EX. NO: 03	**ENCODERS AND DECODERS**
DATE : 19/09/2024	

AIM: To simulate encoders and decoders in behavior, structural, and dataflow model of Verilog HDL.

SOFTWARE USED: Xilinx Vivado

HARDWARE USED: Basys3 FPGA Board

ENCODERS:

VERILOG HDL CODES:

4 to 2 line Encoder in Behavioral Model:

```
module encoder4to2_behav(D,Q);
input [3:0] D;
output reg [1:0] Q;
always@(*)
 begin
 if (D==4'b0001)
 begin Q=2'b00; end
 else if (D==4'b0010)
 begin Q=2'b01; end
 else if (D==4'b0100)
 begin Q=2'b10; end
 else if (D==4'b1000)
 begin Q=2'b11; end
 else
 begin Q=2'b00; end
end
endmodule
```

4 to 2 line Encoder in Structural Model:

```
module encoder4to2_struct(D,Q );
input [3:0] D;
output[1:0] Q;
or (Q[0],D[1],D[3]);
or (Q[1],D[2],D[3]);
endmodule
```

4 to 2 line Encoder in Data flow model:

```
module encoder4to2_data(D,Q );
input [3:0] D;
output[1:0] Q;
assign Q[0]=D[1]|D[3];
assign Q[1]=D[2]|D[3];
endmodule
```

Test bench 4 to 2 line Encoder:

```
module tb_encoder4to2;
reg [3:0] D;
wire [1:0] Q;
//encoder4to2_data EN1(D,Q);
encoder4to2_struct EN2(D,Q);
//encoder4to2_behav EN3(D,Q);
initial
begin
    D=4'b0001;
    #10  D=4'b0010;
    #10  D=4'b0100;
    #10  D=4'b1000;
    #10  D=4'b0000;
    #10 $stop;
end
initial begin
    $monitor(" D = %b, Q = %b",D,Q);
   end
endmodule
```

8 to 3 line Encoder:

```
module encoder8to3(D,Q);
input [7:0] D;
output[2:0] Q;
or (Q[0],D[1],D[3],D[5],D[7]);
or (Q[1],D[2],D[3],D[6],D[7]);
or (Q[2],D[4],D[5],D[6],D[7]);
endmodule
```

Test bench for 8 to 3 line Encoder:

```
module tb_encoder8to3;
reg [7:0] D;
wire [2:0] Q;
encoder8to3 EN3(D,Q);
initial
begin
    D=8'b00000001;
    #10  D=8'b00000010;
    #10  D=8'b00000100;
    #10  D=8'b00001000;
    #10  D=8'b00010000;
    #10  D=8'b00100000;
    #10  D=8'b01000000;
    #10  D=8'b10000000;
    #10 $stop;
end
initial begin
    $monitor(" D = %b, Q = %b",D,Q);
   end
endmodule
```

8 to 3 Priority Encoder:

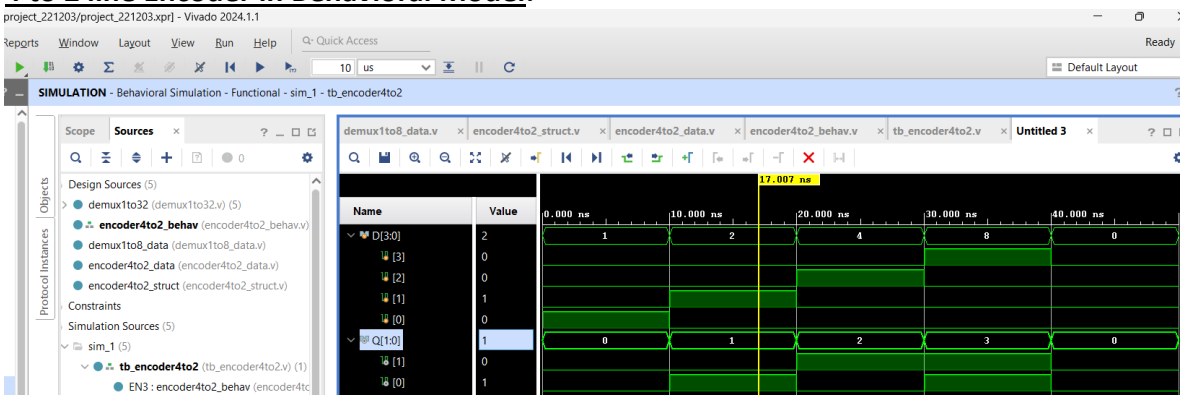
```
module encoder8to3_priority(D, Q);
   input [7:0] D;
   output reg [2:0] Q;

   always @(*) begin
          if (D[7]==1) Q=3'b111;
          else if (D[6]==1) Q=3'b110;
          else if (D[5]==1) Q=3'b101;
          else if (D[4]==1) Q=3'b100;
          else if (D[3]==1) Q=3'b011;
          else if (D[2]==1) Q=3'b010;
          else if (D[1]==1) Q=3'b001;
          else if (D[0]==1) Q=3'b000;
          else
                    Q=3'bxxx;
         end
endmodule
```

Testbench for 8 to 3 Priority Encoder:

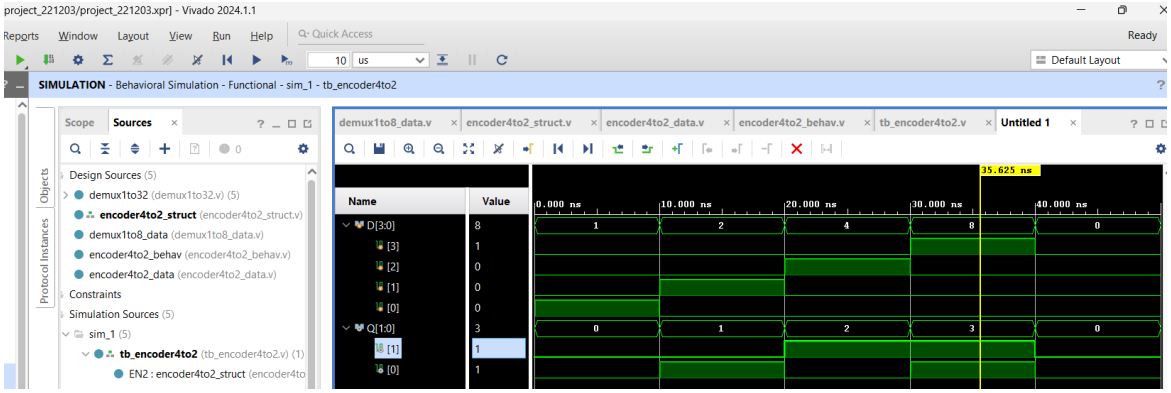
```
module tb_encoder8to3_priority;
reg [7:0] D;
wire [2:0] Q;
encoder8to3_priority EN3(D,Q);
initial
begin
    D=8'b00000001;
    #10  D=8'b00000010;
    #10  D=8'b00000100;
    #10  D=8'b00001000;
    #10  D=8'b00010000;
    #10  D=8'b00100000;
    #10  D=8'b01000000;
    #10  D=8'b10000000;
    #10  D=8'b00000001;
    #10  D=8'b00000011;
    #10  D=8'b00000111;
    #10  D=8'b00001111;
    #10  D=8'b00011111;
    #10  D=8'b00111111;
    #10  D=8'b01111111;
    #10  D=8'b11111111;
    #10 $stop;
end
initial begin
    $monitor(" D = %b, Q = %b",D,Q);
   end
endmodule
```

SIMULATION WAVEFORMS:

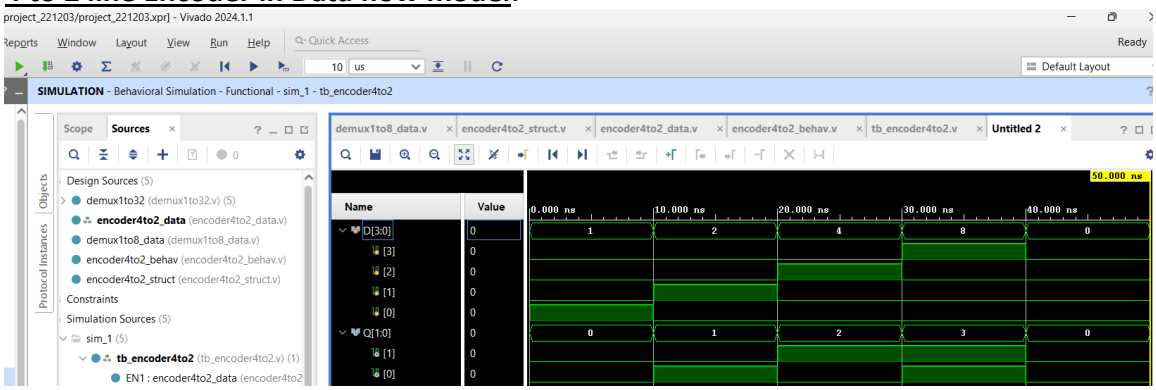
4 to 2 line Encoder in Behavioral Model:



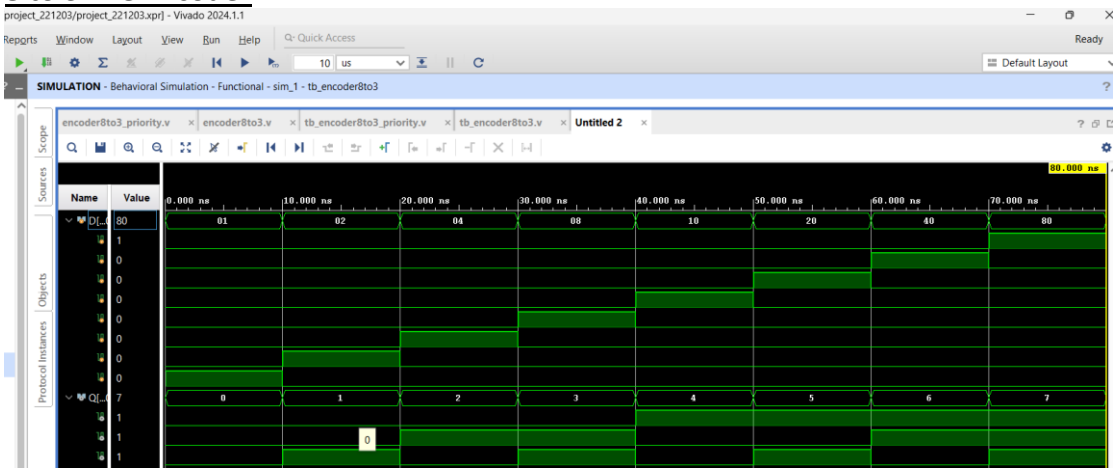
4 to 2 line Encoder in Structural Model:



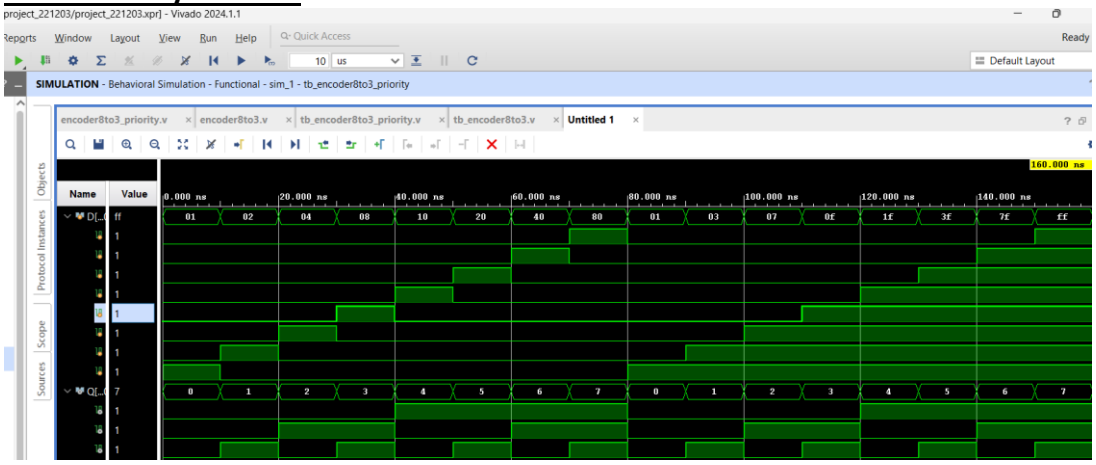
4 to 2 line Encoder in Data flow model:



8 to 3 line Encoder:



8 to 3 Priority Encoder:



HARDWARE OUTPUT:

4 to 2 line Encoder:

I/O ports:

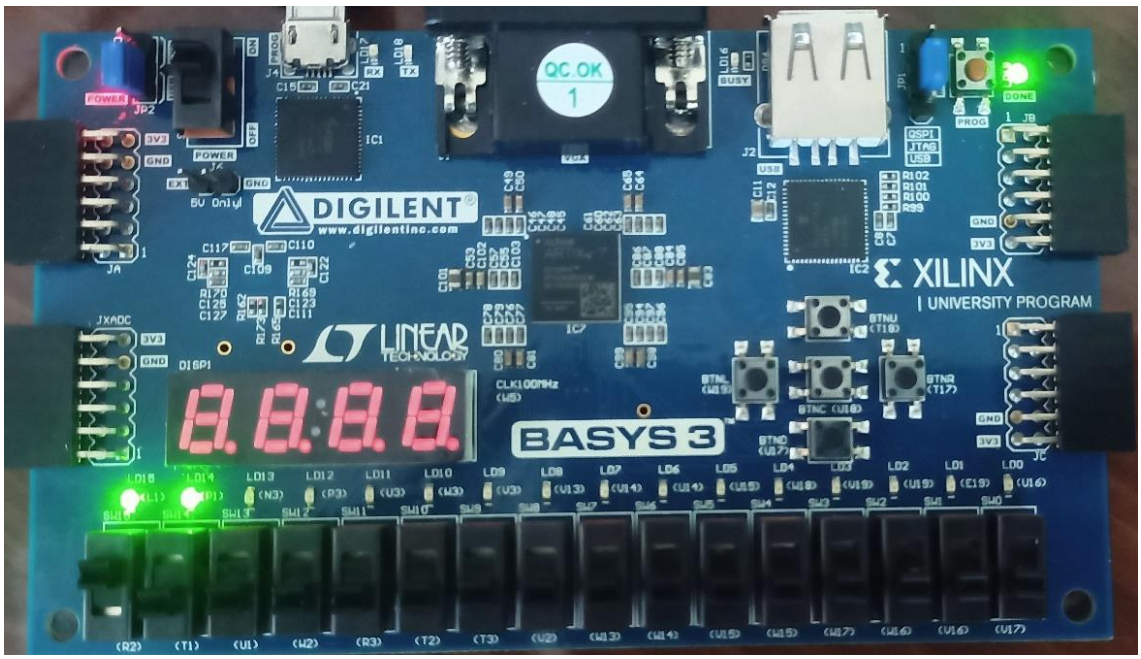
Inputs:

D[3]: R2
D[2]: T1
D[1]: U1
D[0]: W2

Outputs:

Q[1]: L1
Q[0]: P1

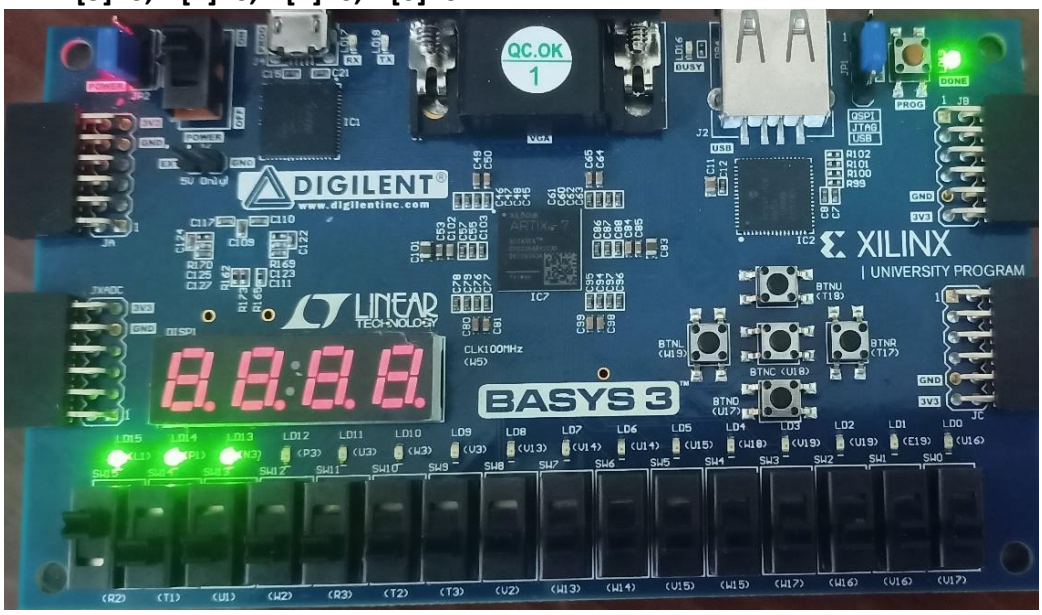
For D[3]=1, D[2]=0, D[1]=0, D[0]=0



8 to 3 line Encoder:
I/O ports:

Inputs:		**Outputs:**
D{7]: R2	D[3]: R3	Q[2]: L1
D[6]: T1	D[2]: T2	Q[1]: P1
D[5]: U1	D[1]: T3	Q[0]: N3
D[4]: W2	D[0]: V2	

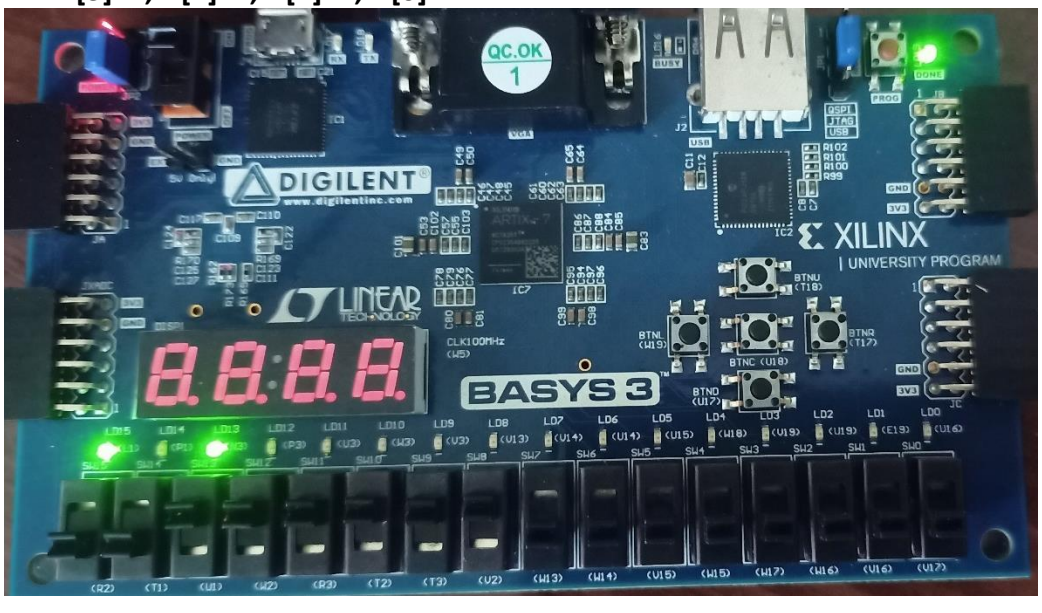
For D[7]=1, D[6]=0, D[5]=0, D[4]=0
D[3]=0, D[2]=0, D[1]=0, D[0]=0



8 to 3 Priority Encoder:

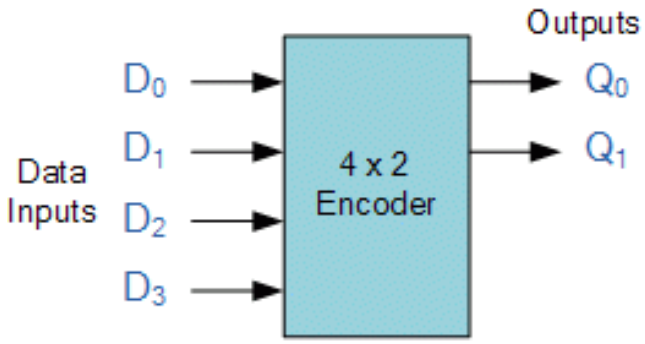
Inputs:		**Outputs:**
D{7]: R2	D[3]: R3	Q[2]: L1
D[6]: T1	D[2]: T2	Q[1]: P1
D[5]: U1	D[1]: T3	Q[0]: N3
D[4]: W2	D[0]: V2	

For D[7]=0, D[6]=0, D[5]=1, D[4]=1
D[3]=1, D[2]=1, D[1]=1, D[0]=1



TRUTH TABLES AND CIRCUIT DIAGRAMS:

4 to 2 line Encoder:



Inputs				Outputs	
D_3	D_2	D_1	D_0	Q_1	Q_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1
0	0	0	0	x	x

Q1 = ~D0 & ~D1 & (D2 ^ D3)
Q0 = ~D0 & ~D3 & (D2 ^ D1)

Q1 = D2 | D3
Q0 = D1 | D3

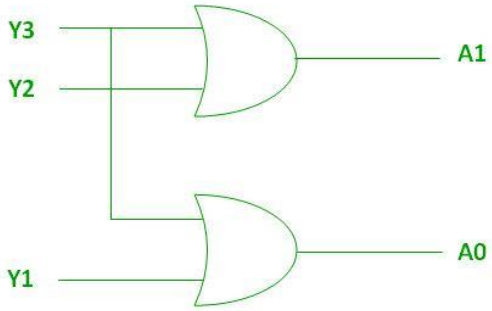


Fig. 4to2 line decoder

8 to 3 line Encoder:

INPUTS								OUTPUTS		
Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	A2	A1	A0
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

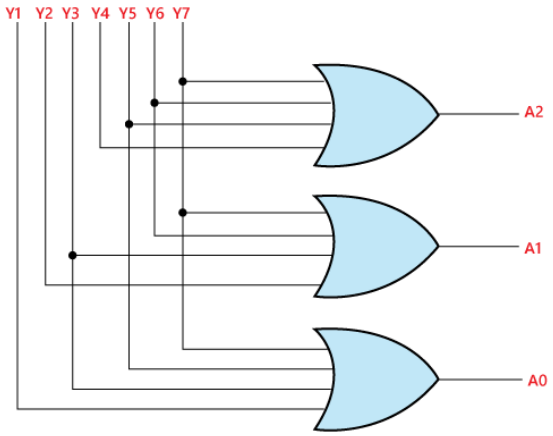


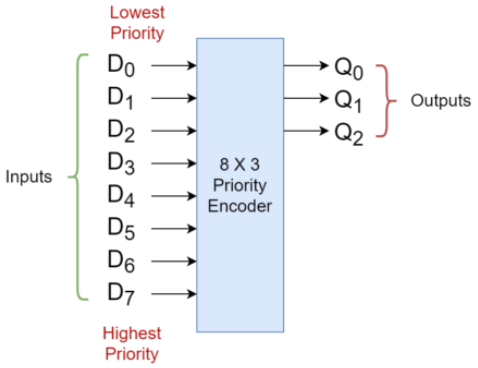
Fig. 8to3 line encoder

$A_2=Y_4+Y_5+Y_6+Y_7$

$A_1=Y_2+Y_3+Y_6+Y_7$

$A_0=Y_7+Y_5+Y_3+Y_1$

8 to 3 Priority Encoder:



Inputs								Outputs		
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Q_2	Q_1	Q_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	x	0	0	1
0	0	0	0	0	1	x	x	0	1	0
0	0	0	0	1	x	x	x	0	1	1
0	0	0	1	x	x	x	x	1	0	0
0	0	1	x	x	x	x	x	1	0	1
0	1	x	x	x	x	x	x	1	1	0
1	x	x	x	x	x	x	x	1	1	1

X = Don't Care

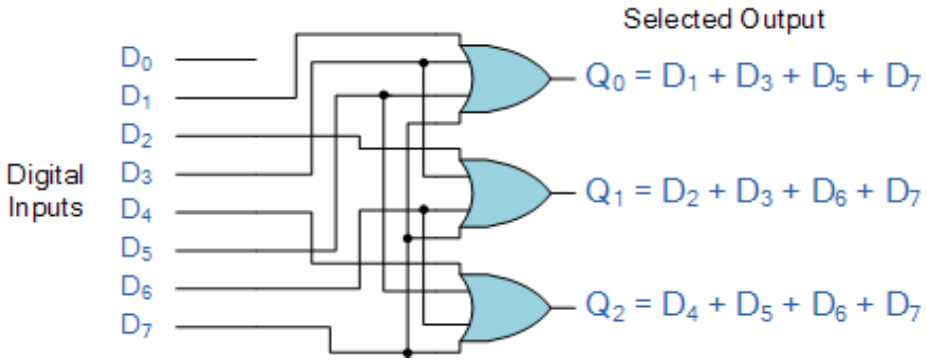


Fig. 8to3 priority encoder

DECODERS:

VERILOG HDL CODES:

2 to 4 Decoder in Behavioral Model:

```
module decoder2to4_behav(A, Y);
   input [1:0] A;
   output reg [3:0] Y;
   always @(*) begin
      Y = 4'b0000; // Default
      case (A)
         2'b00: Y[0] = 1;
         2'b01: Y[1] = 1;
         2'b10: Y[2] = 1;
         2'b11: Y[3] = 1;
      endcase
   end
endmodule
```

2 to 4 Decoder in Dataflow Model:

```
module decoder2to4_data(A, Y);
   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
```

2 to 4 Decoder in Structural Model:

```
module decoder2to4_struct(A, Y);
   input [1:0] A;
   output [3:0] Y;
   wire not_A0, not_A1;
   not (not_A0, A[0]);
   not (not_A1, A[1]);
   and (Y[0], not_A0, not_A1);
   and (Y[1], A[0], not_A1);
   and (Y[2], not_A0, A[1]);
   and (Y[3], A[0], A[1]);
endmodule
```

Test Bench for 2 to 4 Decoder:

```
module tb_decoder2to4;
   reg [1:0] A;
   wire [3:0] Y;
   decoder2to4_behav DC1(A, Y);
//decoder2to4_data DC2(A, Y);
//decoder2to4_struct DC3(A, Y);
   initial begin
      A = 2'b00; #10;
      A = 2'b01; #10;
      A = 2'b10; #10;
      A = 2'b11; #10;
      $stop;
   end
   initial begin
      $monitor("A = %b, Y = %b", A, Y);
   end
endmodule
```

3 to 8 Decoder:

```
module decoder3to8_behav(A, Y);
   input [2:0] A;
   output reg [7:0] Y;
   always @(*) begin
      Y = 8'b00000000; // Default
      case (A)
         3'b000: Y[0] = 1;
         3'b001: Y[1] = 1;
         3'b010: Y[2] = 1;
         3'b011: Y[3] = 1;
         3'b100: Y[4] = 1;
         3'b101: Y[5] = 1;
         3'b110: Y[6] = 1;
         3'b111: Y[7] = 1;
      endcase
   end
endmodule
```

Test Bench for 3 to 8 Decoder:

```
module tb_decoder3to8;
   reg [2:0] A;
   wire [7:0] Y;
   decoder3to8_behav DC(A, Y);
   initial begin
      A = 3'b000; #10;
      A = 3'b001; #10;
      A = 3'b010; #10;
      A = 3'b011; #10;
      A = 3'b100; #10;
      A = 3'b101; #10;
      A = 3'b110; #10;
```

```
    A = 3'b111; #10;
    $stop;
  end
  initial begin
    $monitor("A = %b, Y = %b", A, Y);
  end
endmodule
```

5 to 32 line decoder using 2 to 4 line Decoder and 3 to 8 line decoders:

```
module decoder5to32(A,Y,enable);
  input [4:0] A;
  input enable;
  output [31:0] Y;
  wire [3:0] E;
decoder2to4 D0 (A[4:3],E,enable);
decoder3to8 D1(A[2:0],Y[7:0],E[0]);
decoder3to8 D2(A[2:0],Y[15:8],E[1]);
decoder3to8 D3(A[2:0],Y[23:16],E[2]);
decoder3to8 D4(A[2:0],Y[31:24],E[3]);
endmodule

module decoder2to4(A, E, enable);
  input [1:0] A;
  input enable;
  output reg [3:0] E;
  always @(*) begin
    if (enable==1) begin
      case (A)
        2'b00: E = 4'b0001;
        2'b01: E = 4'b0010;
        2'b10: E = 4'b0100;
        2'b11: E = 4'b1000;
        default: E = 4'b0000;
      endcase
    end else
      E = 4'b0000;
  end
endmodule

module decoder3to8(A, Y, enable);
  input [2:0] A;
  input enable;
  output reg [7:0] Y;
  always @(*) begin
    if (enable==1) begin
      case (A)
        3'b000: Y = 8'b00000001;
        3'b001: Y = 8'b00000010;
        3'b010: Y = 8'b00000100;
        3'b011: Y = 8'b00001000;
        3'b100: Y = 8'b00010000;
        3'b101: Y = 8'b00100000;
```

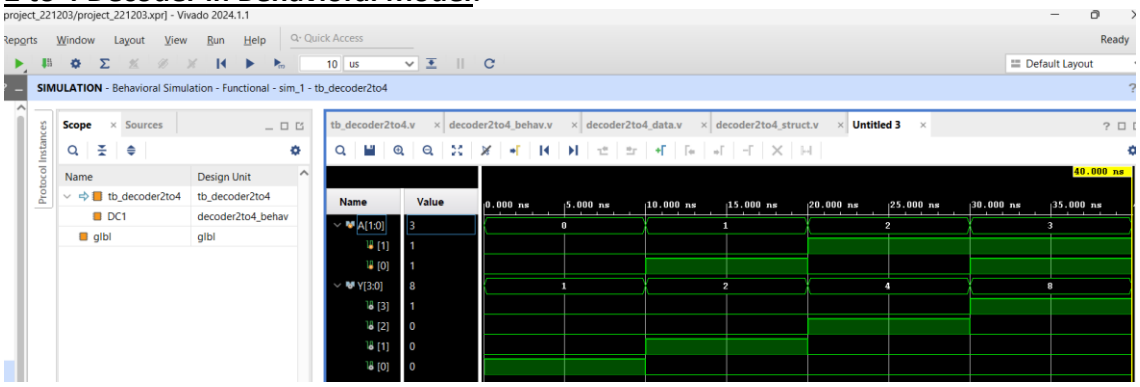
```
            3'b110: Y = 8'b01000000;
            3'b111: Y = 8'b10000000;
            default: Y = 8'b00000000;
        endcase
    end else
        Y = 8'b00000000;
  end
endmodule
```

Test Bench for 5 to 32 Decoder:

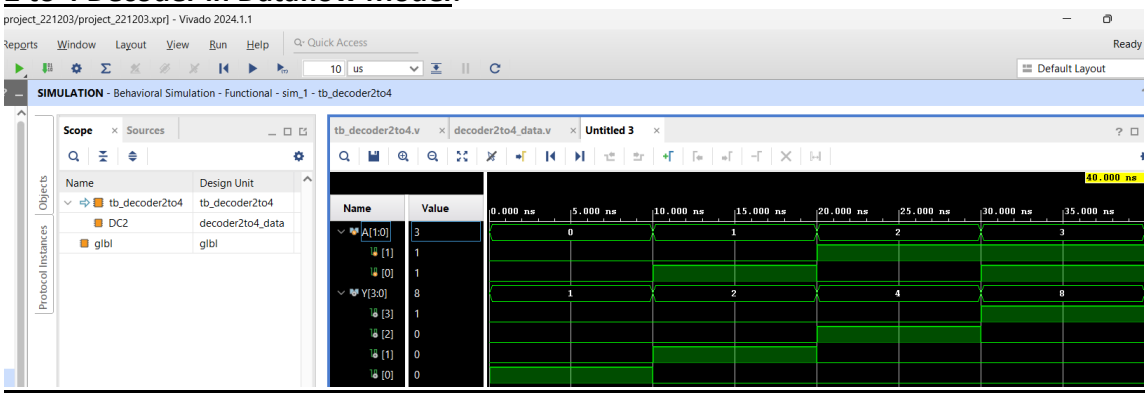
```
module tb_decoder5to32;
  reg [4:0] A;
  wire [31:0] Y;
  reg enable;
  decoder5to32 DC (A,Y,enable);
  initial begin
    $monitor("A = %b, Y = %b,enable = %b", A, Y,enable);
    enable=1'b1;
    for (integer i = 0; i < 32; i = i + 1) begin
      A = i;
      #10;
    end
    enable = 1'b0;
    #10 A=5'b000101;
    $finish;
  end
endmodule
```

SIMULATION WAVEFORMS:

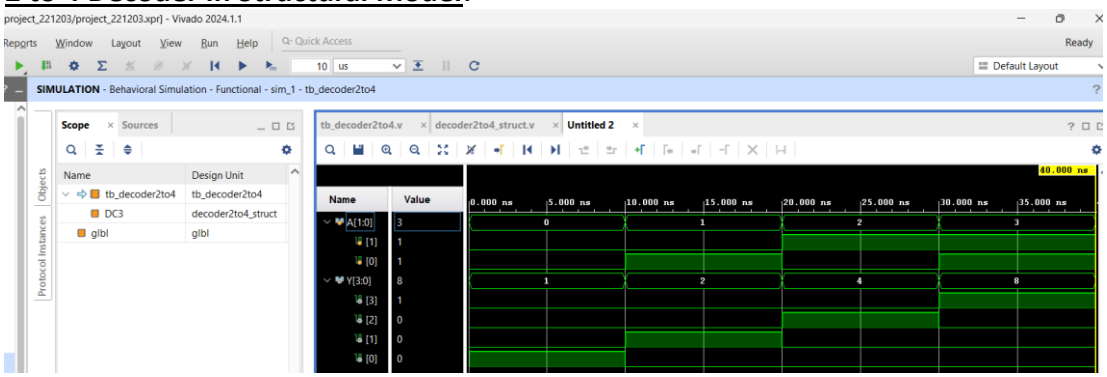
2 to 4 Decoder in Behavioral Model:



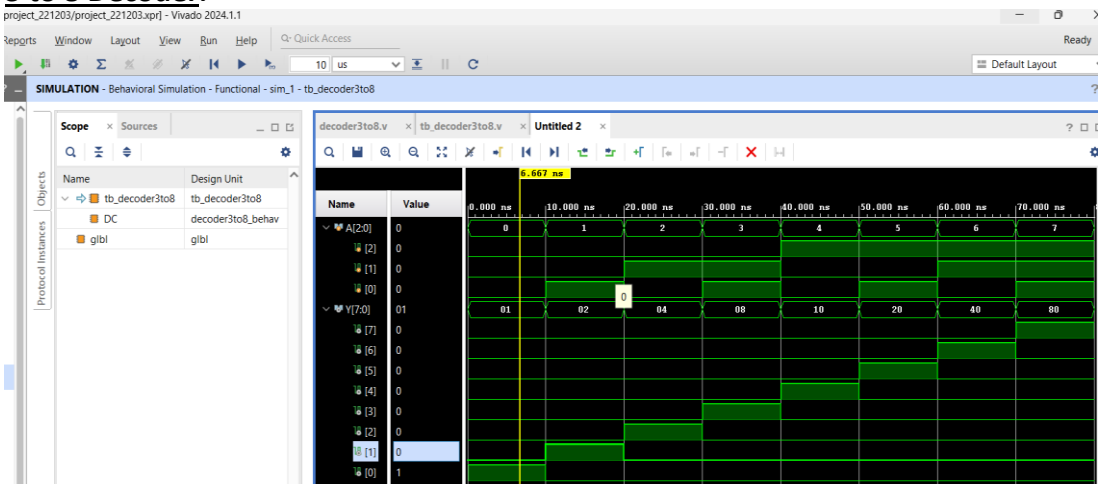
2 to 4 Decoder in Dataflow Model:



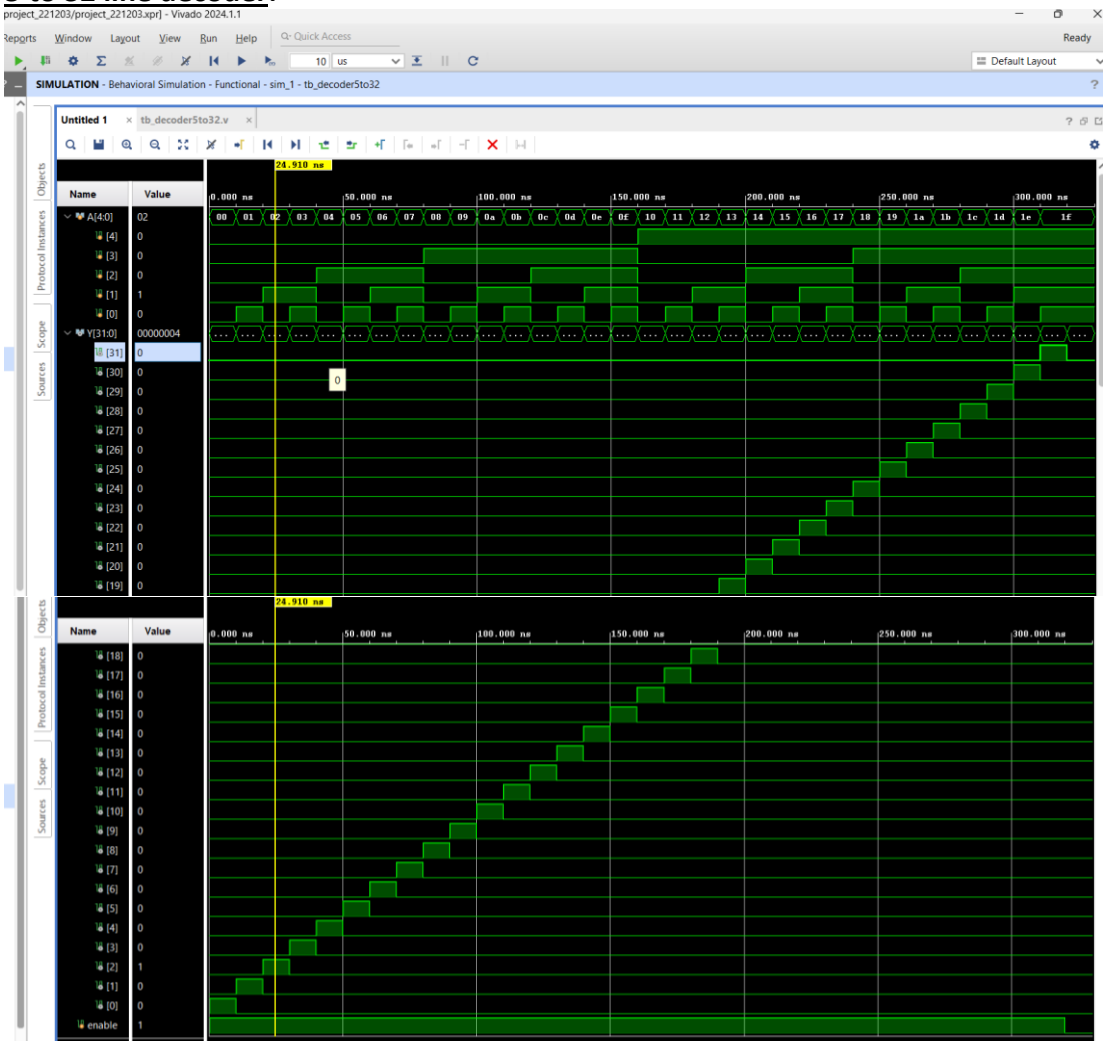
2 to 4 Decoder in Structural Model:



3 to 8 Decoder:



5 to 32 line decoder:



HARDWARE OUTPUT:

2 to 4 Decoder :

I/O ports:

Inputs:

A[1]: R2

A[0]: T1

Outputs:

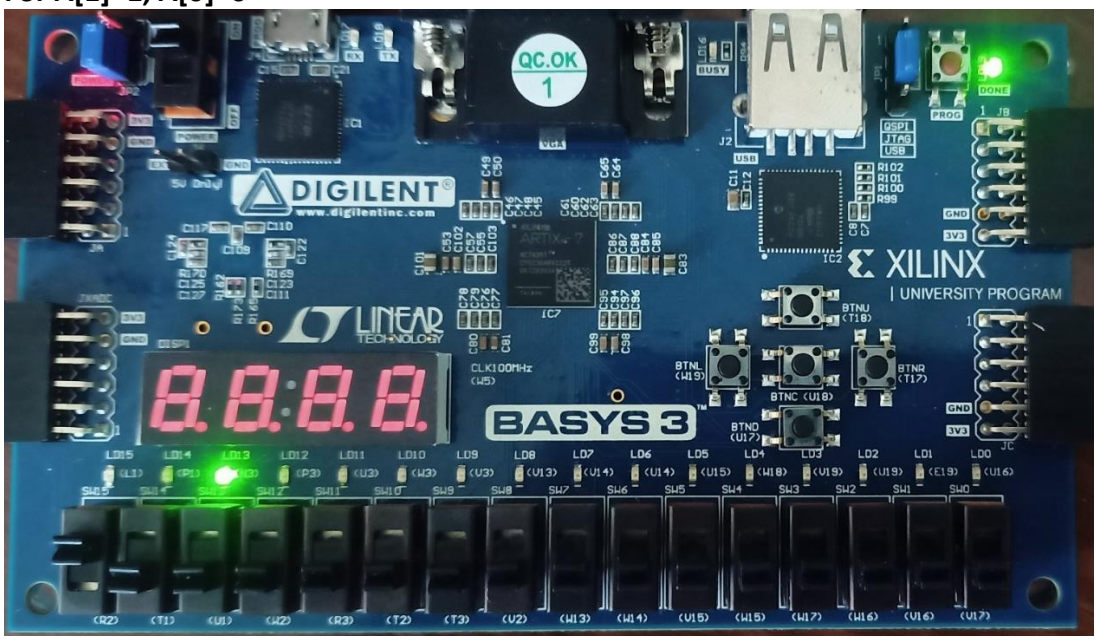
Y[3]: P1

Y[2]: N3

Y[1]: P3

Y[0]: U3

For A[1]=1, A[0]=0



3 to 8 Decoder:

I/O ports:

Inputs:

A[2]: R2

A[1]: T1

A[0]: U1

Outputs:

Y[7]: L1

Y[6]: P1

Y[5]: N3

Y[4]: P3

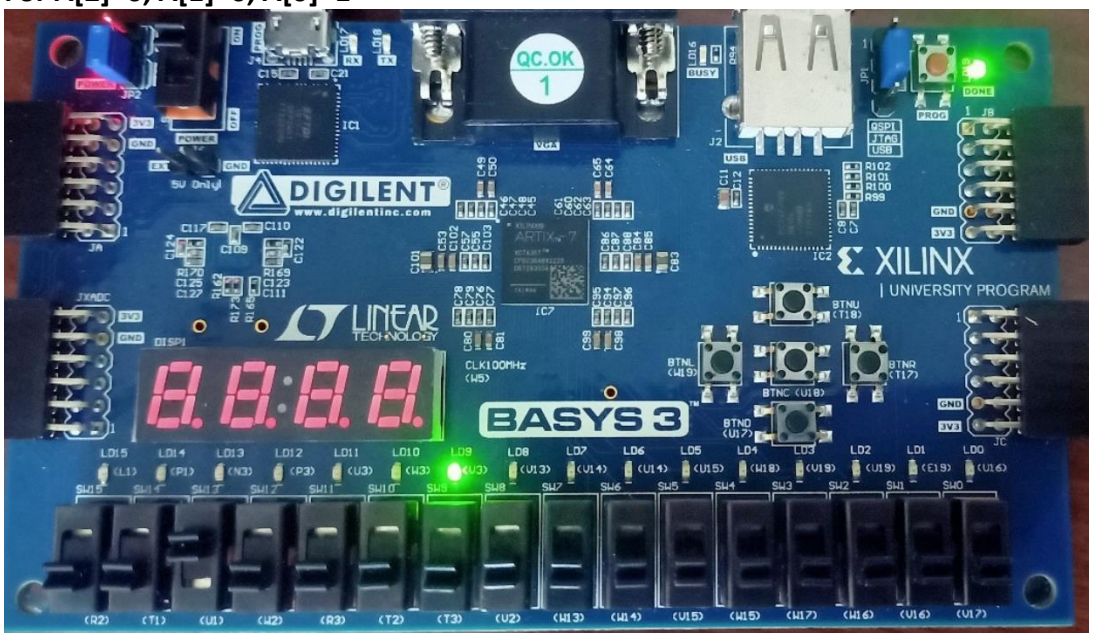
Y[3]: U3

Y[2]: W3

Y[1]: V3

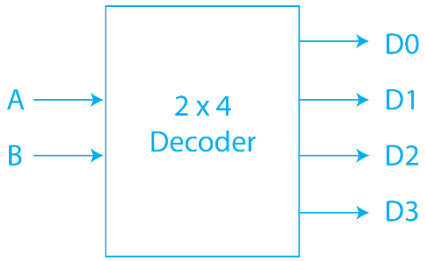
Y[0]: V13

For A[2]=0, A[1]=0, A[0]=1

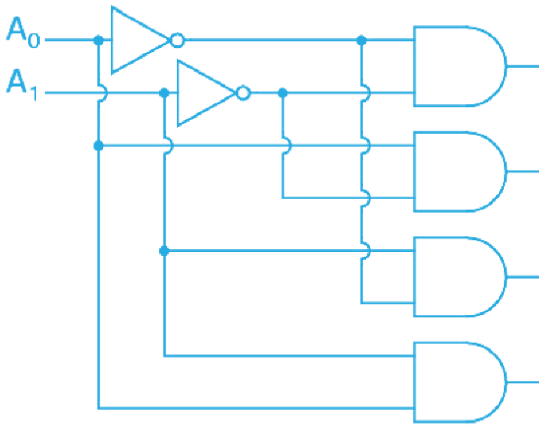


TRUTH TABLES AND CIRCUIT DIAGRAMS:

2 to 4 Decoder:



A	B	Y0	Y1	Y2	Y3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1



$D_0=A_0'A_1'$

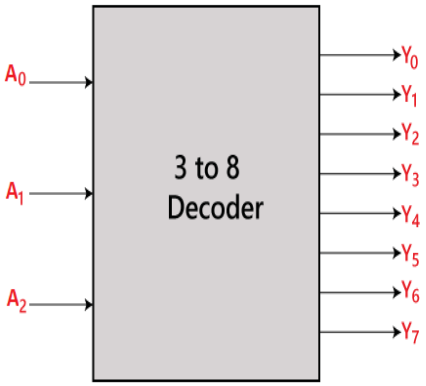
$D_1=A_1'A_0$

$D_2=A_1A_0'$

$D_3=A_1A_0$

Fig. 2 to 4 Decoder

3 to 8 Decoder:



INPUTS			Outputs							
A_2	A_1	A_0	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
x	x	x	0	0	0	0	0	0	0	0
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

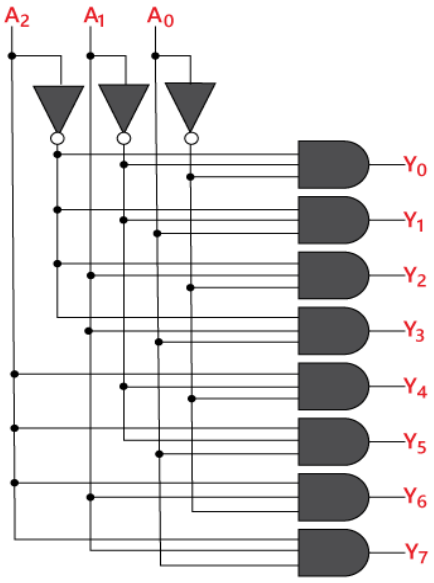


Fig. 3 to 8 Decoder

$Y_0=A_2'A_1'A_0'$

$Y_1=A_2'A_1'A_0$

$Y_2=A_2'A_1A_0'$

$Y_3=A_2'A_1A_0$

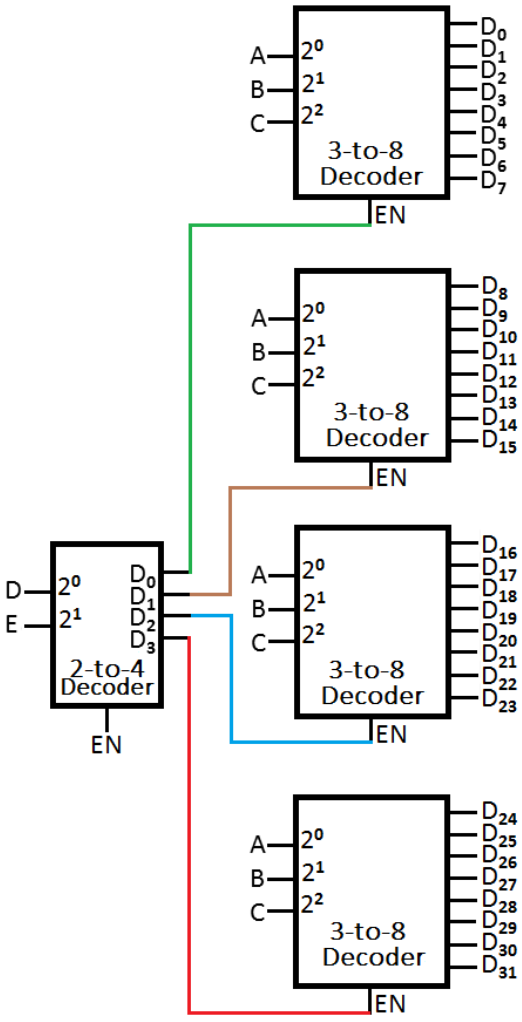
$Y_4=A_2A_1'A_0'$

$Y_5=A_2A_1'A_0$

$Y_6= A_2A_1A_0'$

$Y_7= A_2A_1A_0$

5 to 32 line decoder:



RESULT:

Thus, encoders and decoders were successfully implemented in Verilog HDL using behavioral, structural, and dataflow modeling. Their functionalities were verified through simulation using Xilinx Vivado software on the Basys3 FPGA board.