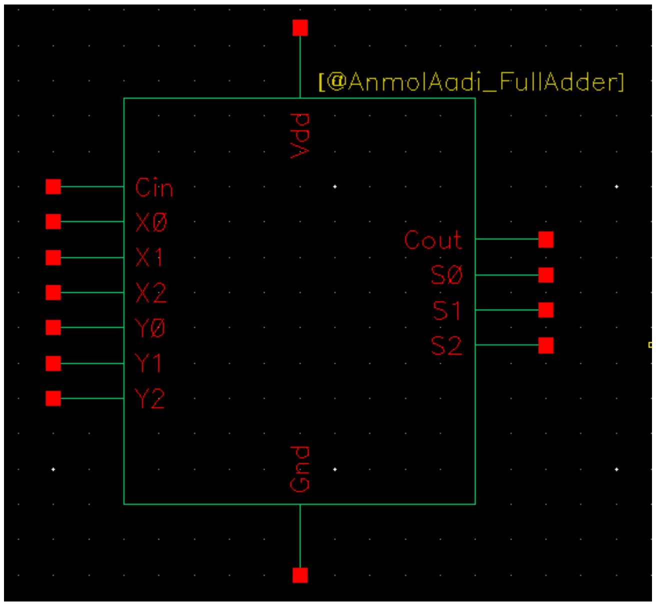


3. Connect Inputs and Outputs:

   a. Connect the inputs (A[2:0], B[2:0]) and the carry-in (Cin) to the corresponding pins of the full adders.
   b. Connect the outputs (Sum[2:0], Cout[2:0]) to the desired pins.

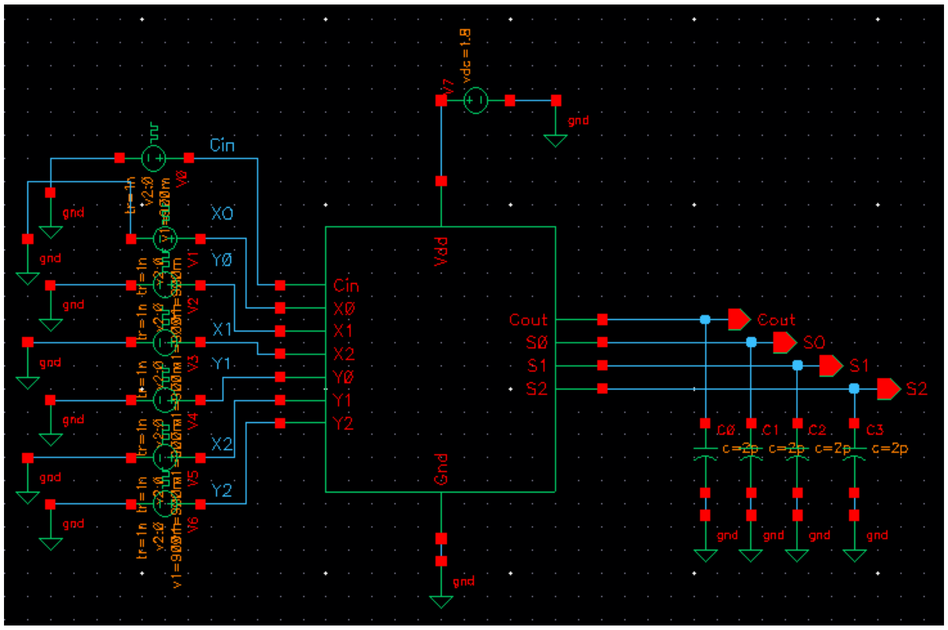
4. Add Power and Ground Connections:

   Connect the power (Vdd) and ground (Gnd) to your circuit.

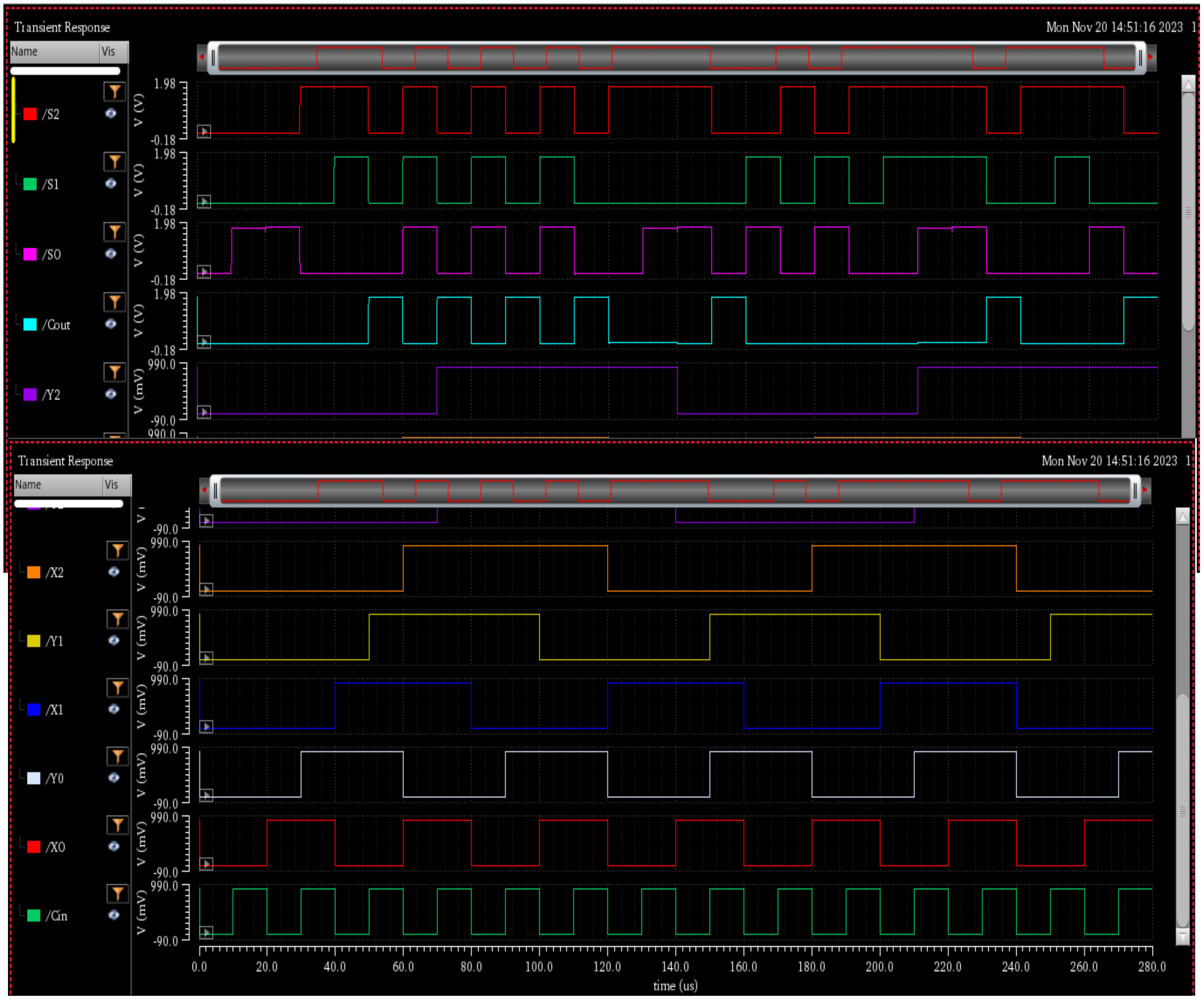


5. Save and Extract:

   Save your schematic and then perform a simulation netlist extraction.



6. Simulation:
   a. Create a new simulation cell view.
   b. Add "vpulse" sources for each input (A[2:0], B[2:0], Cin) with appropriate parameters (voltage levels, delay, etc.).
   c. Add voltage probes to the outputs (Sum[2:0], Cout[2:0]).
   d. Run a transient simulation to observe the 3-bit full adder's behavior.



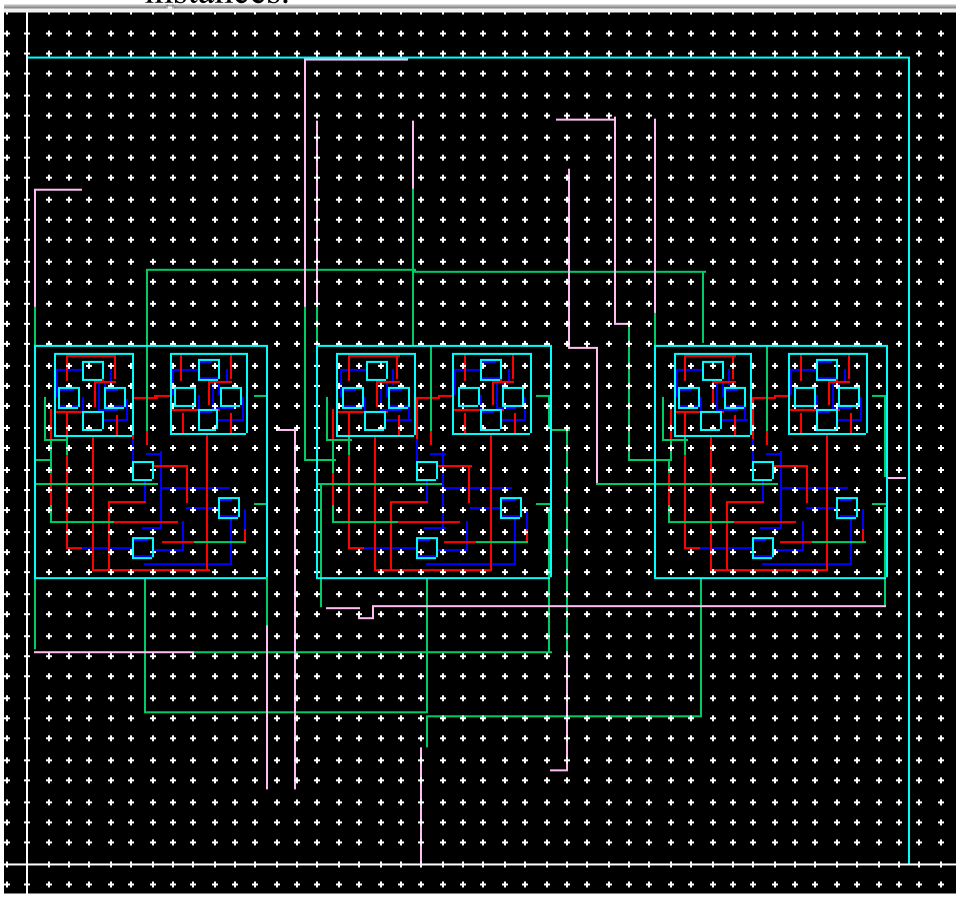
SCHEMATIC:

1. Open Full Adder Layout:
   a. Open Cadence Virtuoso and load your existing project.
   b. Open the layout view of the Full Adder that you have already designed.

2. Create a New Layout Cell View for 3-bit Full Adder:
   a. In the layout window, create a new layout cell view for the 3-bit Full Adder.

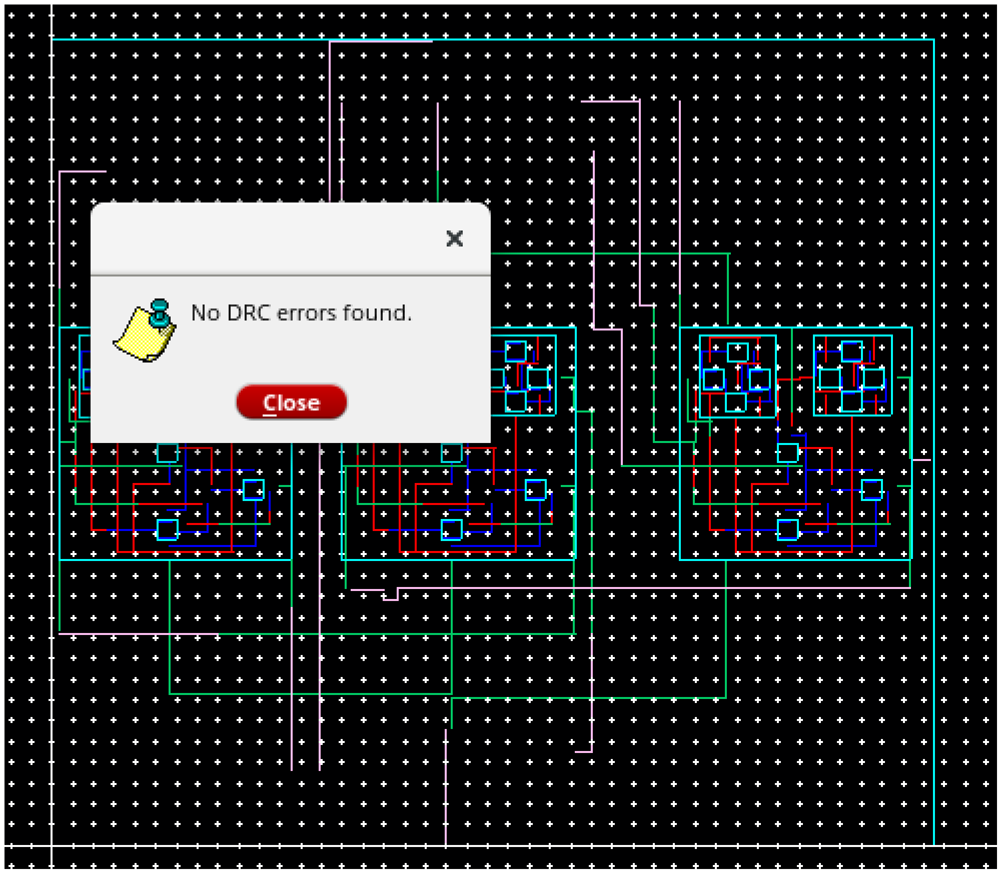
3. Copy and Place Full Adder Instances:
   a. Copy and paste the Full Adder layout instance from the original layout view to the 3-bit Full Adder layout view. Repeat this process for three instances (one for each bit position).

4. Position Full Adder Instances:
   a. Arrange the three Full Adder instances to form the 3-bit Full Adder. Connect the carry-out (Cout) of each bit to the carry-in (Cin) of the next bit.
   b. Ensure proper alignment and spacing between the Full Adder instances.



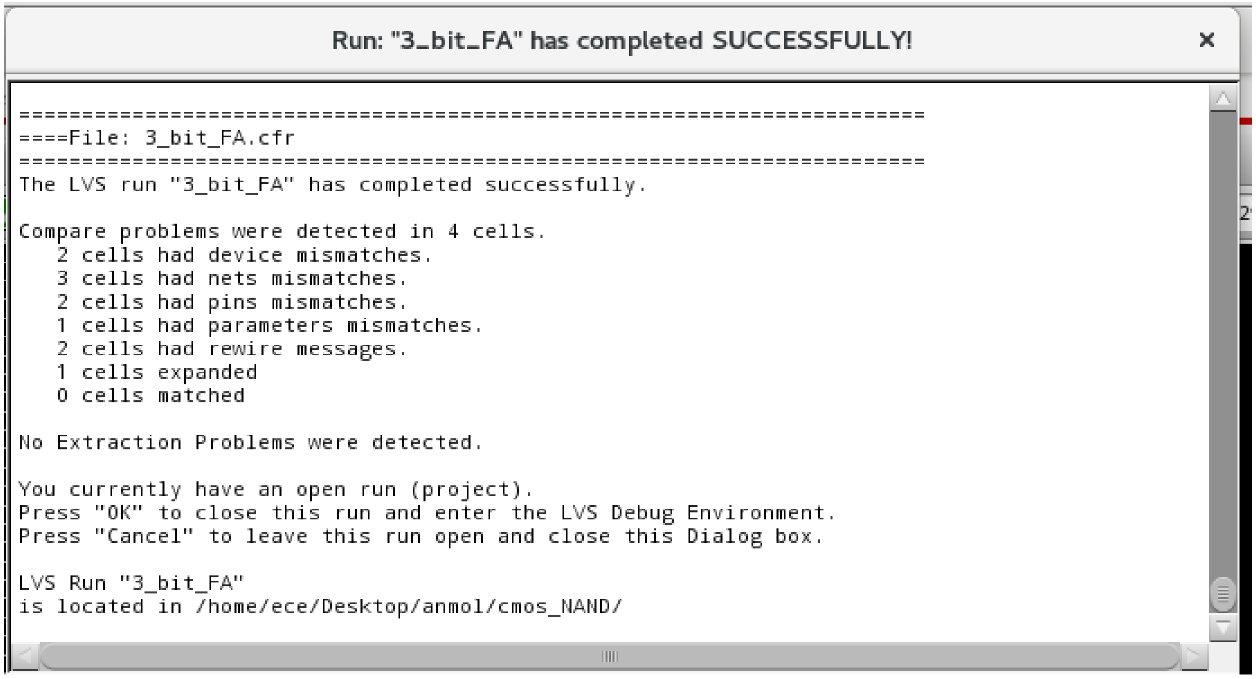
5. Add Power and Ground Connections:
    a. Connect the power (Vdd) and ground (Gnd) to your 3-bit Full Adder layout.

6. Design Rule Check (DRC):
    a. Run a Design Rule Check (DRC) to ensure that your 3-bit Full Adder layout adheres to the manufacturing rules and constraints.

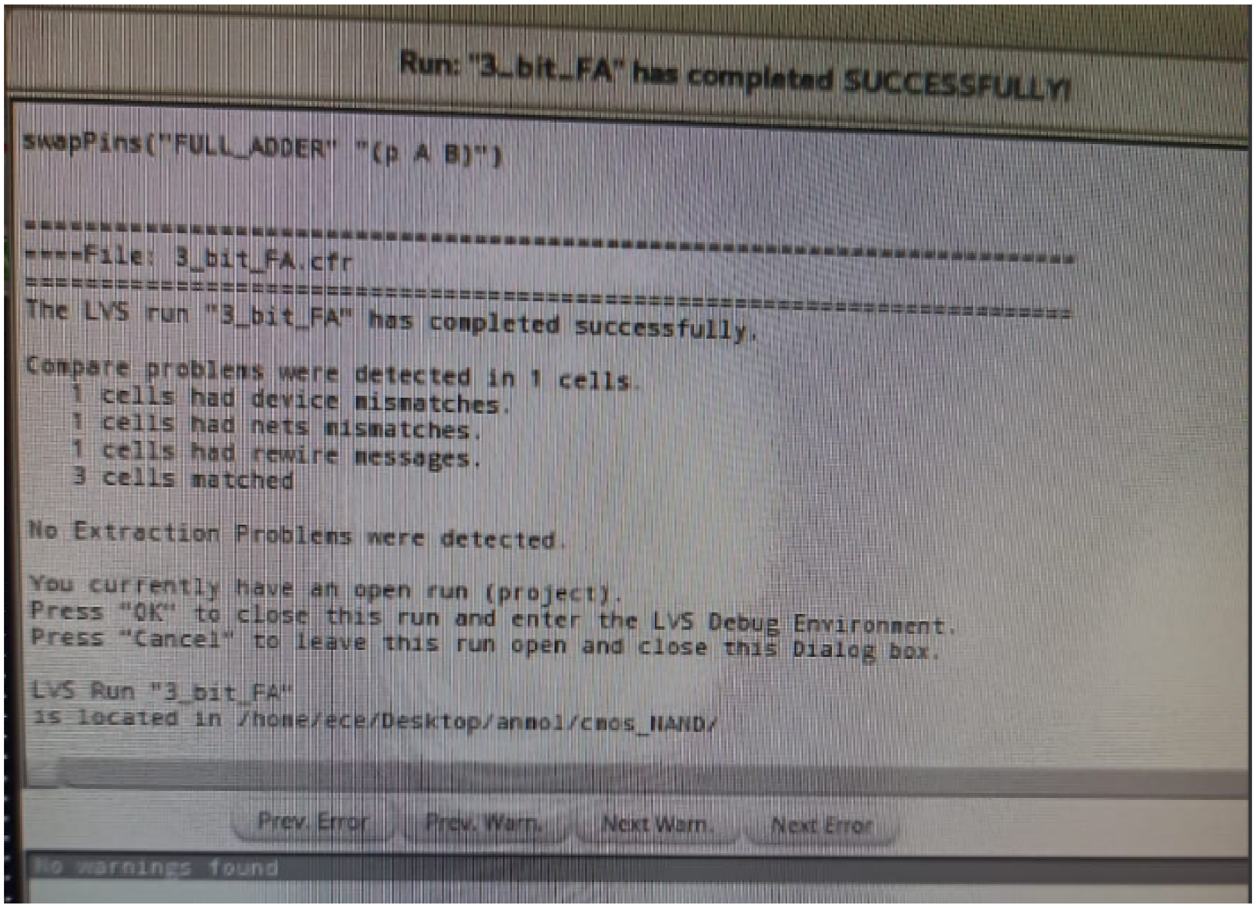


No DRC errors found.

7. Layout vs. Schematic (LVS) Check:
    a. Perform a Layout vs. Schematic (LVS) check to verify that the 3-bit Full Adder layout matches the intended schematic.
    b. The following figures show the before and after of LVS error rectification that was possible under our expertise.



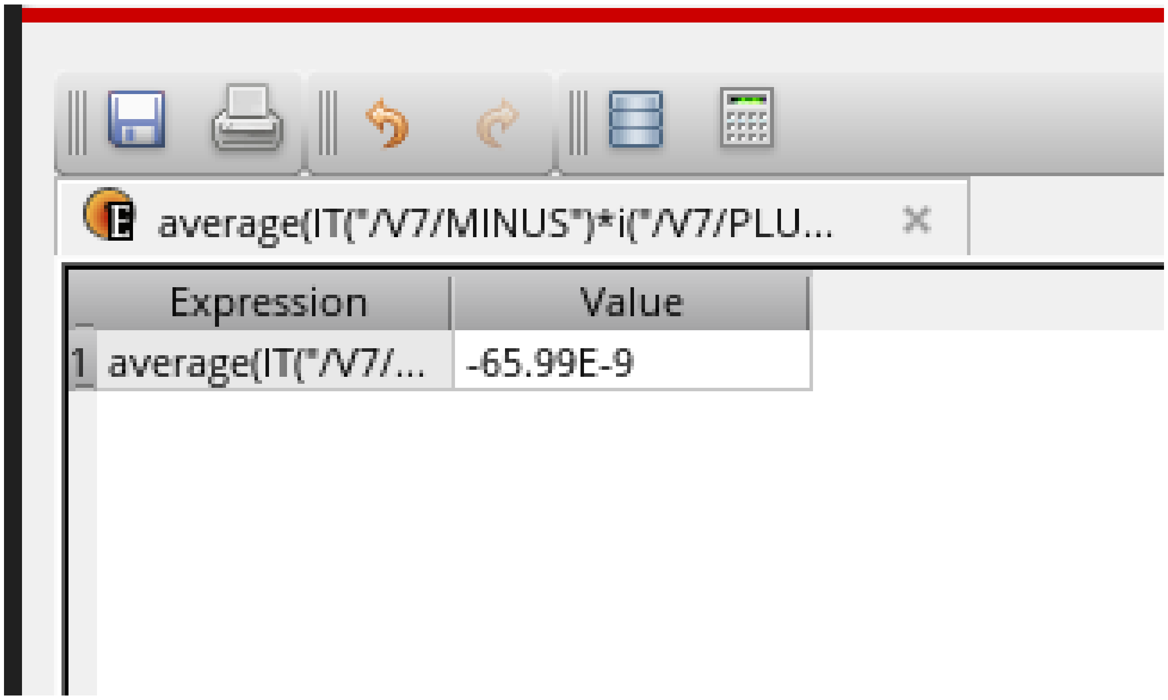
```
========================================================================
====File: 3_bit_FA.cfr
========================================================================
The LVS run "3_bit_FA" has completed successfully.

Compare problems were detected in 4 cells.
   2 cells had device mismatches.
   3 cells had nets mismatches.
   2 cells had pins mismatches.
   1 cells had parameters mismatches.
   2 cells had rewire messages.
   1 cells expanded
   0 cells matched

No Extraction Problems were detected.

You currently have an open run (project).
Press "OK" to close this run and enter the LVS Debug Environment.
Press "Cancel" to leave this run open and close this Dialog box.

LVS Run "3_bit_FA"
is located in /home/ece/Desktop/anmol/cmos_NAND/
```



```
swapPins("FULL_ADDER" "(p A B)")


========================================================================
====File: 3_bit_FA.cfr
========================================================================
The LVS run "3_bit_FA" has completed successfully.

Compare problems were detected in 1 cells.
   1 cells had device mismatches.
   1 cells had nets mismatches.
   1 cells had rewire messages.
   3 cells matched

No Extraction Problems were detected.

You currently have an open run (project).
Press "OK" to close this run and enter the LVS Debug Environment.
Press "Cancel" to leave this run open and close this Dialog box.

LVS Run "3_bit_FA"
is located in /home/ece/Desktop/anmol/cmos_NAND/
```

```
No warnings found
```

AVERAGE POWER CALCULATION OF 3BIT RIPPLE CARRY ADDER:


average(IT("/V7/MINUS")*i("/V7/PLU...

	Expression	Value
1	average(IT("/V7/...	-65.99E-9

The average power calculated is **-65.99e-9**

Steps followed:

1. First, the transient analysis was done,
2. From the graph, Vds positive and negative terminals were selected and sent to the calculator,
3. Then, they were multiplied, and the "average" function was used on them.
4. We obtained the result.

Delay calculation;

1. Calculating the delay in output with respect to each individual output,

delay(Cin) with respect to S0	= **0.04 us**.
delay(Cin) with respect to S1	= **0.04 us**.
delay(Cin) with respect to S2	= **0.28 us**.
delay(Cin) with respect to Cout	= **0.08 us**.