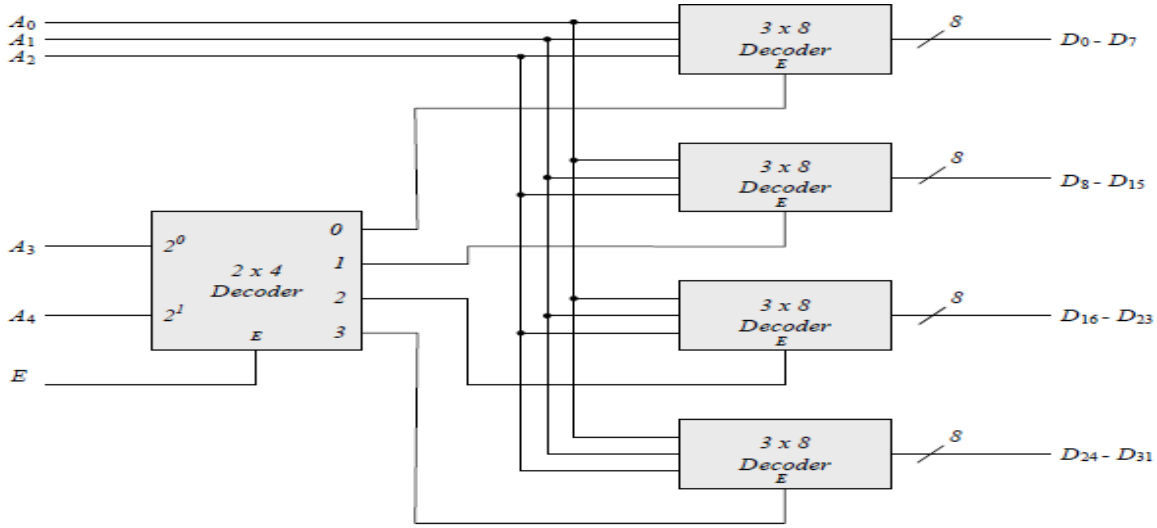
Sheet (ch4)

Q1-Construct a 5-to-32-line decoder with four 3-to-8-line decoders with enable and a 2-to-4-line decoder.use block diagrams for the components.

ANS

(a) Block diagram:-



(b) Code:-

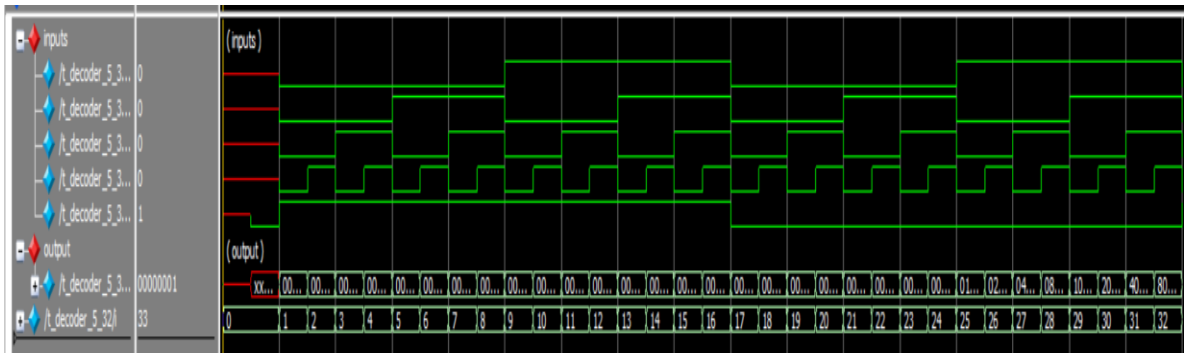
```
module decoder_2x4 ( output [3:0] d,
                     input A,B,E);

wire An,Bn;

not (An,A);
not (Bn,B);
and (d[0],An,Bn,E);
and (d[1],An,B,E);
and (d[2],A,Bn,E);
and (d[3],A,B,E);

endmodule


module decoder_3x8 ( output [7:0] d,
                     input x,y,z,E);

wire xn,yn,zn;

not (xn,x);
not (yn,y);
not (zn,z);
```

```
and (d[0],zn,yn,xn,E);
and (d[1],z,yn,xn,E);
and (d[2],zn,y,xn,E);
and (d[3],z,y,xn,E);
and (d[4],zn,yn,x,E);
and (d[5],z,yn,x,E);
and (d[6],zn,y,x,E);
and (d[7],z,y,x,E);

endmodule


module decoder_5x32 ( output [31:0] D,
                      input [4:0] A,
                      input E);
wire [3:0] c;

decoder_2x4 m(c[3:0],A[4],A[3],E);
decoder_3x8 w1(D[7:0],A[2],A[1],A[0],c[0]);
decoder_3x8 w2(D[15:8],A[2],A[1],A[0],c[1]);
decoder_3x8 w3(D[23:16],A[2],A[1],A[0],c[2]);
decoder_3x8 w4(D[31:24],A[2],A[1],A[0],c[3]);

endmodule

`timescale 1ns/1ps

module decoder_5x32_ts();

reg [4:0] A;
reg E;
wire [31:0] D2;
integer i = 0;

decoder_5x32 c(D2,A,E);

initial
begin

E = 1'b0;
A = 5'b0;

$monitor ("in : %d E : %b out : %b",A,E,D2);

#10 E = 1'b1;
for (i = 1 ; i <= 32 ; i = i + 1)
#10 A = A + 1;
end

endmodule
```

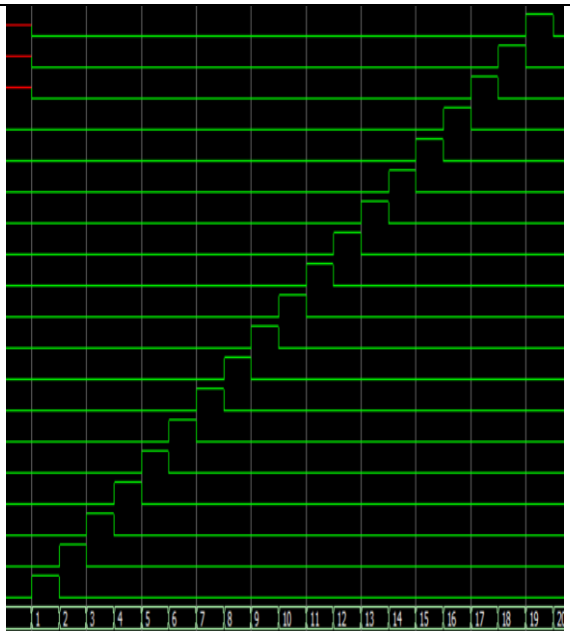
```
module decoder_2x4_22 ( output [3:0] d,
                        input A,B,E);

assign d[0] = E & (~A) & (~B);
assign d[1] = E & (~A) & (B);
assign d[2] = E & (A) & (~B);
assign d[3] = E & (A) & (B);

endmodule


module decoder_3x8_22 ( output [7:0] d,
                        input x,y,z,E);

assign d[0] = E & (~z) & (~y) & (~x);
assign d[1] = E & (z) & (~y) & (~x);
assign d[2] = E & (~z) & (y) & (~x);
assign d[3] = E & (z) & (y) & (~x);
assign d[4] = E & (~z) & (~y) & (x);
assign d[5] = E & (z) & (~y) & (x);
assign d[6] = E & (~z) & (y) & (x);
assign d[7] = E & (z) & (y) & (x);

endmodule


module decoder_5x32_22 ( output [31:0] D,
                         input [4:0] A,
                         input E);
wire [3:0] c;

decoder_2x4_22 m(c[3:0],A[4],A[3],E);
decoder_3x8_22 w1(D[7:0],A[2],A[1],A[0],c[0]);
decoder_3x8_22 w2(D[15:8],A[2],A[1],A[0],c[1]);
decoder_3x8_22 w3(D[23:16],A[2],A[1],A[0],c[2]);
decoder_3x8_22 w4(D[31:24],A[2],A[1],A[0],c[3]);

endmodule

`timescale 1ns/1ps

module decoder_5x32_22_ts();

reg [4:0] A;
reg E;
wire [31:0] D2;
integer i = 0;

decoder_5x32_22 c(D2,A,E);
```

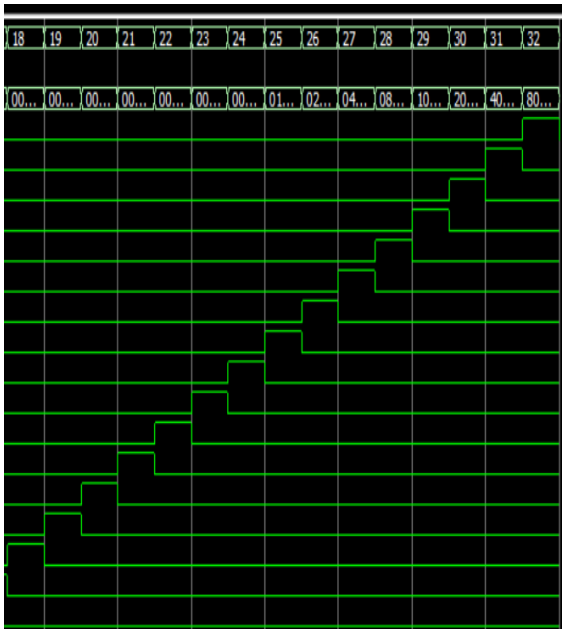
```
initial
begin

E = 1'b0;
A = 5'b0;

$monitor ("in : %d E : %b out : %b",A,E,D2);

#10 E = 1'b1;
for (i = 1 ; i <= 32 ; i = i + 1)
#10 A = A + 1;
end

endmodule
```

(c) Simulation:-



Inputs Waves



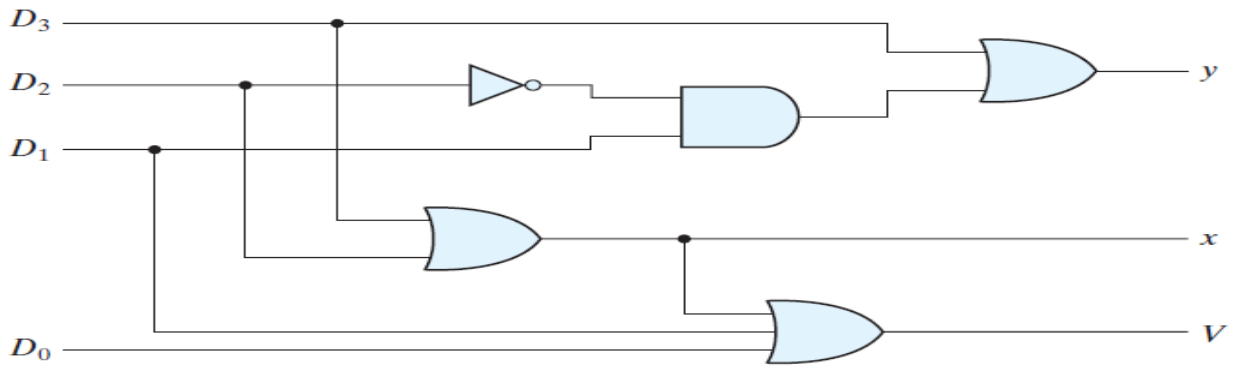


Output Waves

Q2-Write the HDL gate-level description of the priority encoder circuit shown.

ANS

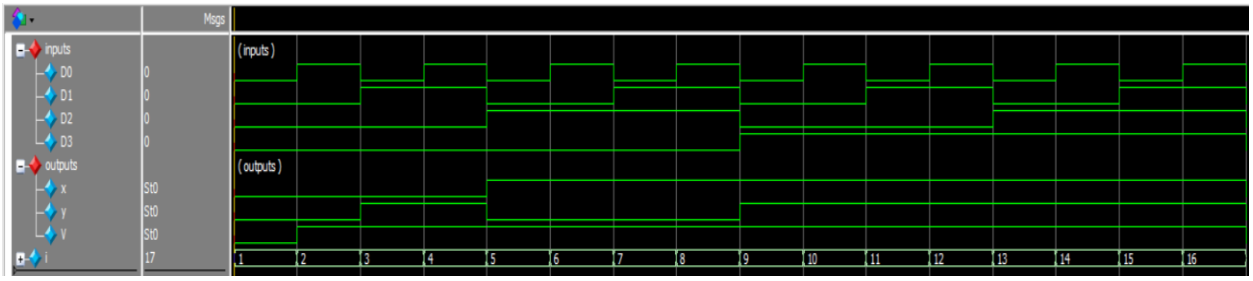
(a) Block diagram:-



(b) Code:-

```
module decoder_4_23_fig (output x,y,V,input D0,D1,D2,D3);
wire n_D2,And;
not (n_D2,D2);
and (And,D1,n_D2);
or (x,D2,D3);
or (y,And,D3);
or (V,x,D0,D1);
endmodule

`timescale 1ns/1ps

module tb_decoder_4_23_fig ();
reg D0,D1,D2,D3;
wire x,y,V;
integer i;
decoder_4_23_fig M (x,y,V,D0,D1,D2,D3);
initial
begin
D0=0;D1=0;D2=0;D3=0;
for (i=1;i<17;i=i+1)
#5 {D3,D2,D1,D0}={D3,D2,D1,D0}+1;
end
endmodule
```

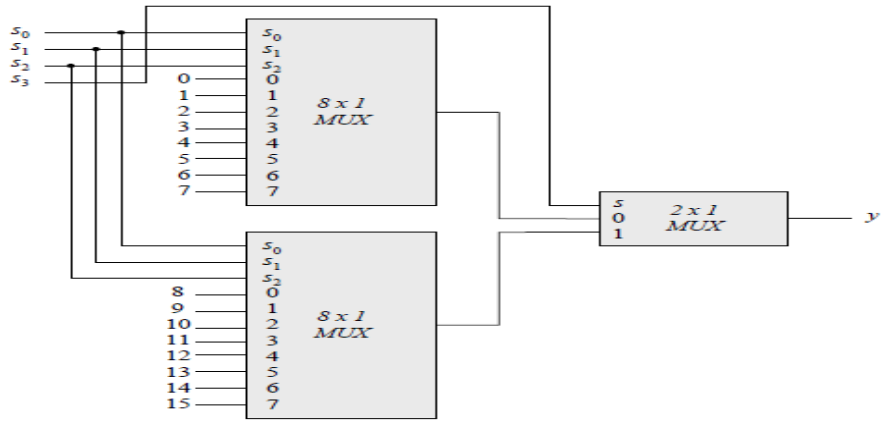
(c) Simulation:-



Q3-Construct a 16×1 multiplexer with two 8×1 and one 2×1 multiplexers.use block diagrams.

ANS

(a) Block diagram:-



(b) Code:-

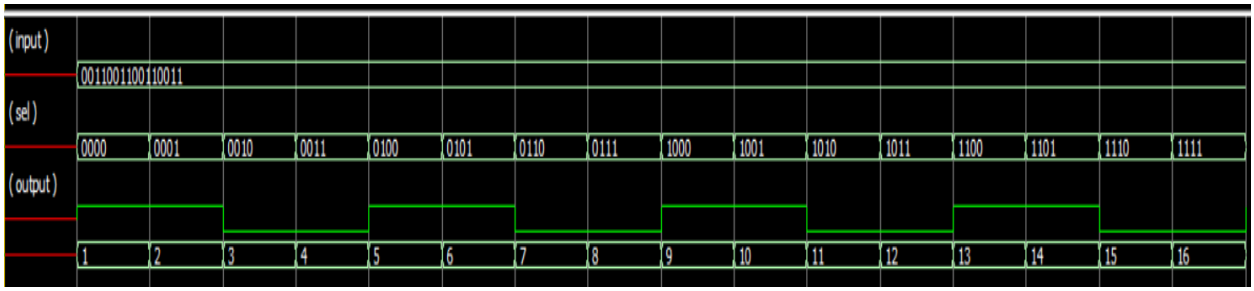
```
module mulyiplexer_2x1 ( output F,
                         input [1:0] I , S);
wire [1:0] W;
wire Sn;

not (Sn,S);
and (W[0],I[0],Sn);
and (W[1],I[1],S);
or (F,W[1],W[0]);

endmodule

module multiplexer_2x1_ts();

reg [1:0] I;
reg S;
wire D;

mulyiplexer_2x1 c(D,I,S);

initial
begin

S = 1'b0;
I = 2'b0;
```

```
$monitor ("in = %d S = %b out = %b",I,S,D);

#10
S = 0; I[0] = 0; I[1] = 1;
#10
S = 0; I[0] = 1; I[1] = 0;
#10
S = 1; I[0] = 1; I[1] = 0;
#10
S = 1; I[0] = 0; I[1] = 1;

end

endmodule

//////////////////////////////////////////////////////

module mulyiplexer_8x1 ( output F,
                         input [7:0] I , input [2:0] S);
wire [7:0] C;
wire [2:0] Sn;

not (Sn[0],S[0]);
not (Sn[1],S[1]);
not (Sn[2],S[2]);
and (C[0],I[0],Sn[2],Sn[1],Sn[0]);
and (C[1],I[1],Sn[2],Sn[1],S[0]);
and (C[2],I[2],Sn[2],S[1],Sn[0]);
and (C[3],I[3],Sn[2],S[1],S[0]);
and (C[4],I[4],S[2],Sn[1],Sn[0]);
and (C[5],I[5],S[2],Sn[1],S[0]);
and (C[6],I[6],S[2],S[1],Sn[0]);
and (C[7],I[7],S[2],S[1],S[0]);
or (F,C[0],C[1],C[2],C[3],C[4],C[5],C[6],C[7]);

endmodule


module multiplexer_8x1_ts();

reg [7:0] I;
reg [2:0] S;
wire D;

mulyiplexer_8x1 c(D,I,S);

initial
begin

S = 3'b0;
I = 8'b0;
```

```
$monitor ("in = %b S = %b out = %b",I,S,D);

#10
S = 3'b000; I[0]=1; I[1]=0; I[2]=0; I[3]=0; I[4]=0; I[5]=0; I[6]=0; I[7]=0;
#10
S = 3'b001; I[0]=0; I[1]=0; I[2]=0; I[3]=0; I[4]=0; I[5]=0; I[6]=0; I[7]=0;
#10
S = 3'b010; I[0]=0; I[1]=0; I[2]=1; I[3]=0; I[4]=0; I[5]=0; I[6]=0; I[7]=0;
#10
S = 3'b011; I[0]=1; I[1]=0; I[2]=0; I[3]=0; I[4]=0; I[5]=0; I[6]=0; I[7]=0;
#10
S = 3'b100; I[0]=1; I[1]=0; I[2]=0; I[3]=0; I[4]=1; I[5]=0; I[6]=0; I[7]=0;
#10
S = 3'b101; I[0]=1; I[1]=0; I[2]=0; I[3]=0; I[4]=0; I[5]=0; I[6]=0; I[7]=0;
#10
S = 3'b110; I[0]=1; I[1]=0; I[2]=0; I[3]=0; I[4]=0; I[5]=0; I[6]=1; I[7]=0;
#10
S = 3'b111; I[0]=1; I[1]=0; I[2]=0; I[3]=0; I[4]=0; I[5]=0; I[6]=0; I[7]=0;

end

endmodule

////////////////////////////////////

module mulyiplexer_16x1 ( output F,
                          input [15:0] I , input [3:0] S);
wire [1:0] W;

mulyiplexer_8x1 C0 (W[0],I[7:0],S[2:0]);
mulyiplexer_8x1 C1 (W[1],I[15:8],S[2:0]);
mulyiplexer_2x1 C3 (F,W,S[3]);

endmodule

module multiplexer_16x1_ts();

reg [15:0] I;
reg [3:0] S;
wire D;

mulyiplexer_16x1 c(D,I,S);

initial
begin

S = 4'b0;
I = 16'b0;

$monitor ("in = %b S = %b out = %b",I,S,D);

#10
```

```
S = 4'b0000; I[0]=1; I[1]=0; I[2]=0; I[3]=0; I[4]=0; I[5]=0; I[6]=0; I[7]=0;
#10
S = 4'b0001; I[0]=0; I[1]=0; I[2]=0; I[3]=0; I[4]=0; I[5]=0; I[6]=0; I[7]=0;
#10
S = 4'b0010; I[0]=0; I[1]=0; I[2]=1; I[3]=0; I[4]=0; I[5]=0; I[6]=0; I[7]=0;
#10
S = 4'b0011; I[0]=1; I[1]=0; I[2]=0; I[3]=0; I[4]=0; I[5]=0; I[6]=0; I[7]=0;
#10
S = 4'b0100; I[0]=1; I[1]=0; I[2]=0; I[3]=0; I[4]=1; I[5]=0; I[6]=0; I[7]=0;

#10
S = 4'b0101; I[0]=1; I[1]=0; I[2]=0; I[3]=0; I[4]=0; I[5]=0; I[6]=0; I[7]=0;
#10
S = 4'b0110; I[0]=1; I[1]=0; I[2]=0; I[3]=0; I[4]=0; I[5]=0; I[6]=1; I[7]=0;
#10
S = 4'b0111; I[0]=1; I[1]=0; I[2]=0; I[3]=0; I[4]=0; I[5]=0; I[6]=0; I[7]=0;
#10
S = 4'b1000; I[8]=1; I[9]=0; I[10]=0; I[11]=0; I[12]=0; I[13]=0; I[14]=0; I[15]=0;
#10
S = 4'b1001; I[8]=0; I[9]=1; I[10]=0; I[11]=0; I[12]=0; I[13]=0; I[14]=0; I[15]=0;
#10
S = 4'b1010; I[8]=0; I[9]=0; I[10]=1; I[11]=0; I[12]=0; I[13]=0; I[14]=0; I[15]=0;
#10
S = 4'b1011; I[8]=0; I[9]=0; I[10]=0; I[11]=1; I[12]=0; I[13]=0; I[14]=0; I[15]=0;
#10
S = 4'b1100; I[8]=0; I[9]=0; I[10]=0; I[11]=0; I[12]=1; I[13]=0; I[14]=0; I[15]=0;
#10
S = 4'b1101; I[8]=0; I[9]=0; I[10]=0; I[11]=0; I[12]=0; I[13]=1; I[14]=0; I[15]=0;
#10
S = 4'b1110; I[8]=0; I[9]=0; I[10]=0; I[11]=0; I[12]=0; I[13]=0; I[14]=1; I[15]=0;
#10
S = 4'b1111; I[8]=0; I[9]=0; I[10]=0; I[11]=0; I[12]=0; I[13]=0; I[14]=0; I[15]=1;
end

endmodule
```

(c) Simulation:-



Q4-Write a hierarchical gate-level model of the multiplexer described in Q3 (diagram block, code completion and simulation is in Q3):-

ANS

Code:-

```
module mux_2_1 (output X,input [1:0]in,input sel);
wire nsel;
wire [1:0] A;
not (nsel,sel);
and (A[0],in[0],nsel);
and (A[1],in[1],sel);
or (X,A[0],A[1]);
endmodule

module mux_8_1 (output D,input [7:0] in,input [2:0] sel);
wire [7:0] A;
wire [2:0] nsel;
not (nsel[0],sel[0]);
not (nsel[1],sel[1]);
not (nsel[2],sel[2]);
and (A[0],in[0],nsel[2],nsel[1],nsel[0]);
and (A[1],in[1],nsel[2],nsel[1],sel[0]);
and (A[2],in[2],nsel[2],sel[1],nsel[0]);
and (A[3],in[3],nsel[2],sel[1],sel[0]);
and (A[4],in[4],sel[2],nsel[1],nsel[0]);
and (A[5],in[5],sel[2],nsel[1],sel[0]);
and (A[6],in[6],sel[2],sel[1],nsel[0]);
and (A[7],in[7],sel[2],sel[1],sel[0]);
or (D,A[0],A[1],A[2],A[3],A[4],A[5],A[6],A[7]);
endmodule
```

Q5-Write an HDL behavioral description of a four-bit comparator with a six-bit output Y[5:0].Bit 5 of Y is for “equals”,bit 4 for “not equal”,bit 3 for “greater than”,bit 2 for “less than”,bit 1 for “greater than or equal”,bit 0 for “less than or equal”.

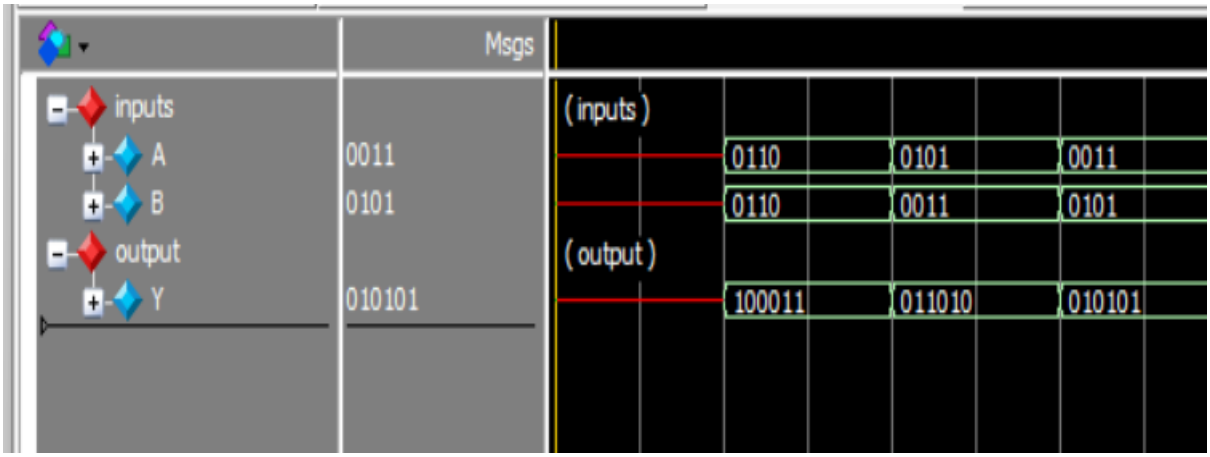
ANS

(a) Code:-

```
module comparator_4_bit( output reg [5:0] Y,
                         input [3:0] A,
                         input [3:0] B);

always @* begin
   // Equals
   Y[5] = (A == B);

   // Not Equal
   Y[4] = (A != B);

   // Greater Than
   Y[3] = (A > B);
```

```
    // Less Than
    Y[2] = (A < B);

    // Greater Than or Equal
    Y[1] = (A >= B);

    // Less Than or Equal
    Y[0] = (A <= B);
end

endmodule



module comparator_4_bit_ts ();

  reg [3:0] A;
  reg [3:0] B;
  wire [5:0] Y;

  comparator_4_bit C (Y,A,B);
  initial
  begin
    $monitor("A=%b, B=%b, Y=%b", A, B, Y);

    // Test case 1: A = B
    A = 4'b1111; B = 4'b1111; #10;

    // Test case 2: A > B
    A = 4'b1110; B = 4'b1101; #10;

    // Test case 3: A < B
    A = 4'b0001; B = 4'b0110; #10;

    // Test case 4: A >= B
    A = 4'b1111; B = 4'b1110; #10;

    // Test case 5: A <= B
    A = 4'b0010; B = 4'b0010; #10;

  end

endmodule
```

(b) Simulation:-



Q6-Explain the function of the circuit specified by the following HDL description:-

```
module Prob4_43(A, B, S, E, Q);
input [1:0] A, B;
input S, E;
output [1:0] Q;
assign Q=E ? (S ?A : B ) : 'bz;
endmodule
```

ANS

Mux 2(A,B)×1(Q) with 2 bit inputs,selector(S) and enable(E),2 bit output

If E = 0 → Q = z ,if E = 1 → Q will be depend on S.

When E=1

If S = 1 → Q=A.

If S = 0 → Q=B.