SIMULATION AND IMPLEMENTATION OF COMBINATIONAL LOGIC CIRCUITS

AIM:

To simulate the following circuits using VIVADO 2023.2.

1) Encoder
2) Decoder
3) Multiplexer
4) Demultiplexer
5) Magnitude Comparator

APPARATUS REQUIRED:

VIVADO 2023.2

PROCEDURE:

STEP:1 Launch the Vivado 2023.2 software.

STEP:2 Click on “create project ” from the starting page of vivado .

STEP:3 Choose the design entry method:RTL(verilog/VHDL)

STEP:4 Crete design source and give name to it and click finish.

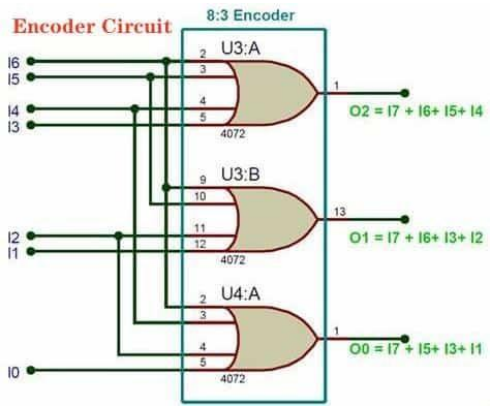
STEP:5 Write the verilog code and check the syntax.

STEP:6 Click “run simulation” in the navigator window and click “Run behavioral simulation”.

STEP:7 Verify the output in the simulation window.

ENCODER

LOGIC DIAGRAM

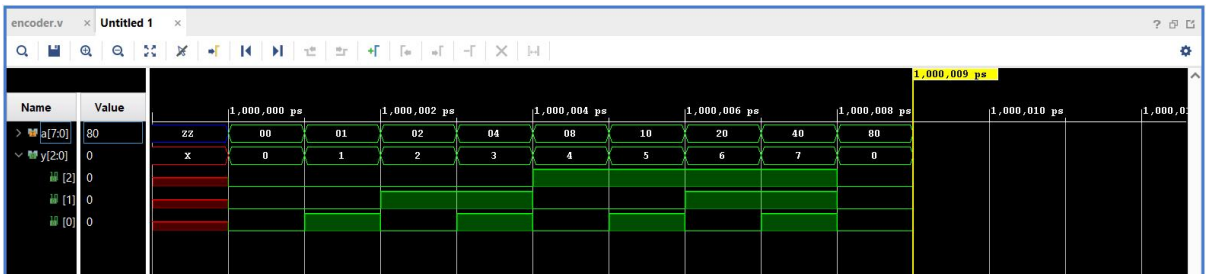


INPUTS								OUTPUTS		
Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0	A_2	A_1	A_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

VERILOG CODE

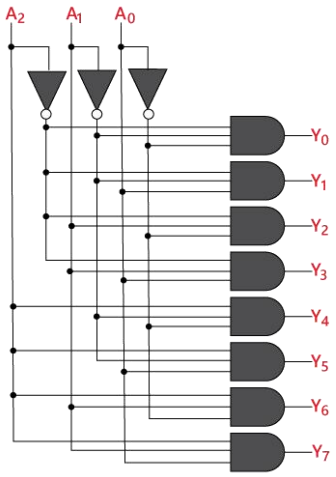
```
module encoder(a,y);
input [7:0]a;
output[2:0]y;
or(y[2],a[6],a[5],a[4],a[3]);
or(y[1],a[6],a[5],a[2],a[1]);
or(y[0],a[6],a[4],a[2],a[0]);
endmodule
```

OUTPUT WAVEFORM



DECODER

LOGIC DIAGRAM



A2	A1	A0	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

VERILOG CODE

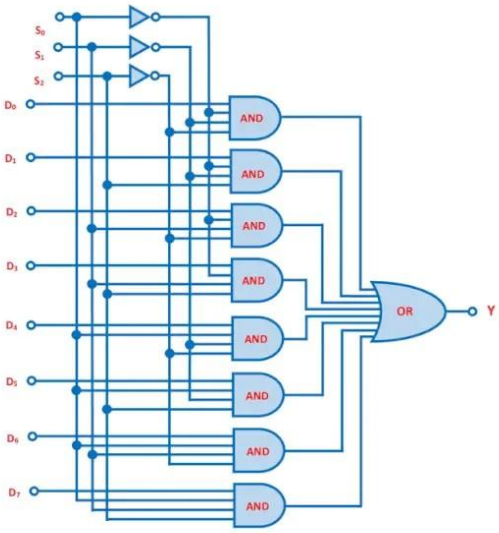
```
module decoder1(a,y);
input [2:0]a;
output[7:0]y;
and(y[0],~a[2],~a[1],~a[0]);
and(y[1],~a[2],~a[1],a[0]);
```

```
and(y[2],~a[2],a[1],~a[0]);
and(y[3],~a[2],a[1],a[0]);
and(y[4],a[2],~a[1],~a[0]);
and(y[5],a[2],~a[1],a[0]);
and(y[6],a[2],a[1],~a[0]);
and(y[7],a[2],a[1],a[0]);
endmodule
```

OUTPUT WAVEFORM

MULTIPLEXER

LOGIC DIAGRAM

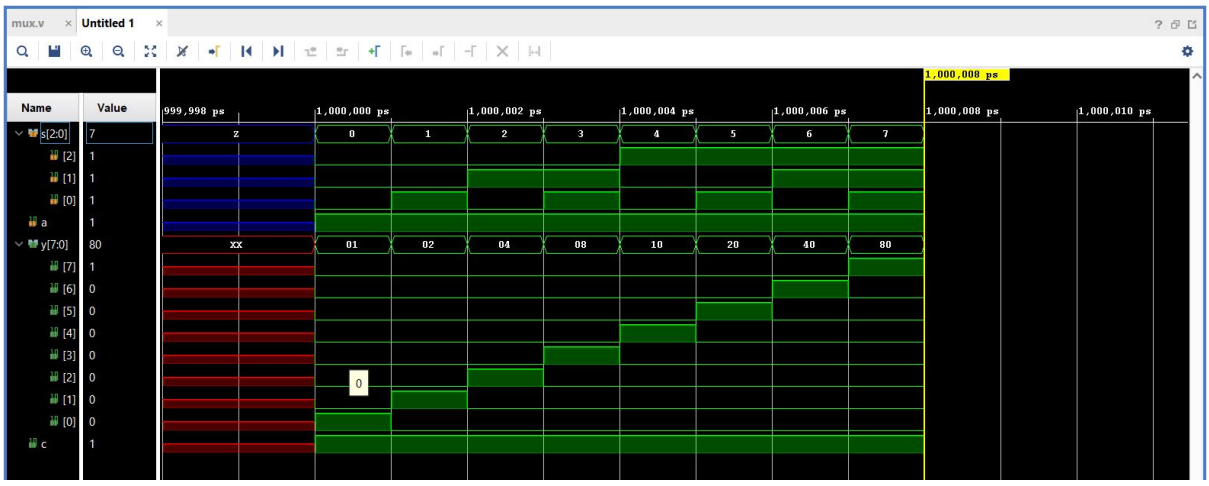


S2	S1	S0	O0	O1	O2	O3	O4	O5	O6	O7
0	0	0	I	0	0	0	0	0	0	0
0	0	1	0	I	0	0	0	0	0	0
0	1	0	0	0	I	0	0	0	0	0
0	1	1	0	0	0	I	0	0	0	0
1	0	0	0	0	0	0	I	0	0	0
1	0	1	0	0	0	0	0	I	0	0
1	1	0	0	0	0	0	0	0	I	0
1	1	1	0	0	0	0	0	0	0	I

VERILOG CODE

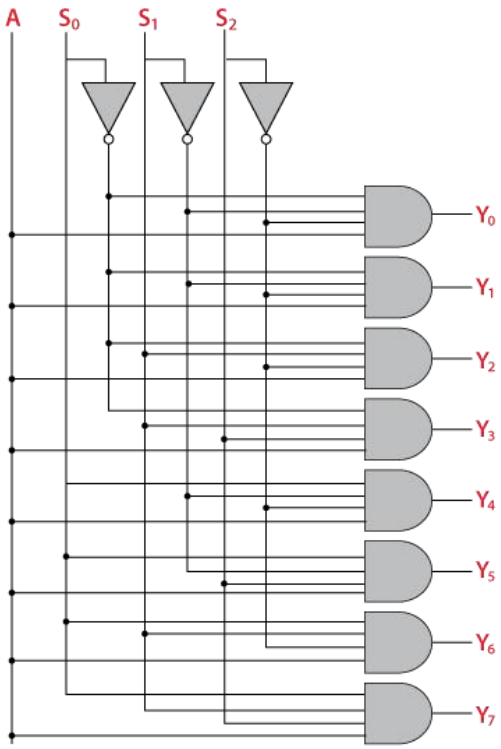
```
module mux(s,c,a);
input [2:0]s;
input [7:0]a;
wire [7:0]w;
output c;
and(w[0],a[0],~s[2],~s[1],~s[0]);
and(w[1],a[1],~s[2],~s[1],s[0]);
and(w[2],a[2],~s[2],s[1],~s[0]);
and(w[3],a[3],~s[2],s[1],s[0]);
and(w[4],a[4],s[2],~s[1],~s[0]);
and(w[5],a[5],s[2],~s[1],s[0]);
and(w[6],a[6],s[2],s[1],~s[0]);
and(w[7],a[7],s[2],s[1],s[0]);
or (c,w[0],w[1],w[2],w[3],w[4],w[5],w[6],w[7]);
endmodule
```

OUTPUT WAVEFORM



DEMULTIPLEXER

LOGIC DIAGRAM



INPUTS			Output							
S_2	S_1	S_0	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0	0	0	A
0	0	1	0	0	0	0	0	0	A	0
0	1	0	0	0	0	0	0	A	0	0
0	1	1	0	0	0	0	A	0	0	0
1	0	0	0	0	0	A	0	0	0	0
1	0	1	0	0	A	0	0	0	0	0
1	1	0	0	A	0	0	0	0	0	0
1	1	1	A	0	0	0	0	0	0	0

VERILOG CODE

```
module demux_8(s,a,y);
input [2:0]s;
input a;
output [7:0]y;
and(y[0],a,~s[2],~s[1],~s[0]);
and(y[1],a,~s[2],~s[1],s[0]);
and(y[2],a,~s[2],s[1],~s[0]);
and(y[3],a,~s[2],s[1],s[0]);
and(y[4],a,s[2],~s[1],~s[0]);
and(y[5],a,s[2],~s[1],s[0]);
and(y[6],a,s[2],s[1],~s[0]);
and(y[7],a,s[2],s[1],s[0]);
endmodule
```

OUTPUT WAVEFORM

MAGNITUDE COMPARATOR

LOGIC DIAGRAM

Inputs		Outputs		
A	B	A<B	A=B	A>B
00001100	00001100	0	1	0
00001010	00010001	1	0	0
00001111	00000101	0	0	1
00011000	00011000	0	1	0

VERILOG CODE

```
module comparator(a,b,eq,lt,gt);
input [3:0] a,b;
output reg eq,lt,gt;
always @(a,b)
begin
 if (a==b)
 begin
  eq = 1'b1;
  lt = 1'b0;
  gt = 1'b0;
 end
 else if (a>b)
 begin
  eq = 1'b0;
```

```
 lt = 1'b0;
 gt = 1'b1;
end
else
begin
 eq = 1'b0;
 lt = 1'b1;
 gt = 1'b0;
end
end
endmodule
```

OUTPUT WAVEFORM

RESULT

Thus the simulation and implementation of combinational logic circuits is done and outputs are verified successfully.