DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERIN
MTECH – MICROELCTRONICS AND VLSI

DIGITAL IC DESIGN LAB (EC1062)

LAB ASSIGNMENT - 1

Submitted By:

ANKIT KUMAR

24EC4224

Submitted To:

Dr. Hemanta Kumar Mondal

Assistant Professor, Department of ECE, NIT Durgapur

Submitted on:

Sign-____________

Design 1: Half Adder

Objective:

To design and implement a Half Adder circuit using basic logic gates.

Specification:

The Half Adder adds two 1-bit binary numbers (A and B) and produces a Sum (S) and Carry (C).
Formulas:
Sum (S) = A B
Carry (C) = A · B

Truth Table:

A	B	Sum (S)	Carry (C)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Logic/ Block-level implementation:

The Half Adder is implemented using an XOR gate for Sum and an AND gate for Carry.

Block Diagram:

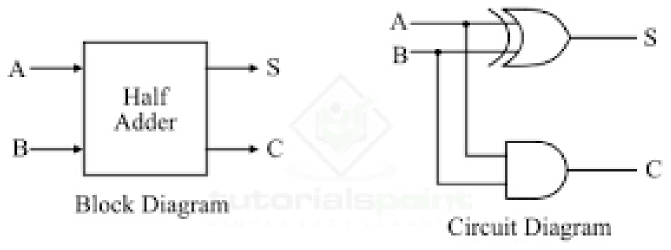


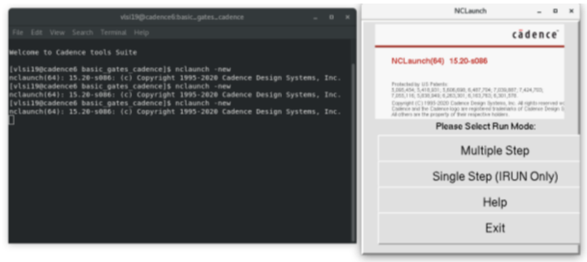
Figure 1 - Half Adder

RTL Coding:

```
module half_adder (
    input A,
    input B,
    output Sum,
    output Carry
);
    assign Sum = A ^ B;
    assign Carry = A & B;
endmodule
```

Simulation Setup:

The simulation was carried out using Cadence NCLaunch. The input waveforms were provided, and output waveforms were observed.



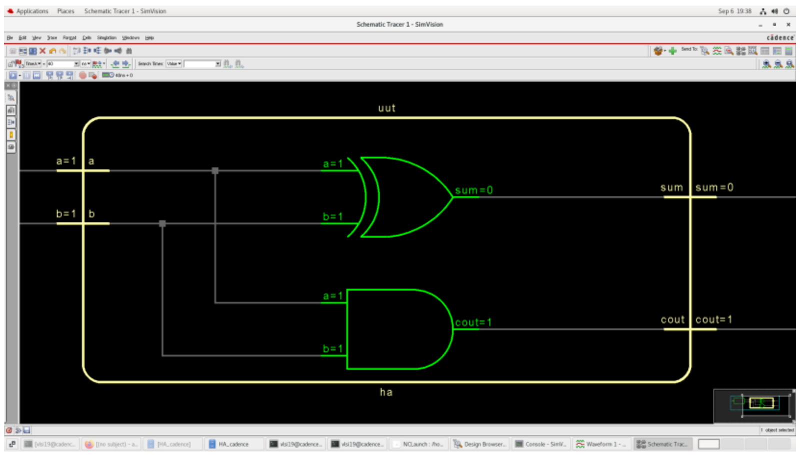
```
Welcome to Cadence tools Suite

[vlsi19@cadence6 basic_gates_cadence]$ nclaunch -new
nclaunch(64): 15.20-s086: (c) Copyright 1995-2020 Cadence Design Systems, Inc.
[vlsi19@cadence6 basic_gates_cadence]$ nclaunch -new
nclaunch(64): 15.20-s086: (c) Copyright 1995-2020 Cadence Design Systems, Inc.
[vlsi19@cadence6 basic_gates_cadence]$ nclaunch -new
nclaunch(64): 15.20-s086: (c) Copyright 1995-2020 Cadence Design Systems, Inc.
```

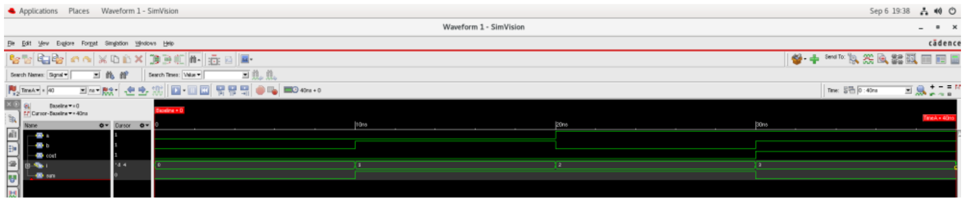
Experimental Results:

The expected results were observed from the simulation, and they matched the theoretical truth table for the circuit.

Schematic Diagram:



Functional Verification Waveform:



Observations:

The circuit worked as expected. Propagation delay was observed for the combinational circuit.

Sign-____________

Design 2: Full Adder

Objective:

To design and implement a Full Adder circuit using Half Adders.

Specification:

The Full Adder adds three 1-bit binary numbers (A, B, Cin) and produces a Sum (S) and Carry (Cout).
Formulas:
Sum (S) = A □ B □ Cin
Carry (Cout) = (A · B) + (Cin · (A □ B))

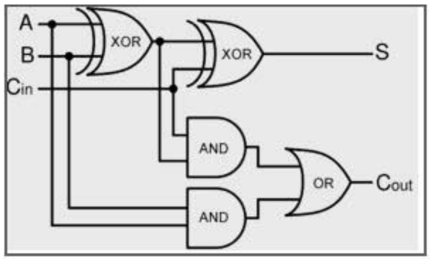
Truth Table:

A	**B**	**Cin**	**Sum (S)**	**Carry (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

Logic/ Block-level implementation:

The Full Adder is implemented using two Half Adders and an OR gate for final Carry.

Block Diagram:



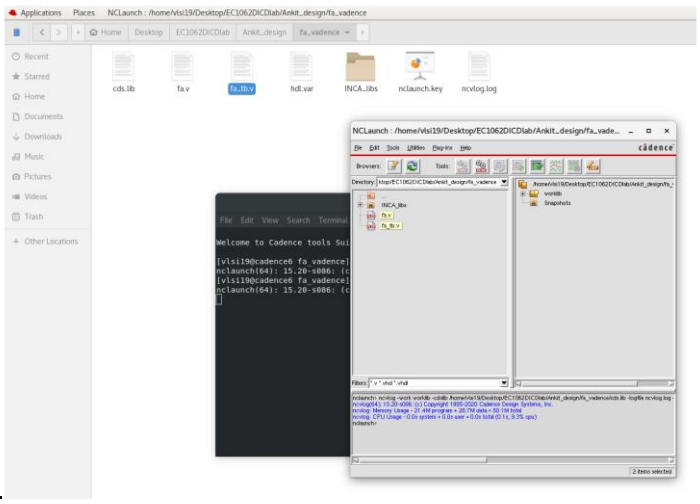
RTL Coding:

```
module full_adder (input A,nput B, input Cin, output Sum,output Carry);
   wire S1, C1, C2;
   assign S1 = A ^ B;
   assign Sum = S1 ^ Cin;
   assign C1 = A & B;
   assign C2 = S1 & Cin;
   assign Carry = C1 | C2;
endmodule
```

Sign-____________

Simulation Setup:

The simulation was carried out using Cadence NCLaunch. The input waveforms were provided, and output waveforms were observed


Applications
Places
NCLaunch : /home/vlsi19/Desktop/EC1062DICDlab/Ankit_design/fa_vadence
Home
Desktop
EC1062DICDlab
Ankit_design
fa_vadence
Recent
Starred
Home
Documents
Downloads
Music
Pictures
Videos
Trash
Other Locations
cds.lib
fa.v
fa_tb.v
hdl.var
INCA_libs
nclaunch.key
ncvlog.log
NCLaunch : /home/vlsi19/Desktop/EC1062DICDlab/Ankit_design/fa_vade...
File Edit Tools Utilities Plug-ins Help
cadence
Browsers:
Tools:
Directory: ktop/EC1062DICDlab/Ankit_design/fa_vadence
INCA_libs
fa.v
fa_tb.v
/home/vlsi19/Desktop/EC1062DICDlab/Ankit_design/fa_
worklib
Snapshots
Filters: *.v *.vhd *.vhdl
2 items selected
File Edit View Search Terminal

```
Welcome to Cadence tools Sui

[vlsi19@cadence6 fa_vadence]
nclaunch(64): 15.20-s086: (c
[vlsi19@cadence6 fa_vadence]
nclaunch(64): 15.20-s086: (c
```

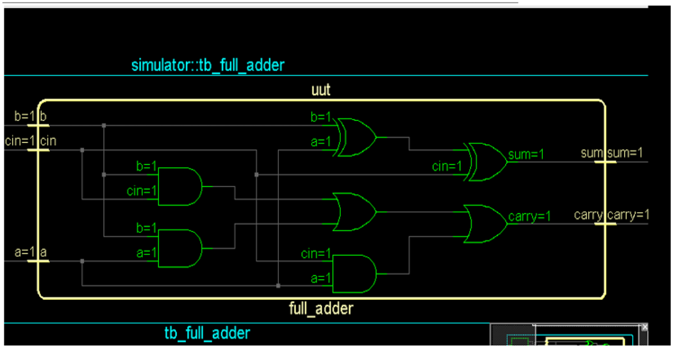
```
nclaunch> ncvlog -work worklib -cdslib /home/vlsi19/Desktop/EC1062DICDlab/Ankit_design/fa_vadence/cds.lib -logfile ncvlog.log -
ncvlog(64): 15.20-s086: (c) Copyright 1995-2020 Cadence Design Systems, Inc.
ncvlog: Memory Usage - 21.4M program + 28.7M data = 50.1M total
ncvlog: CPU Usage - 0.0s system + 0.0s user = 0.0s total (0.1s, 9.3% cpu)
nclaunch>
```

.

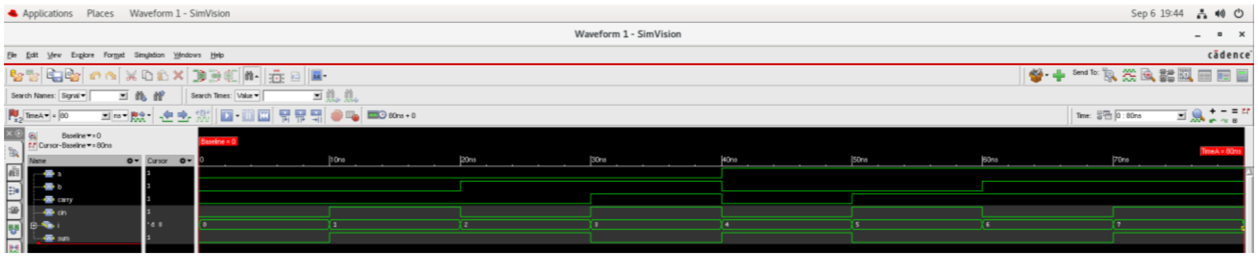
Experimental Results:

The expected results were observed from the simulation, and they matched the theoretical truth table for the circuit.

Schematic Diagram:



Functional Verification Waveform:



Observations:

The circuit worked as expected.

Design 3: Half Subtractor

Objective:

To design and implement a Half Subtractor circuit using basic logic gates.

Specification:

The Half Subtractor subtracts two 1-bit binary numbers (A and B) and produces a Difference (D) and Borrow (B).
Formulas:
Difference (D) = A B
Borrow (B) = B · ¬A

Truth Table:

A	B	Difference (D)	Borrow (B)
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Logic/ Block-level implementation:

The Half Subtractor is implemented using an XOR gate for Difference and an AND gate for Borrow, with the input B inverted.

RTL Coding:

```
module half_subtractor (
   input A,
   input B,
   output Difference,
   output Borrow
);
   assign Difference = A ^ B;
   assign Borrow = ~A & B;
endmodule
```

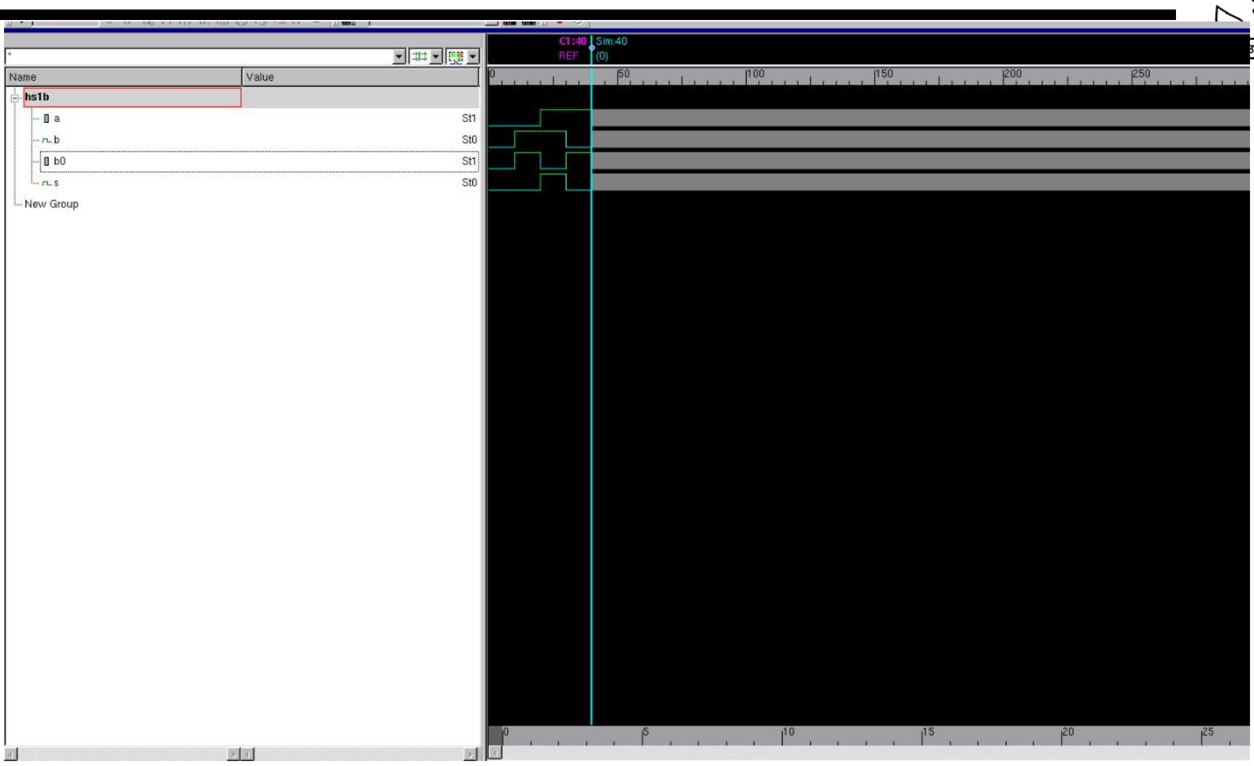
Simulation Setup:

The simulation was carried out using Cadence NCLaunch. The input waveforms were provided, and output waveforms were observed.

Experimental Results:

The expected results were observed from the simulation, and they matched the theoretical truth table for the circuit.

Sign-____________



Observations:

The circuit worked as expected.

Design 4: Full Subtractor

Objective:

To design and implement a Full Subtractor circuit using Half Subtractors.

Specification:

The Full Subtractor subtracts three 1-bit binary numbers (A, B, Bin) and produces a Difference (D) and Borrow (Bout).
Formulas:
Difference (D) = A □ B □ Bin
Borrow (Bout) = (¬A · B) + (¬(A □ B) · Bin)

Truth Table:

A	**B**	**Bin**	**Difference (D)**	**Borrow (Bout)**
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	0
1	1	1	1	1

Logic/ Block-level implementation:

The Full Subtractor is implemented using two Half Subtractors and an OR gate for final Borrow.

RTL Coding:

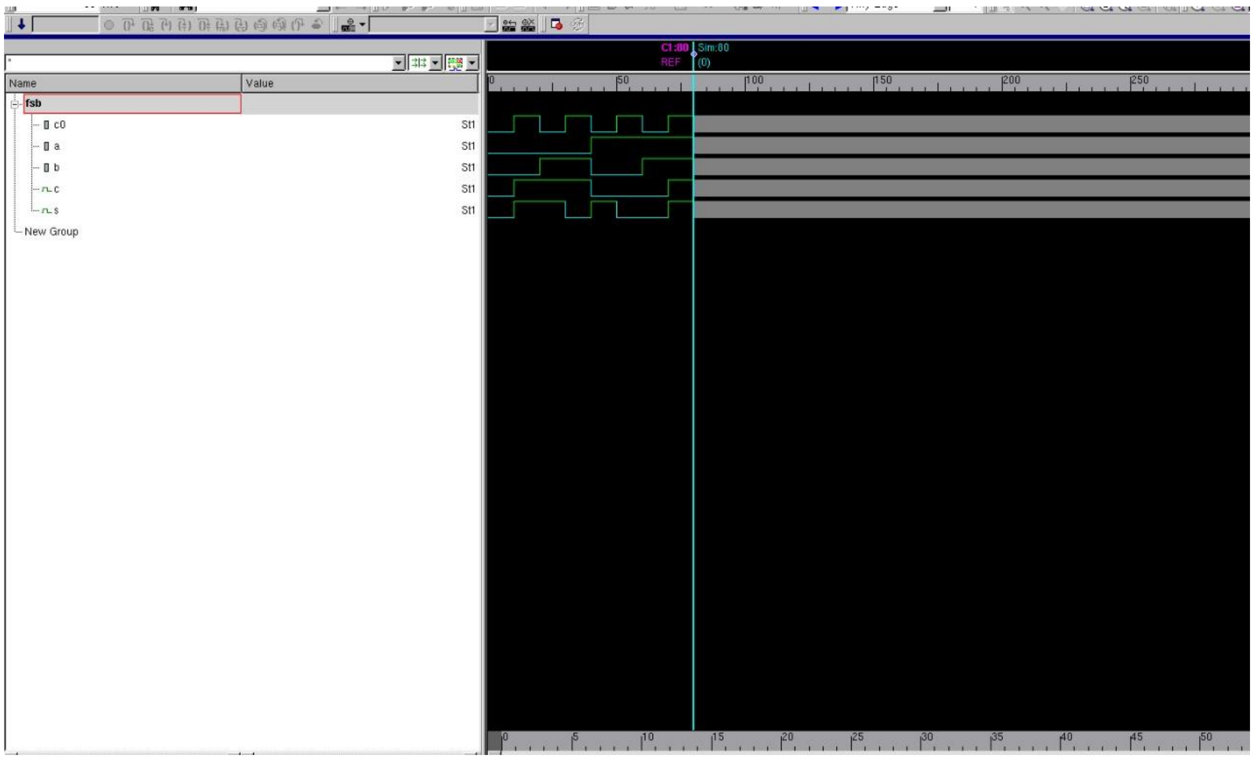
```
module full_subtractor (
    input A,
    input B,
    input Bin,
    output Difference,
    output Borrow
);
    wire D1, B1, B2;
    assign D1 = A ^ B;
    assign Difference = D1 ^ Bin;
    assign B1 = ~A & B;
    assign B2 = ~D1 & Bin;
    assign Borrow = B1 | B2;
endmodule
```

Simulation Setup:

The simulation was carried out using Cadence NCLaunch. The input waveforms were provided, and output waveforms were observed.

Experimental Results:

The expected results were observed from the simulation, and they matched the theoretical truth table for the circuit.



Observations:

The circuit worked as expected.

Sign-____________

Design 5: Multiplexer

Objective:

To design and implement a 2:1 Multiplexer circuit using logic gates.

Specification:

The 2:1 Multiplexer selects one of the two input signals (A, B) based on the select line (S).
Formula:
Y = (¬S · A) + (S · B)

Truth Table:

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

Logic/ Block-level implementation:

The Multiplexer is implemented using AND, OR, and NOT gates.

RTL Coding:

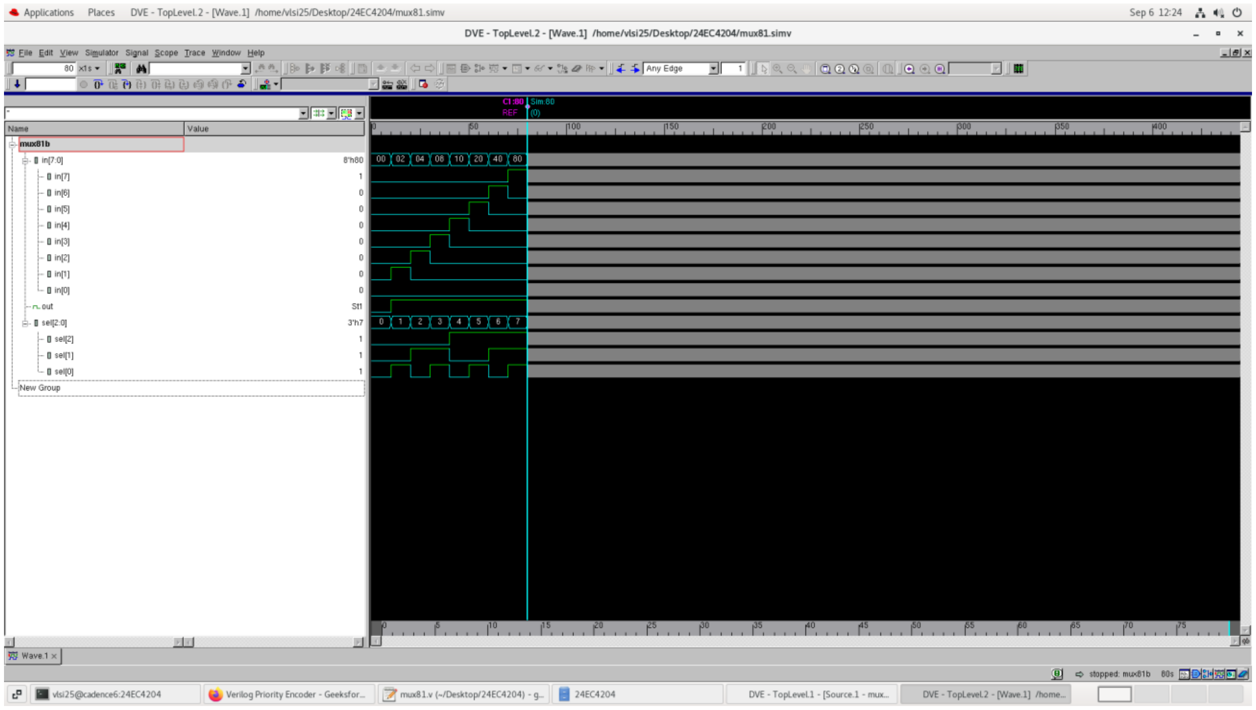
```
module multiplexer_2to1 (
    input A,
    input B,
    input S,
    output Y
);
    assign Y = (~S & A) | (S & B);
endmodule
```

Simulation Setup:

The simulation was carried out using Cadence NCLaunch. The input waveforms were provided, and output waveforms were observed.

Experimental Results:

The expected results were observed from the simulation, and they matched the theoretical truth table for the circuit.



Observations:

The circuit worked as expected. Propagation delay was observed for the combinational circuit.

Sign-____________

Design 6: De-Multiplexer

Objective:

To design and implement a 1:2 De-Multiplexer circuit using logic gates.

Specification:

The 1:2 De-Multiplexer sends an input signal (I) to one of the two outputs (Y0, Y1) based on the select line (S).
Formulas:
$Y0 = I \cdot \neg S$
$Y1 = I \cdot S$

Truth Table:

S	I	Y0	Y1
0	0	0	0
0	1	1	0
1	0	0	0
1	1	0	1

Logic/ Block-level implementation:

The De-Multiplexer is implemented using AND and NOT gates.

RTL Coding:

```
module demultiplexer_1to2 (
   input I,
   input S,
   output Y0,
   output Y1
);
   assign Y0 = I & ~S;
   assign Y1 = I & S;
endmodule
```

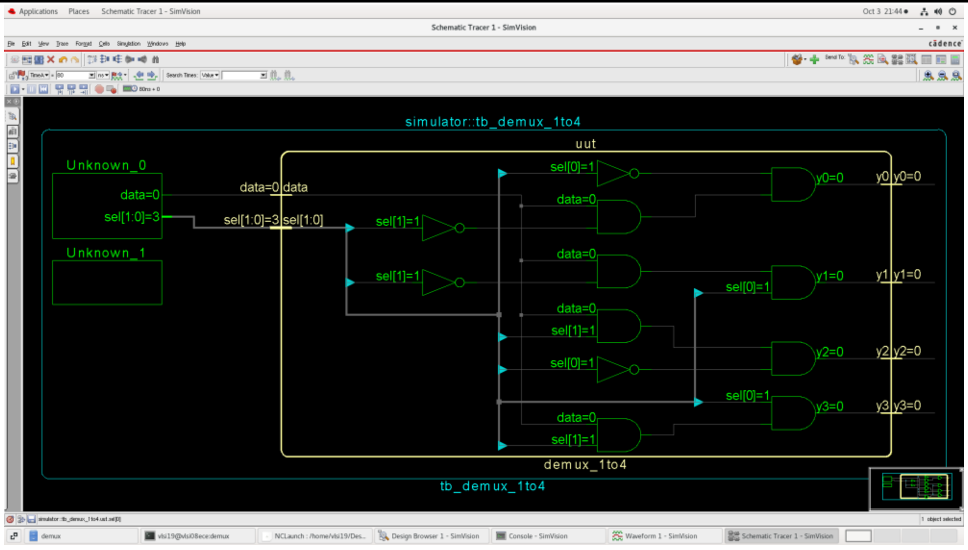
Simulation Setup:

The simulation was carried out using Cadence NCLaunch. The input waveforms were provided, and output waveforms were observed.

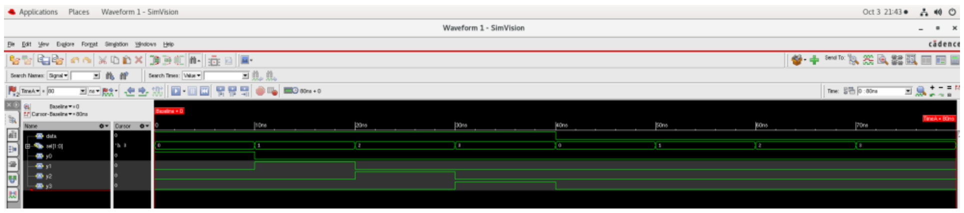
Experimental Results:

The expected results were observed from the simulation, and they matched the theoretical truth table for the circuit.

Schematic Diagram:



Functional Verification Waveform:



Observations:

The circuit worked as expected. Propagation delay was observed for the combinational circuit.

Sign-____________

Design 7: Decoder

Objective:

To design and implement a 2:4 Decoder circuit using logic gates.

Specification:

The 2:4 Decoder takes a 2-bit input and decodes it to activate one of the four output lines.
Formulas:
$Y0 = \neg A1 \cdot \neg A0$
$Y1 = \neg A1 \cdot A0$
$Y2 = A1 \cdot \neg A0$
$Y3 = A1 \cdot A0$

Truth Table:

A1	**A0**	**Y3**	**Y2**	**Y1**	**Y0**
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

Logic/ Block-level implementation:

The Decoder is implemented using AND and NOT gates.

RTL Coding:

```
module decoder_2to4 (
   input [1:0] A,
   output [3:0] Y
);
   assign Y[0] = ~A[1] & ~A[0];
   assign Y[1] = ~A[1] & A[0];
   assign Y[2] = A[1] & ~A[0];
   assign Y[3] = A[1] & A[0];
endmodule
```

Simulation Setup:

The simulation was carried out using Cadence NCLaunch. The input waveforms were provided, and output waveforms were observed.

Experimental Results:

The expected results were observed from the simulation, and they matched the theoretical truth table for the circuit.

Observations:

The circuit worked as expected.

Sign-____________

Design 8: Priority Encoder

Objective:

To design and implement a 4:2 Encoder circuit using logic gates.

Specification:

The 4:2 Encoder takes four input signals and encodes them to produce a 2-bit output.
Formulas:
Y1 = A2 + A3
Y0 = A1 + A3

Truth Table:

A3	A2	A1	A0	Y1	Y0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

RTL Coding:

```
module priority_encoder (
   input [3:0] in,  // 4 input lines
   output reg [1:0] out  // 2 output lines
);
always @(*) begin
   casez (in)
      4'b0001: out = 2'b00;  // Lowest priority
      4'b001?: out = 2'b01;
      4'b01??: out = 2'b10;
      4'b1???: out = 2'b11;  // Highest priority
      default: out = 2'b00;  // Default case
   endcase
end
endmodule
```

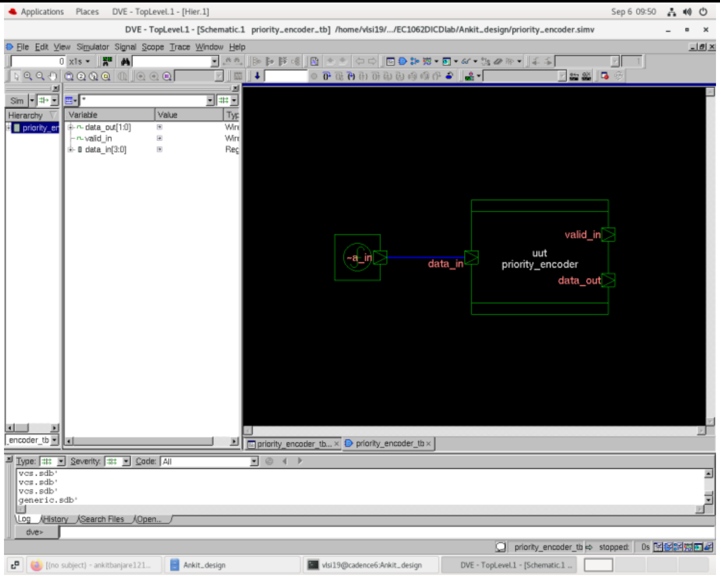
Simulation Setup:

The simulation was carried out using Cadence NCLaunch. The input waveforms were provided, and output waveforms were observed.

Experimental Results:

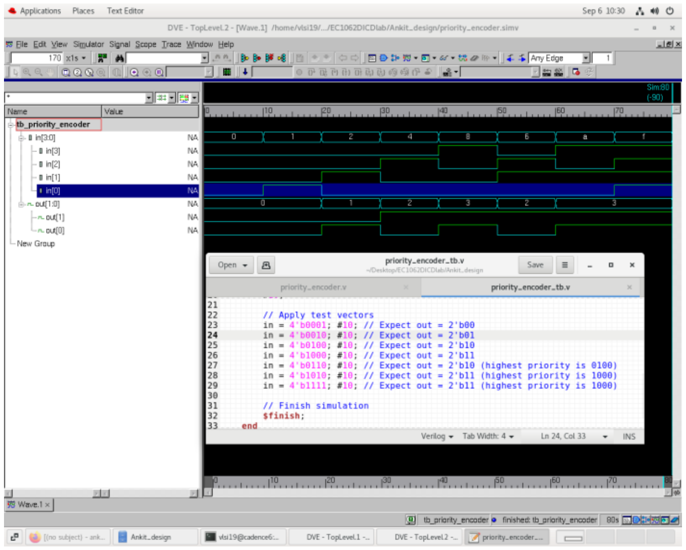
The expected results were observed from the simulation, and they matched the theoretical truth table for the circuit.

Schematic Diagram:


DVE - TopLevel.1 - [Schematic.1 priority_encoder_tb] /home/vlsi19/.../EC1062DICDlab/Ankit_design/priority_encoder.simv
data_out[1:0]
valid_in
data_in[3:0]
~a_in
data_in
valid_in
uut
priority_encoder
data_out

```
vcs.sdb'
vcs.sdb'
vcs.sdb'
generic.sdb'
```

Functional Verification Waveform:


DVE - TopLevel.2 - [Wave.1] /home/vlsi19/.../EC1062DICDlab/Ankit_design/priority_encoder.simv
tb_priority_encoder
in[3:0]
in[3]
in[2]
in[1]
in[0]
out[1:0]
out[1]
out[0]
New Group
priority_encoder_tb.v
~/Desktop/EC1062DICDlab/Ankit_design
priority_encoder.v
priority_encoder_tb.v
finished: tb_priority_encoder

```verilog
        // Apply test vectors
        in = 4'b0001; #10; // Expect out = 2'b00
        in = 4'b0010; #10; // Expect out = 2'b01
        in = 4'b0100; #10; // Expect out = 2'b10
        in = 4'b1000; #10; // Expect out = 2'b11
        in = 4'b0110; #10; // Expect out = 2'b10 (highest priority is 0100)
        in = 4'b1010; #10; // Expect out = 2'b11 (highest priority is 1000)
        in = 4'b1111; #10; // Expect out = 2'b11 (highest priority is 1000)

        // Finish simulation
        $finish;
    end
```

Observations:

The circuit worked as expected. Propagation delay was observed for the combinational circuit.

Sign-______________