SIMULATION AND IMPLEMENTATION OF COMBINATIONAL LOGIC CIRCUITS

AIM

To simulate and synthesis ENCODER, DECODER, MULTIPLEXER, DEMULTIPLEXER, MAGNITUDE COMPARATOR using Xilinx ISE

PROCEDURE

STEP:1 Start the Xilinx navigator, Select and Name the New project.

STEP:2 Select the device family, device, package and speed.

STEP:3 Select new source in the New Project and select Verilog Module as the Source type.

STEP:4 Type the File Name and Click Next and then finish button. Type the code and save it.

STEP:5 Select the Behavioral Simulation in the Source Window and click the check syntax.

STEP:6 Click the simulation to simulate the program and give the inputs and verify the outputs as per the truth table.

STEP:7 Select the Implementation in the Sources Window and select the required file in the Processes Window.

STEP:8 Select Check Syntax from the Synthesize XST Process. Double Click in the FloorplanArea/IO/Logic-Post Synthesis process in the User Constraints process group. UCF(User constraint File) is obtained.

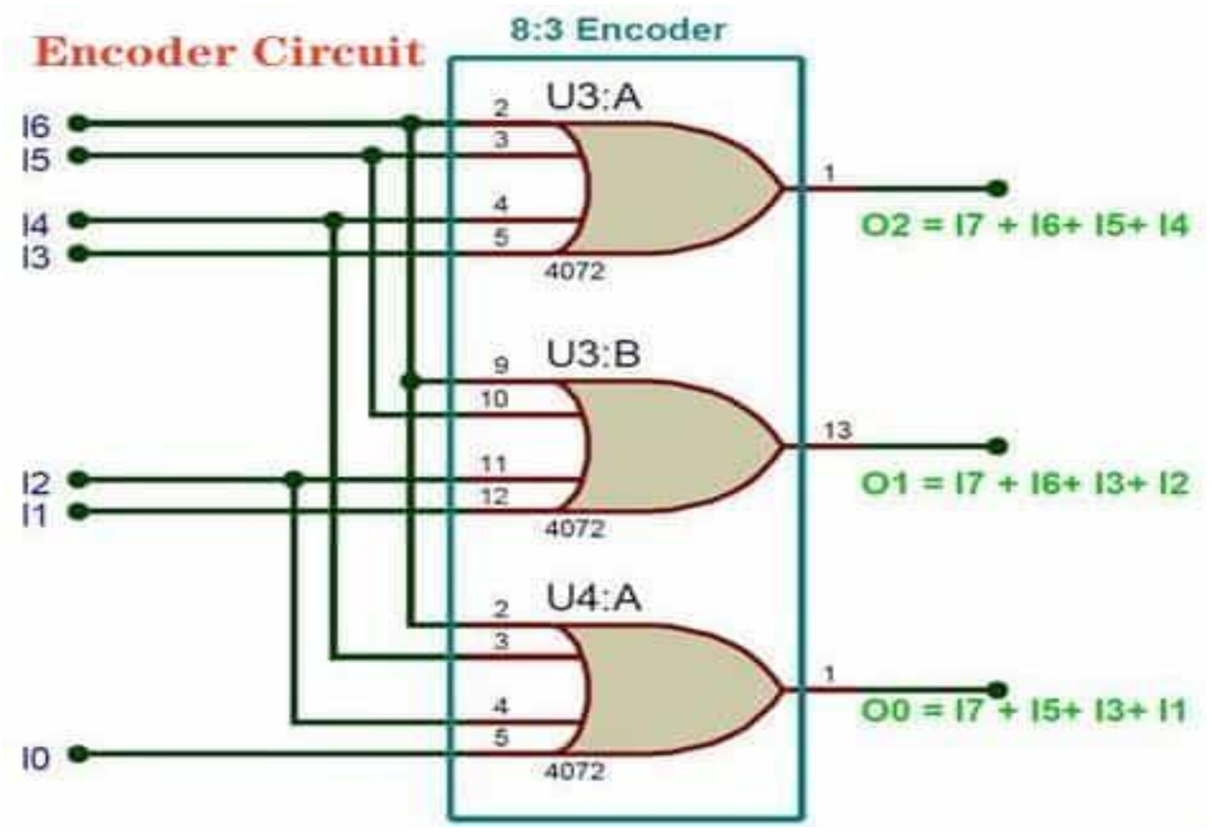
STEP:9 In the Design Object List Window, enter the pin location for each pin in the Loc column Select save from the File menu.

STEP:10 Double click on the Implement Design and double click on the Generate Programming File to create a bitstream of the design.(.v) file is converted into .bit file here.

STEP:11 On the board, by giving required input, the LEDs starts to glow light, indicating the output.

LOGIC DIAGRAM

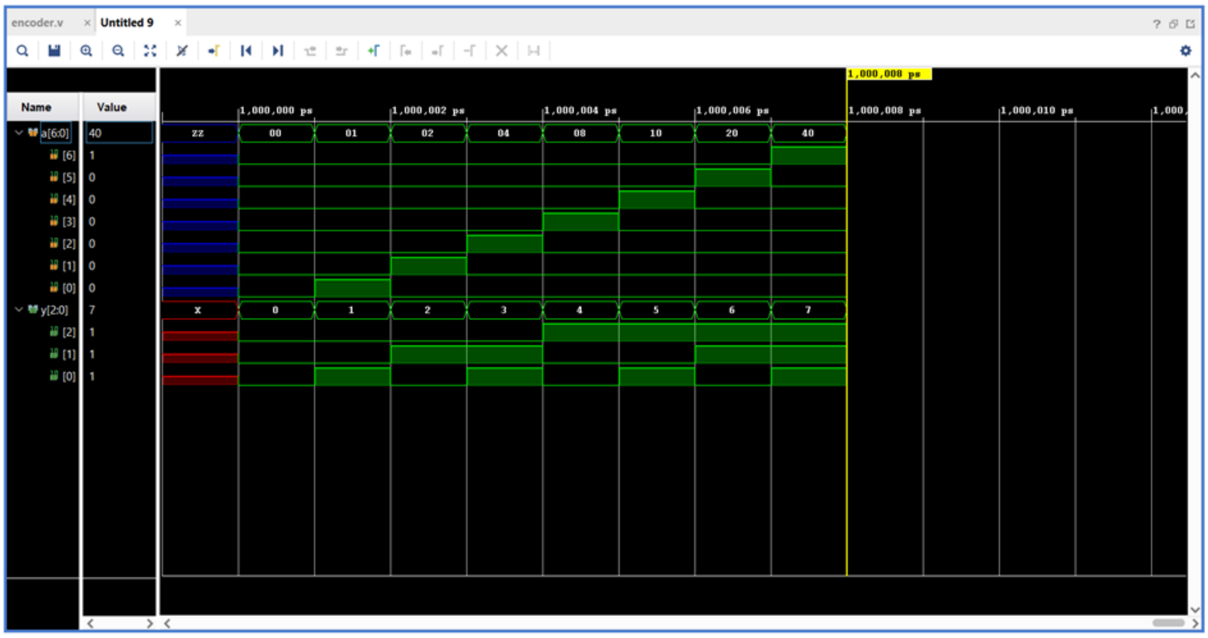
\\ENCODER\\



VERILOG CODE

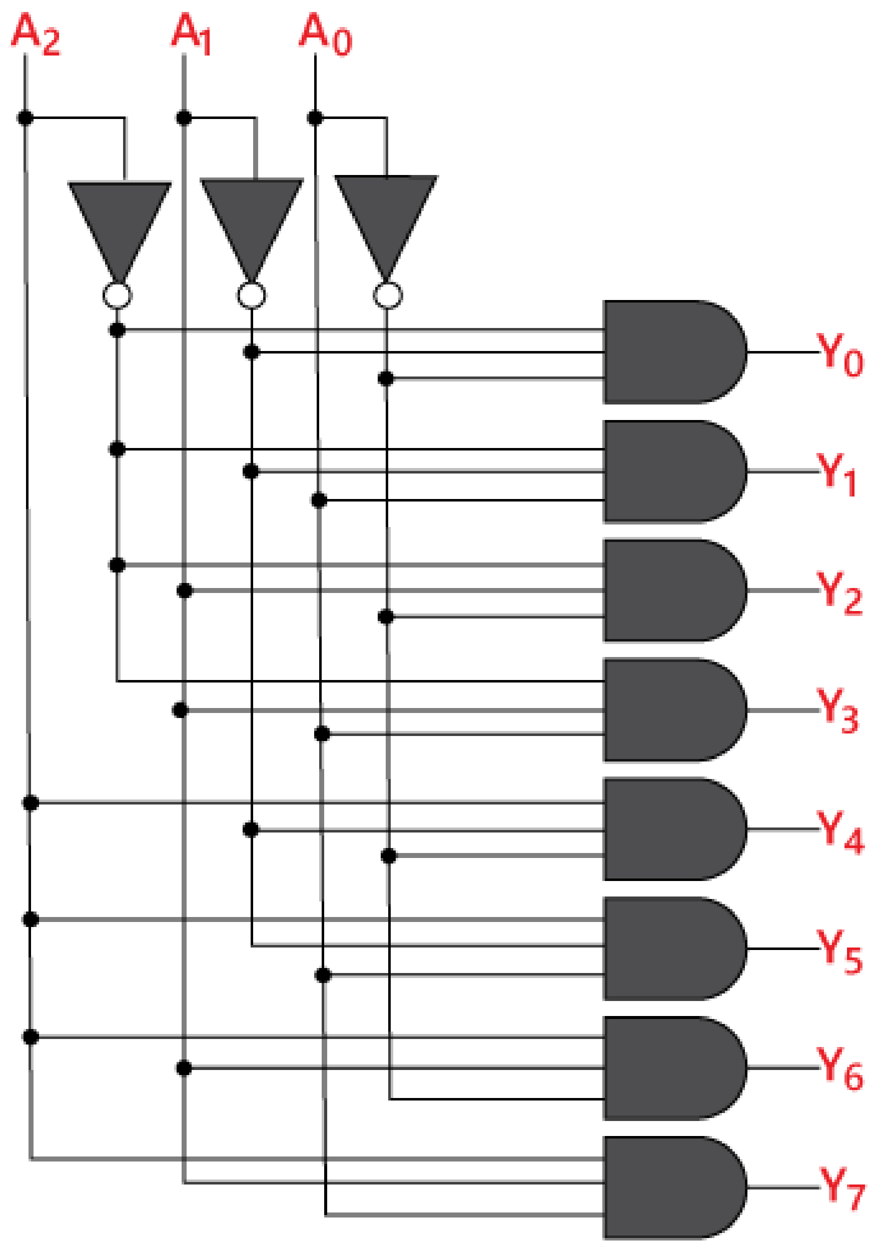
```
module encoder(a,y);

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



LOGIC DIAGRAM

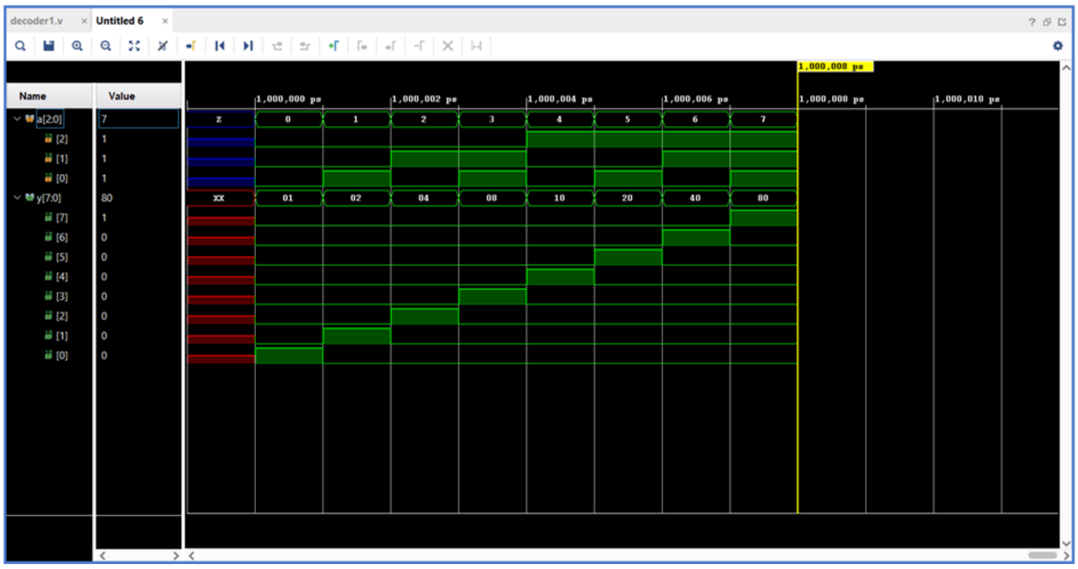
\\DECODER\\



VERILOG CODE

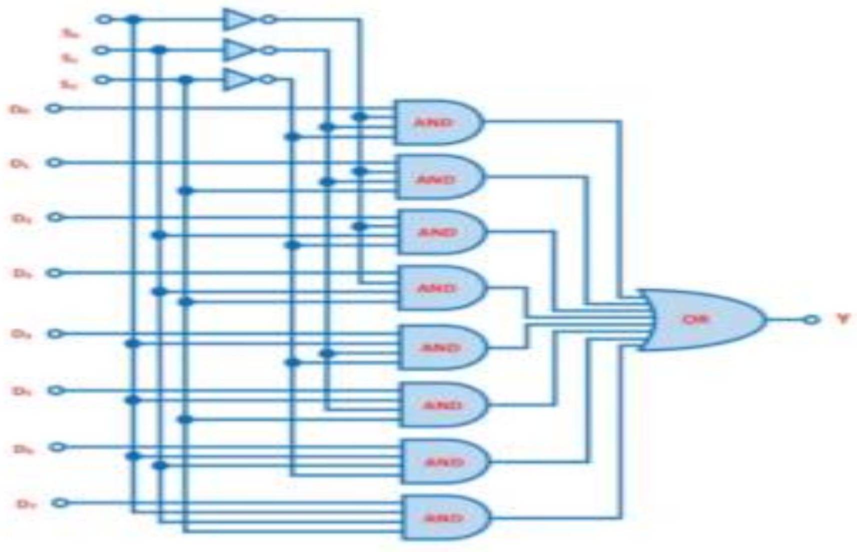
```
module decoder1(a,y);

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



LOGIC DIAGRAM

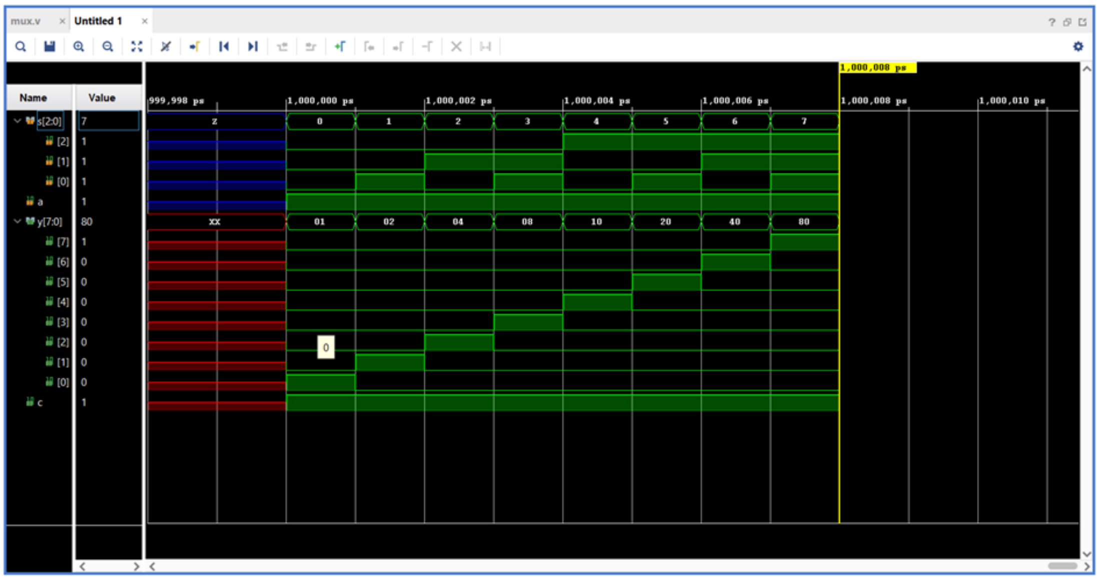
\\ MULTIPLEXER \\



VERILOG CODE

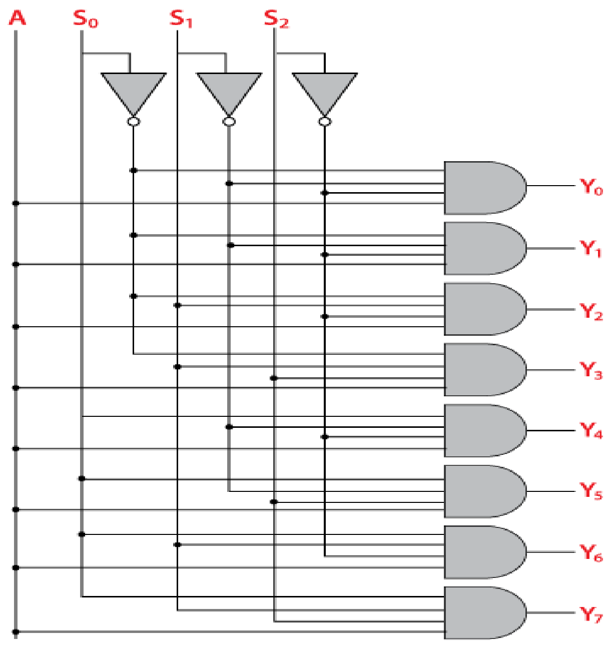
```
module mux(s,c,a);
input [2:0]s;
input [7:0]a;
wire [7:0]w;
output c;
and(w[0],a[0],~s[2],~s[1],~s[0]);
and(w[1],a[1],~s[2],~s[1],s[0]);
and(w[2],a[2],~s[2],s[1],~s[0]);
and(w[3],a[3],~s[2],s[1],s[0]);
and(w[4],a[4],s[2],~s[1],~s[0]);
and(w[5],a[5],s[2],~s[1],s[0]);
and(w[6],a[6],s[2],s[1],~s[0]);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



LOGIC DIAGRAM

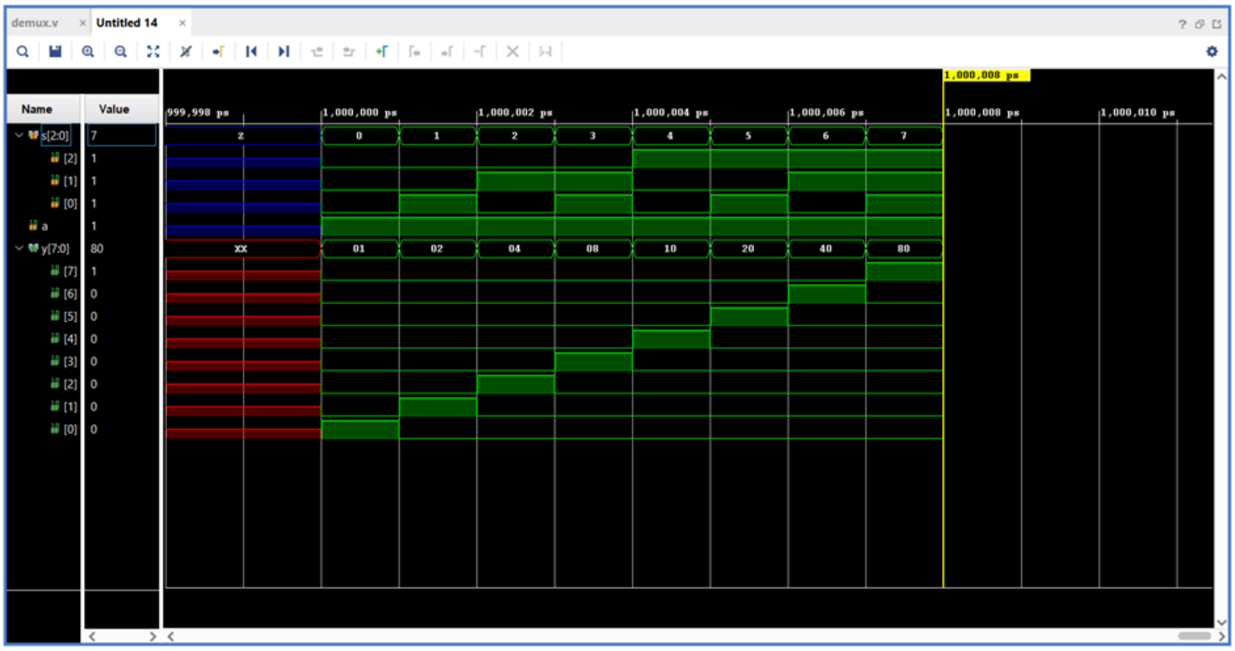
\\DEMULTIPLEXER\\



VERILOG CODE

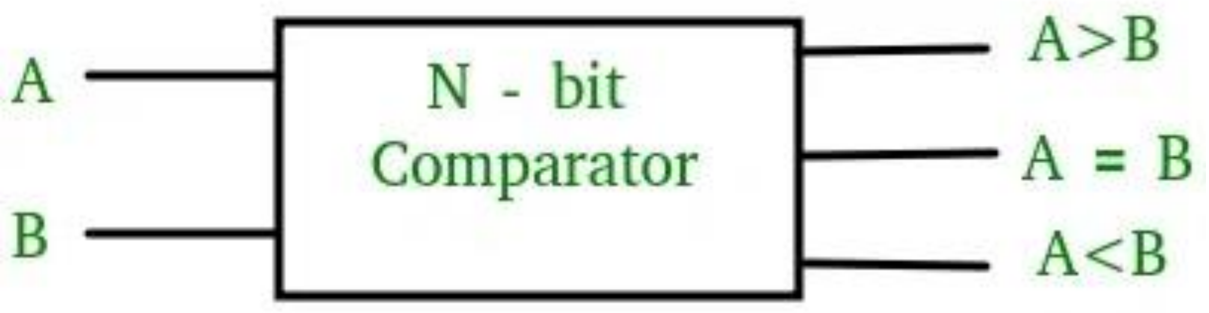
```
module demux_8(s,a,y);

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



LOGIC DIAGRAM

\\MAGNITUDE COMPARATOR\\

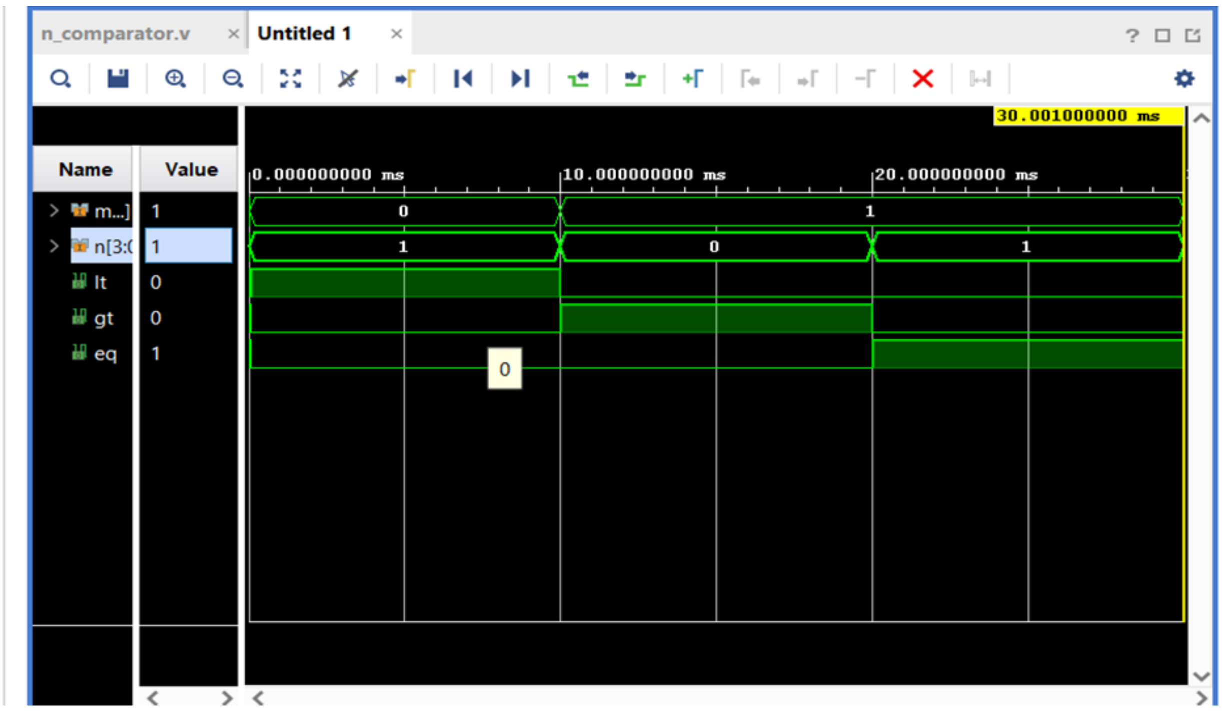


VERILOG CODE

```
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



RESULT

Thus The Simulate And Synthesis Encoder, Decoder, Multiplexer, Demultiplexer, Magnitude Comparator Using Xilinx Ise Are Verified Successfully