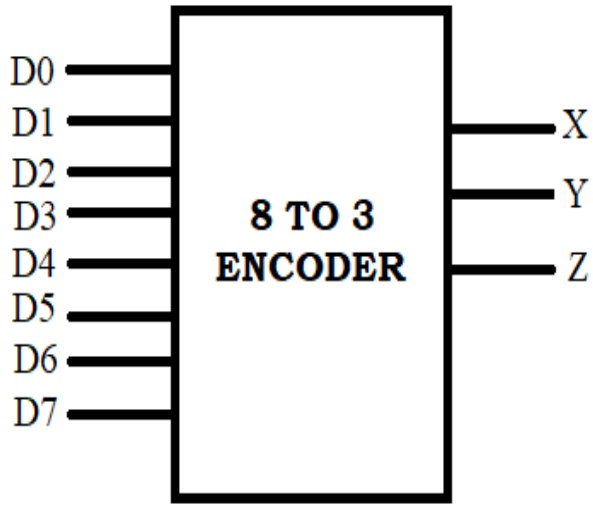
EXP 3: DESIGN OF 8-TO-3 ENCODER (WITHOUT AND WITH PRIORITY)

AIM: Design of 8-to-3 encoder (without and with priority) using HDL code.

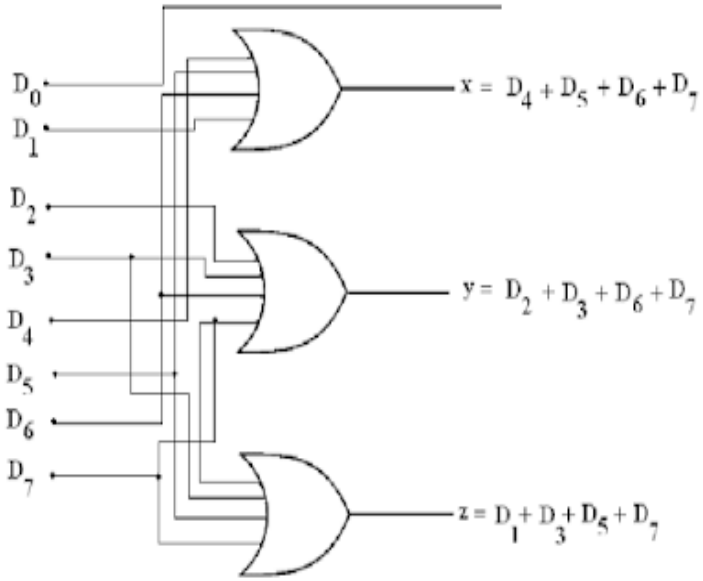
SOFTWARE & HARDWARE:

1. XILINX VIVADO 2018.1 VERSION.
2. FPGA-ZYNQ BOARD **XC7Z020CLG484-1.**
3. JUMPER CABLE WITH POWER SUPPLY.

8:3 encoder Block diagram:



8:3 encoder logic Diagram :



Digital Inputs								Binary Output		
D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**	**X**	**Y**	**Z**
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

Truth Table of 8:3 encoder

VERILOG CODE :

Structural Model	Data Flow Model
module encoder (din, dout); input [7:0] din; output [2:0] dout; reg [2:0] dout; Or g1(dout[0],din[1],din[3],din[5],din[7]); Or g2(dout[1],din[2],din[3],din[6],din[7]); Or g3(dout[2],din[4]4,din[5],din[6],din[7]); Endmodule	module encoder (din, dout); input [7:0] din; output [2:0] dout; reg [2:0] dout; Assign=dout[0],din[1],din[3],din[5],din[7]); Or g2(dout[1],din[2],din[3],din[6],din[7]); Or g3(dout[2],din[4]4,din[5],din[6],din[7]); Endmodule

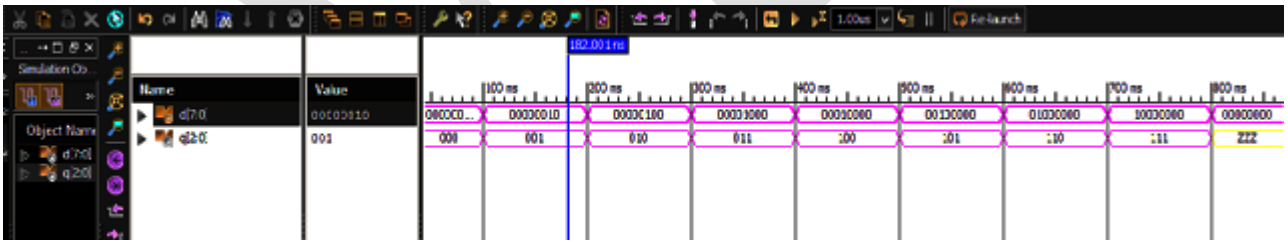
BehaviouralModel	BehaviouralModel with Enable

module encoder (din, dout);
input [7:0] din;
output [2:0] dout;
reg [2:0] dout;
always @(din)
begin
if (din ==8'b00000001) dout=3'b000;
else if (din==8'b00000010) dout=3'b001;
else if (din==8'b00000100) dout=3'b010;
else if (din==8'b00001000) dout=3'b011;
else if (din==8'b00010000) dout=3'b100;
else if (din ==8'b00100000) dout=3'b101;
else if (din==8'b01000000) dout=3'b110;
else if (din==8'b10000000) dout=3'b111;
else dout=3'bX;
end
endmodule
	module encwtoutprio(a,en,y);
	input [7:0] a;
	input en;
	output reg [2:0] y;
	always@(a or en)
	begin
	if(!en)
	y<=1'b0;
	else
	case(a)
	8'b00000001:y<=3'b000;
	8'b00000010:y<=3'b001;
	8'b00000100:y<=3'b010;
	8'b00001000:y<=3'b011;
	8'b00010000:y<=3'b100;
	8'b00100000:y<=3'b101;
	8'b01000000:y<=3'b110;
	8'b10000000:y<=3'b111;
	endcase
	end
	endmodule

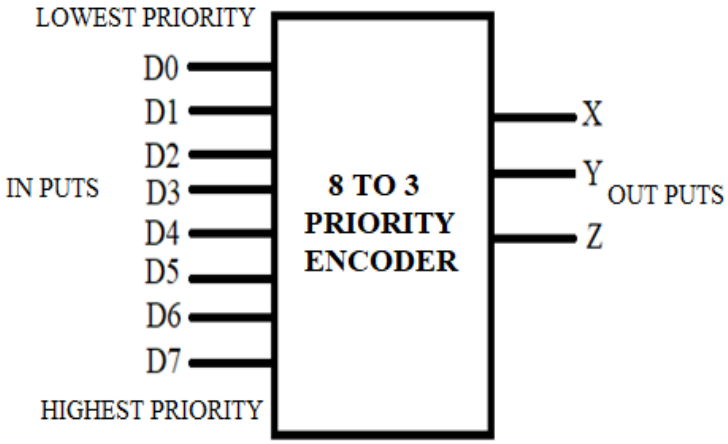
TEST BENCH

```
module encodert_b;
reg [0:7] d;
wire a;
wire b;
wire c;
encodermod uut (.d(d), .a(a), .b(b),.c(c) );
initial begin
#10 d=8'b10000000;
#10 d=8'b01000000;
#10 d=8'b00100000;
#10 d=8'b00010000;
#10 d=8'b00001000;
#10 d=8'b00000100;
#10 d=8'b00000010;
#10 d=8'b00000001;
#10 $stop;
end
endmodule
```

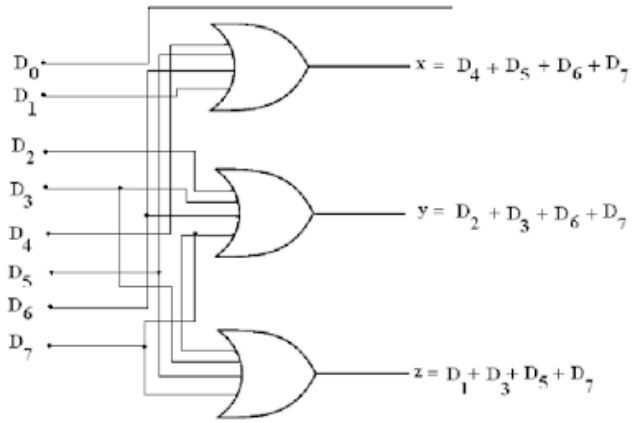
Simulation output: Waveform window: Displays output waveform for verification.



8:3 Priority encoder Block diagram:



8:3 Priority encoder logic Diagram :



Digital Inputs								Binary Output		
D7	D6	D5	D4	D3	D2	D1	D0	X	Y	Z
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

Truth Table of 8:3 encoder

VERILOG CODE :

Structural Model

```
module prior_otb_enco(DOUT, D);
  output   [2:0] DOUT;
  input    [7:0] din;
    wire        din7_not,  din6_not,  din5_not,
din4_not, din2_not;
  wire    wa0, wa1, wa2, wa3, wa4;;
  //instanitate gates
  not    g0 (din7_not, din[7]),
         g1 (din6_not, din[6]),
         g2 (din5_not, din[5]),
         g3 (din4_not, din[4]),
         g4 (din2_not, din[2]);
  and    g5 (wa0, din6_not, din4_not, din[3]),
         g6 (wa1, din5_not, din4_not, din[3]),
         g7 (wa2, din5_not, din4_not, din[2]),
```

Data Flow Model

```
module prio_enco_8x3(dout, din);
   output [2:0] dout;
   input [7:0] din ;
assign dout = (din[7] ==1'b1 ) ? 3'b111:
              (din[6] ==1'b1 ) ? 3'b110:
              (din[5] ==1'b1 ) ? 3'b101:
              (din[4] ==1'b1) ? 3'b100:
              (din[3] ==1'b1) ? 3'b011:
              (din[2] ==1'b1) ? 3'b010:
              (din[1] ==1'b1) ? 3'b001:
              (din[0] ==1'b1) ? 3'b000: 3'bxxx;
endmodule
```

```
        g8 (wa3, din6_not, din[5]),
        g9 (wa4, din6_not, din4_not, din2_not, din[1]);
  or    g11(dout[2], din[7], din[6], din[5], din[4]),
        g12(dout[1], din[7], din[6], wa1, wa2),
        g13(dout[0], din[7], wa0, wa3, wa4),
        g14(V, din[0], din[1], din[2], din[3], din[4], din[5], din[6], din[7]);
endmodule
```

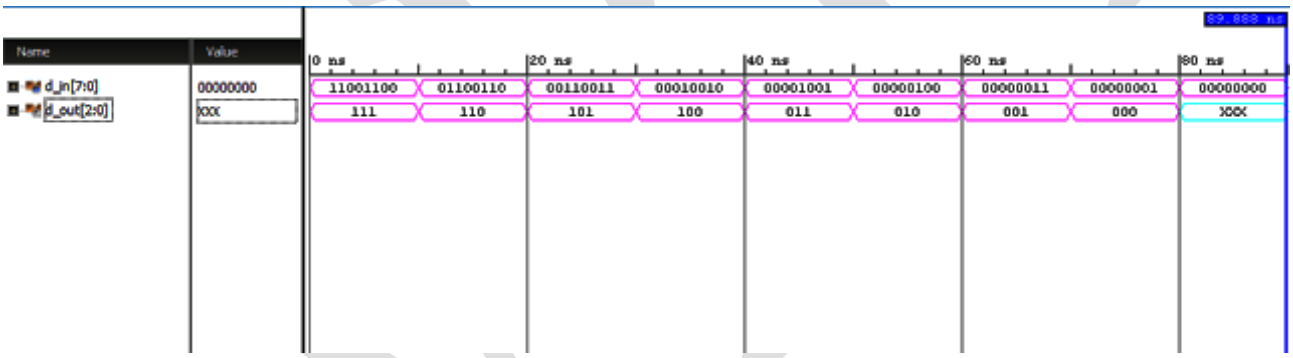
BehaviouralModel	**BehaviouralModel with Enable**

module encoder (din, dout);
input [7:0] din;
output [2:0] dout;
reg [2:0] dout;
always @(din)
begin
if (din ==8'b00000001) dout=3'b000;
else if (din==8'b0000001 X) dout=3'b001;
else if (din==8'b000001 XX) dout=3'b010;
else if (din==8'b00001XXX) dout=3'b011;
else if (din==8'b0001XXXX) dout=3'b100;
else if (din ==8'b001XXXXX) dout=3'b101;
else if (din==8'b01XXXXXX) dout=3'b110;
else if (din==8'b1XXXXXXX) dout=3'b111;
else dout=3'bX;
end
endmodule
	module priori (en,din,dout);
	input en;
	input [7 : 0] din;
	output [2 : 0] dout;
	reg [2 : 0] dout;
	always@(en,din)
	begin
	if(en == 1) // Active high enable
	begin
	dout = 3'bZZZ; // Initializing dout to high Impedance
	end
	else
	begin
	casex(din)
	8'b00000001 :dout = 3'b000;
	8'b0000001X :dout = 3'b001;
	8'b000001XX :dout = 3'b010;
	8'b00001XXX :dout = 3'b011;
	8'b0001XXXX :dout = 3'b100;
	8'b001XXXXX :dout = 3'b101;
	8'b01XXXXXX :dout = 3'b110;
	8'b1XXXXXXX :dout = 3'b111;
	endcase
	end
	end
	endmodule

Test Bench Code :

```
module prio_enco_8x3_tst;
  reg [7:0] d_in;
  wire[2:0] d_out;
  prio_enco_8x3 u1 (.d_out(d_out), .d_in(d_in) );
  initial
   begin
   d_in=8'b11001100;    #10;
   d_in=8'b01100110;    #10;
   d_in=8'b00110011;     #10;
   d_in=8'b00010010;    #10;
   d_in=8'b00001001;    #10;
   d_in=8'b00000100;    #10;
   d_in=8'b00000011;    #10;
   d_in=8'b00000001;    #10;
   d_in=8'b00000000;    # 10;
   $stop;
    end // initial begin
endmodule
```

Simulation output: Waveform window: Displays output waveform for verification.



RESULT:

Thus the OUTPUT of 8 to 3 decoder (without and with priority) is verified by simulating the VERILOG HDL code.