**LAB 3 – Raja Aadhithan**

Design – 4x1 mux using behavioral:

Code:

module mux4\_1(input [3:0] data\_in, [1:0]sel\_in, output y\_out);

reg x;

   //Step2 : Write the MUX behaviour as a parallel logic using "case"

always@(\*) begin

   case(sel\_in)

   2'b00 : x <= data\_in[0];

   2'b01 : x <= data\_in[1];

   2'b10 : x <= data\_in[2];

   2'b11 : x <= data\_in[3];

   endcase

end

assign y\_out = x;

endmodule

Testbench:

module mux4\_1\_tb();

      reg [3:0] a;

      reg[1:0] s;

      wire y;

      integer i;

mux4\_1 dut(a,s,y);

initial begin

   a = 4'b0000;

   s = 2'b00;

end

initial begin

   $monitor("@time %3d: select line: %b, data: %b, output: %b",$time,s,a,y);

   for (i=0;i<64;i=i+1)

   begin

      {s,a}=i;

      #10;

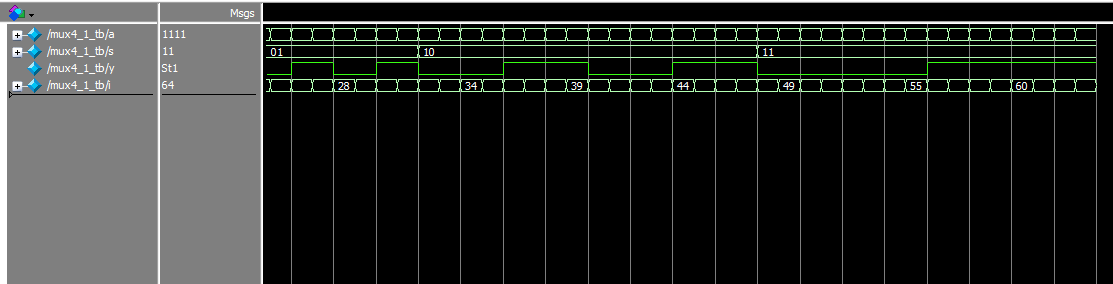
   end

   $finish;

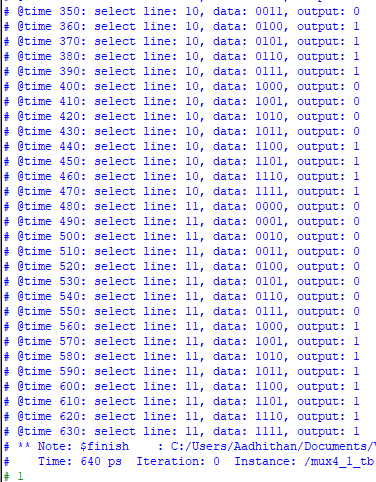
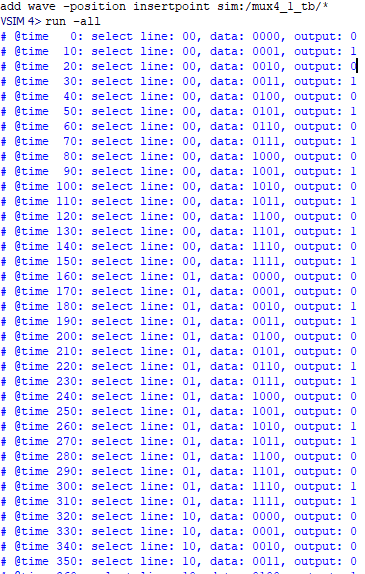
end

endmodule

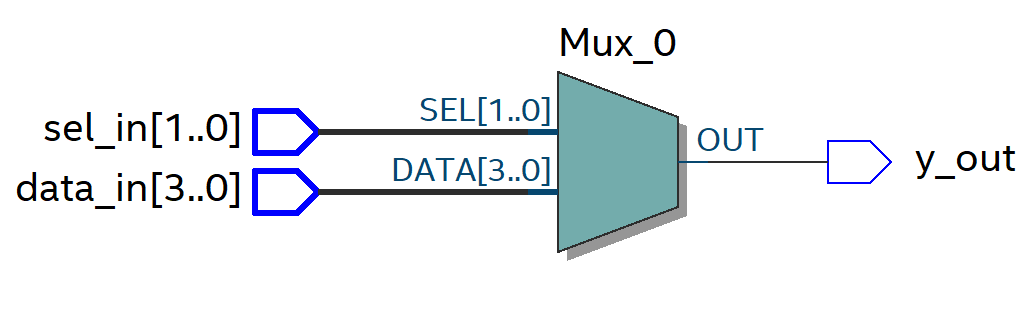
Wave:



Output:



RTL:



Design: 3:8 Decoder:

Code:

module decoder(input [2:0] in, output [7:0] out);

reg [7:0]temp;

always@(\*)begin

    case(in)

    3'd0: temp <= 8'd1;

    3'd1: temp <= 8'd2;

    3'd2: temp <= 8'd4;

    3'd3: temp <= 8'd8;

    3'd4: temp <= 8'd16;

    3'd5: temp <= 8'd32;

    3'd6: temp <= 8'd64;

    3'd7: temp <= 8'd128;

    default : temp <= 8'd0;

    endcase

end

assign out = temp;

endmodule;

Test bench:

module decoder\_tb();

reg [2:0]a;

wire [7:0]y;

integer i;

decoder dut(a,y);

initial begin

    $monitor("@time: %2dps - input is %b , output is %b",$time,a,y);

    for(i=0;i<8;i=i+1)begin

        a = i;

        #10;

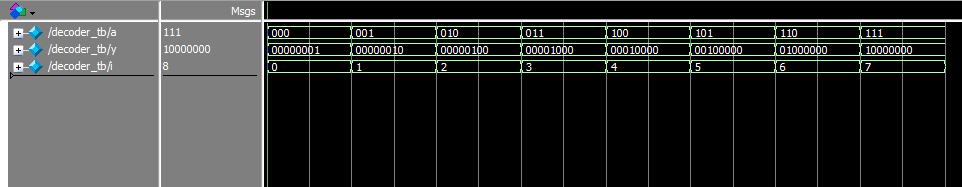
    end

    $finish;

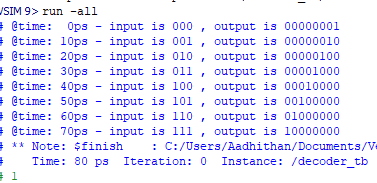
end

endmodule

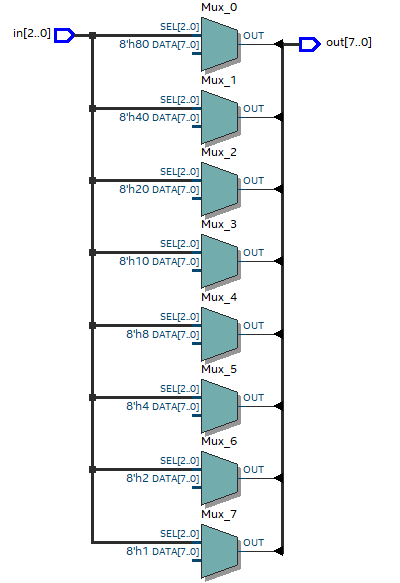
Wave



Output :



RTL:



Design: 8:3 priority encoder:

Code:

module encoder(input [7:0]in , output [2:0] out);

assign out[2] = in[7]|in[6]|in[5]|in[4];

assign out[1] = in[7]|in[6]| (~in[5]&~in[4]&(in[3]|in[2]));

assign out[0] = in[7] | (~in[6]&in[5]) | (~in[6]&~in[5]&~in[4]&~in[3]&~in[2]&in[1]) | (~in[6]&~in[5]&~in[4]&in[3]);

endmodule

Test bench:

module encoder\_tb();

reg [7:0] x;

wire [2:0] y;

integer i;

encoder dat(x,y);

initial begin

    for(i=0; i<8; i = i+1)

    begin

        x = 2\*\*i;

        #10;

        x=2\*i;

        #10;

    end

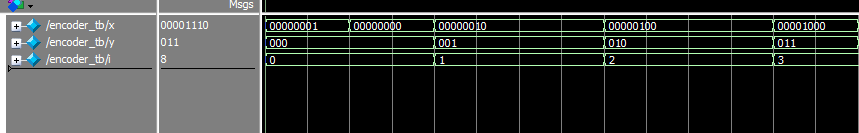
$finish;

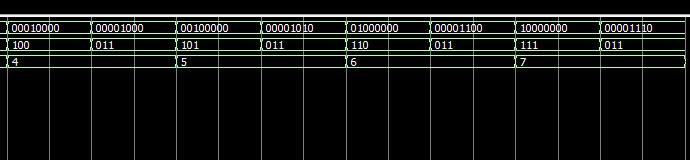
end

initial $monitor("@ time: %3dps the input is %8b output is %3b",$time,x,y);

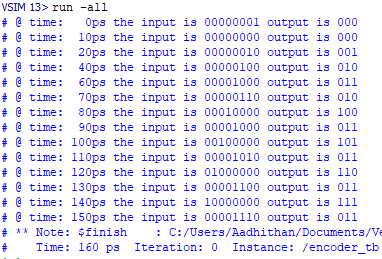
endmodule

Wave:





Output:



RTL :

