Worksheet of Experiment

Date of Experiment	10 sep, 2024
Name	Akshada Jotiram Ghorpade.
Registration Number	12302956.

Title of Experiment: Develop RTL designs in verilog to implement multiplexer and demultiplexer and verify their functionality using cadence NCSIM.

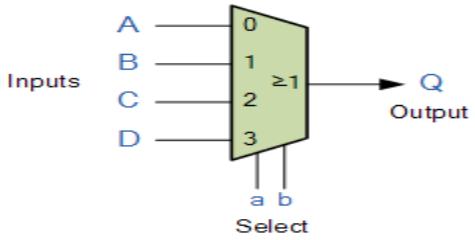
Name of Language: Verilog.

Name of Software: Cadence NCSIM, simvision.

Theory of Experiment:

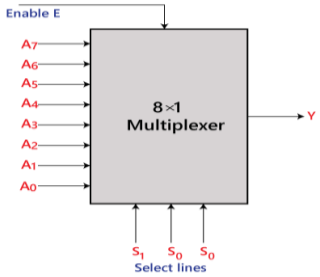
Multiplexer:

- A multiplexer is a combinational circuit that has 2^n input lines and a single output line. Simply, the multiplexer is a multi-input and single-output combinational circuit.
- The binary information is received from the input lines and directed to the output line. On the basis of the values of the selection lines, one of these data inputs will be connected to the output.



- **8 to 1 Multiplexer**
  - In the 8 to 1 multiplexer, there are total eight inputs, i.e., A_0, A_1, A_2, A_3, A_4, A_5, A_6, and A_7, 3 selection lines, i.e., S_0, $S_1$and S_2 and single output, i.e., Y.
  - On the basis of the combination of inputs that are present at the selection lines S^0, $S^{1,}$ and S_2, one of these 8 inputs are connected to the output.

Block Diagram:



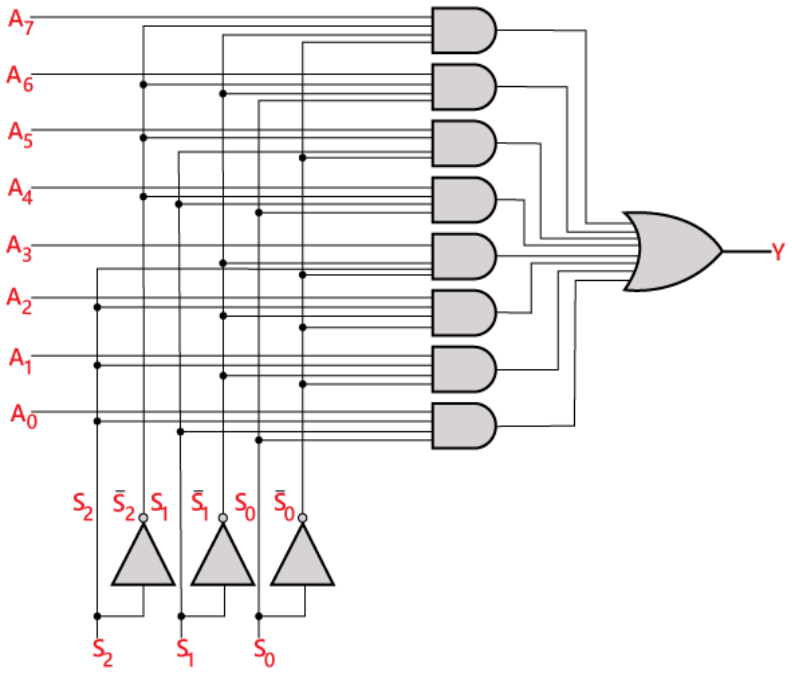
Truth table:

INPUTS			Output
S_2	S_1	S_0	Y
0	0	0	A_0
0	0	1	A_1
0	1	0	A_2
0	1	1	A_3
1	0	0	A_4
1	0	1	A_5
1	1	0	A_6
1	1	1	A_7

Boolean expression:

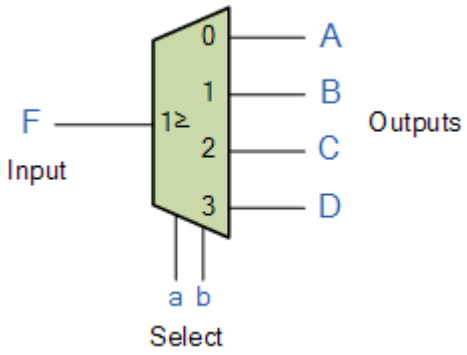
$$Y=S_0'.S_1'.S_2'.A_0+S_0.S_1'.S_2'.A_1+S_0'.S_1.S_2'.A_2+S_0.S_1.S_2'.A_3+$$
$$S_0'.S_1'.S_2\ A_4+S_0.S_1'.S_2\ A_5+S_0'.S_1.S_2\ .A_6+S_0.S_1.S_3.A_7$$

Circuit Diagram:



Demultiplexer:

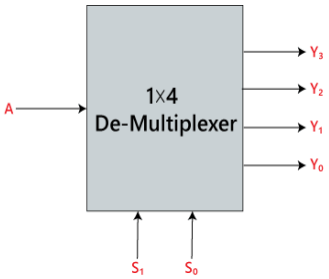
- A De-multiplexer is a combinational circuit that has only 1 input line and 2^N output lines. Simply, the multiplexer is a single-input and multi-output combinational circuit.
- The information is received from the single input lines and directed to the output line. On the basis of the values of the selection lines, the input will be connected to one of these outputs.



1×4 De-multiplexer:

- In 1 to 4 De-multiplexer, there are total of four outputs, i.e., Y_0, Y_1, Y_2, and Y_3, 2 selection lines, i.e., S_0 and S_1 and single input, i.e., A.
- On the basis of the combination of inputs which are present at the selection lines S_0 and S_1, the input be connected to one of the outputs.

Block Diagram:



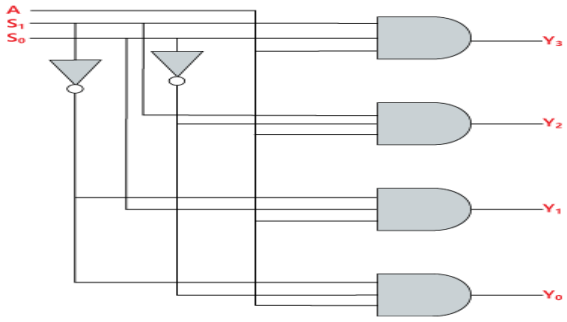
Truth table

INPUTS		Output			
S_1	S_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	A
0	1	0	0	A	0
1	0	0	A	0	0
1	1	A	0	0	0

Boolean expression:

$$Y_0=S_1' S_0' A$$
$$Y_1=S_1' S_0 A$$
$$Y_2=S_1 S_0' A$$
$$Y_3=S_1 S_0 A$$

Circuit Diagram:



Program and Stimulus (Testbench) program:

PROGRAM (Multiplexer):

```
module mux8(i, s, y);
input [7:0] i;
input [2:0] s;
output y;
assign y=i[0]&(~s[2])&(~s[1])&(~s[0])|
         i[1]&(~s[2])&(~s[1])&(s[0])|
         i[2]&(~s[2])&(s[1])&(~s[0])|
         i[3]&(~s[2])&(s[1])&(s[0])|
         i[4]&(s[2])&(~s[1])&(~s[0])|
         i[5]&(s[2])&(~s[1])&(s[0])|
         i[6]&(s[2])&(s[1])&(~s[0])|
         i[7]&(s[2])&(s[1])&(s[0]);
endmodule
```

TESTBENCH (Multiplexer):

```
module mux8tb;
reg [7:0] i;
reg [2:0] s;
wire y;
mux8 dut(i, s, y);
initial
begin
i=8’ b10000000;
s=3’ b000;
#10 s=3’ b001;
#10 s=3’ b010;
#10 s=3’ b011;
#10 s=3’ b100;
#10 s=3’ b101;
#10 s=3’ b110;
#10 s=3’ b111;
#10 $stop;
end
endmodule
```

PROGRAM (Demultiplexer):

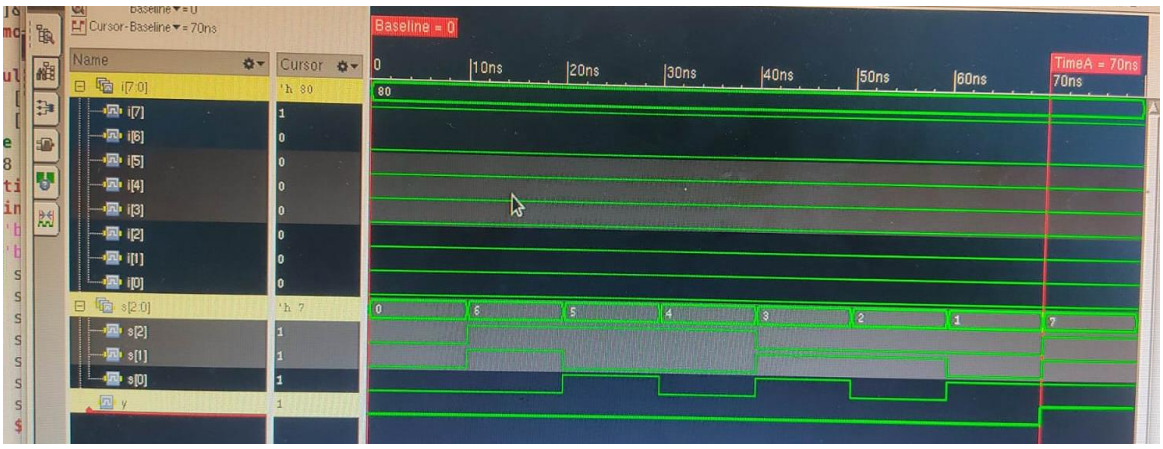
```
module demux(i,s1,s0,y0,y1,y2,y3);
input i,s1,s0;
output y0,y1,y2,y3;
wire w1,w2;
not(w1,s1);
not(w2,s0);
and (y0,w1,w2,i);
and (y1,wl,s0,i);
and (y2,s1,w2,i);
and (y3,s1,s0,i);
endmodule
```

TESTBENCH (Demultiplexer):

```
module demuxtb;
reg i,s1,s0;
wire y0,y1,y2, y3;
 demux dut(i,s1,s0,y0,y1,y2,y3);
initial
begin
i=1;
s1=0;s0=0;
#10 s1=0;s0=1;
 #10 s1=1;s0=0;
#10 s1=1;s0=1;
#10 $stop;
end
endmodule
```

Simulation waveform:

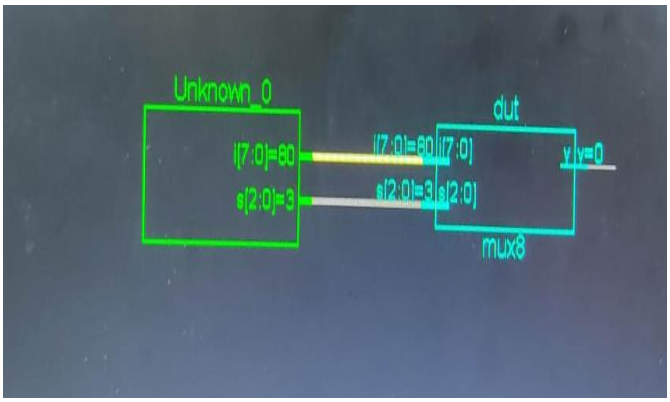
Multiplexer



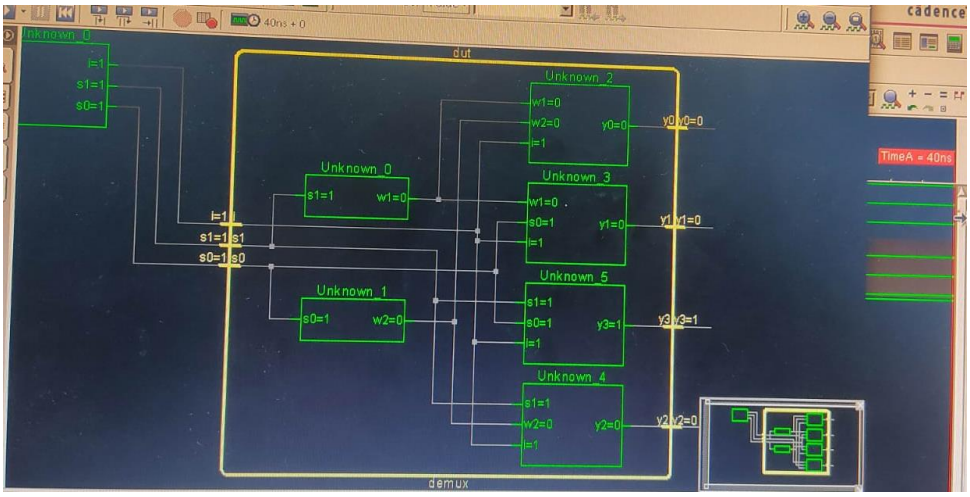
Demultiplexer

RTL Schematic:

Multiplexer



Demultiplexer



Learning Outcome:

1. Learn to write syntactically correct Verilog code for multiplexer and demultiplexer.

2. Create testbenches to simulate and verify the functionality of multiplexer and demultiplexer.

3. Apply techniques to optimize logic circuits for speed, area, and power efficiency.

4. Perform timing analysis to ensure circuits meet required timing constraints.

5. Analyze the performance of multiplexer and demultiplexer and circuits in terms of timing, power, and area using simulation results.