```
/////////////////Even Priority encoder////////////////////////
//code
module even_pri_enc(input [3:0] i,output reg [1:0] out);


  always @* begin
    case(i)
      4'b0000: out = 2'b00;
      4'b0010: out = 2'b01;
      4'b0011: out = 2'b01;
      4'b0100: out = 2'b10;
      4'b0101: out = 2'b10;
      4'b0110: out = 2'b10;
      4'b0111: out = 2'b10;
      4'b1000: out = 2'b11;
      default: out = 2'b11;
    endcase
  end


endmodule


//Test Bench
// Code your testbench here
// or browse Examples
module tb;
  reg [3:0] i;
```

```
 wire [1:0] out;


 even_pri_enc uut (.i(i),.out(out));


 initial begin
  $monitor("Time=%0t in_data=%b out=%b", $time, i, out);
  i = 4'b0000; #10;
  i = 4'b0010; #10;
  i = 4'b0100; #10;
  i = 4'b1000; #10;
  i = 4'b1111; #10;
  $finish;
 end


endmodule
```

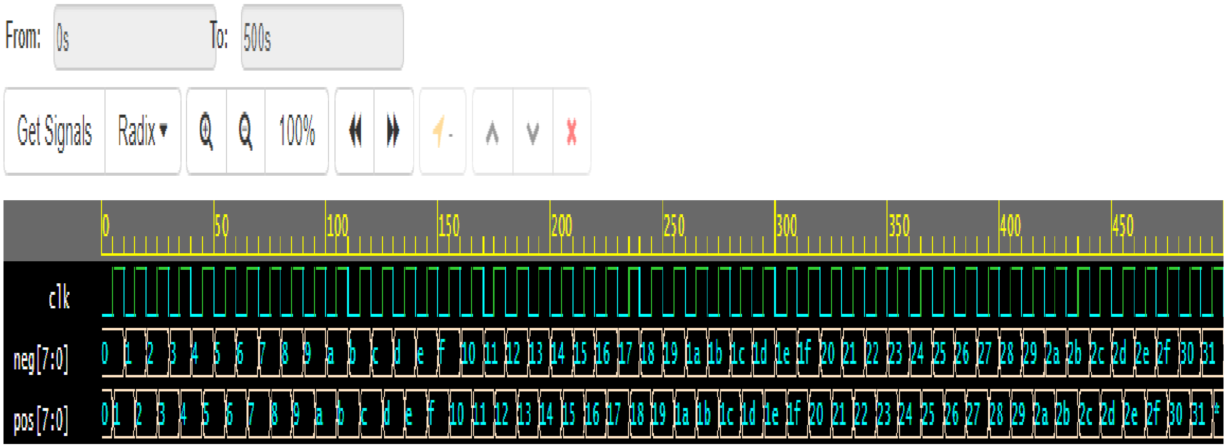
//

////////////////////////////////RISING AND FALLING EDGE COUNTER////////////////////////////////

```
//CODE
module edge_counter(
  input wire clk,
  output reg [7:0] pos,
  output reg [7:0] neg
);

  reg ss;
  initial
    begin
    pos<=0;
    neg<=0;
  end
  always @(posedge clk) begin
    if(ss==0)
      pos <= pos + 1;
    ss=1;
  end

  always @(negedge clk) begin
    if(ss==1)
      neg <= neg + 1;
    ss=0;
  end
```

```
endmodule
/// TEST BENCH



module edge_counter_tb;


 reg clk;
 wire [7:0] pos;
 wire [7:0] neg;


 edge_counter uut (
  .clk(clk),
  .pos(pos),
  .neg(neg)
 );


 initial begin
  clk=0;
  $monitor("clk=%b pos_count=%d neg_count=%d", clk, pos, neg);
  repeat (100) #5 clk = ~clk;
  $finish();
 end


endmodule
```

//

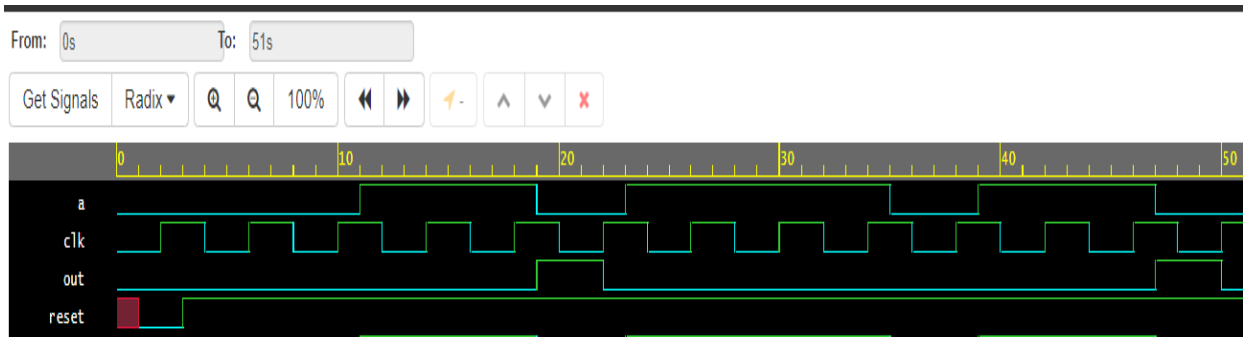



```
///////////////////////    FSM SEQ DETECTOR 0110//////////////////////////////////////////////////////////////

module seq0110(input bit clk, rst, a, output out);

 bit [3:0] st, n_st;


 always @(posedge clk or negedge rst) begin

  if (!rst) begin

   st <= 4'h1;

  end

  else begin

   st <= n_st;

  end

 end


 always @(st or a) begin
```

```
   case(st)
     4'h1: begin
       if (a == 0) n_st = 4'h2;
       else n_st = 4'h1;
     end
     4'h2: begin
       if (a== 1) n_st = 4'h3;
       else n_st = 4'h2;
     end
     4'h3: begin
       if (a == 1) n_st = 4'h4;
       else n_st = 4'h2;
     end
     4'h4: begin
       if (a == 0) n_st = 4'h1;
       else n_st = 4'h1;
     end
     default: n_st = 4'h1;
   endcase
  end

  assign out = (st == 4'h4) && (a == 0) ? 1 : 0;
endmodule

//TEST BENCH
```

```
module tb();
  reg clk, reset, a;
  wire out;

  seq0110 d(clk, reset, a, out);
  initial clk = 0;
  always #2 clk = ~clk;

  initial begin
    a = 0;
    #1 reset = 0;
    #2 reset = 1;



    #4 a = 0;
    #4 a = 1;
    #4 a = 1;
    #4 a = 0;

     #4 a = 1;
    #4 a = 1;
    #4 a = 1;
    #4 a = 0;

     #4 a = 1;
```

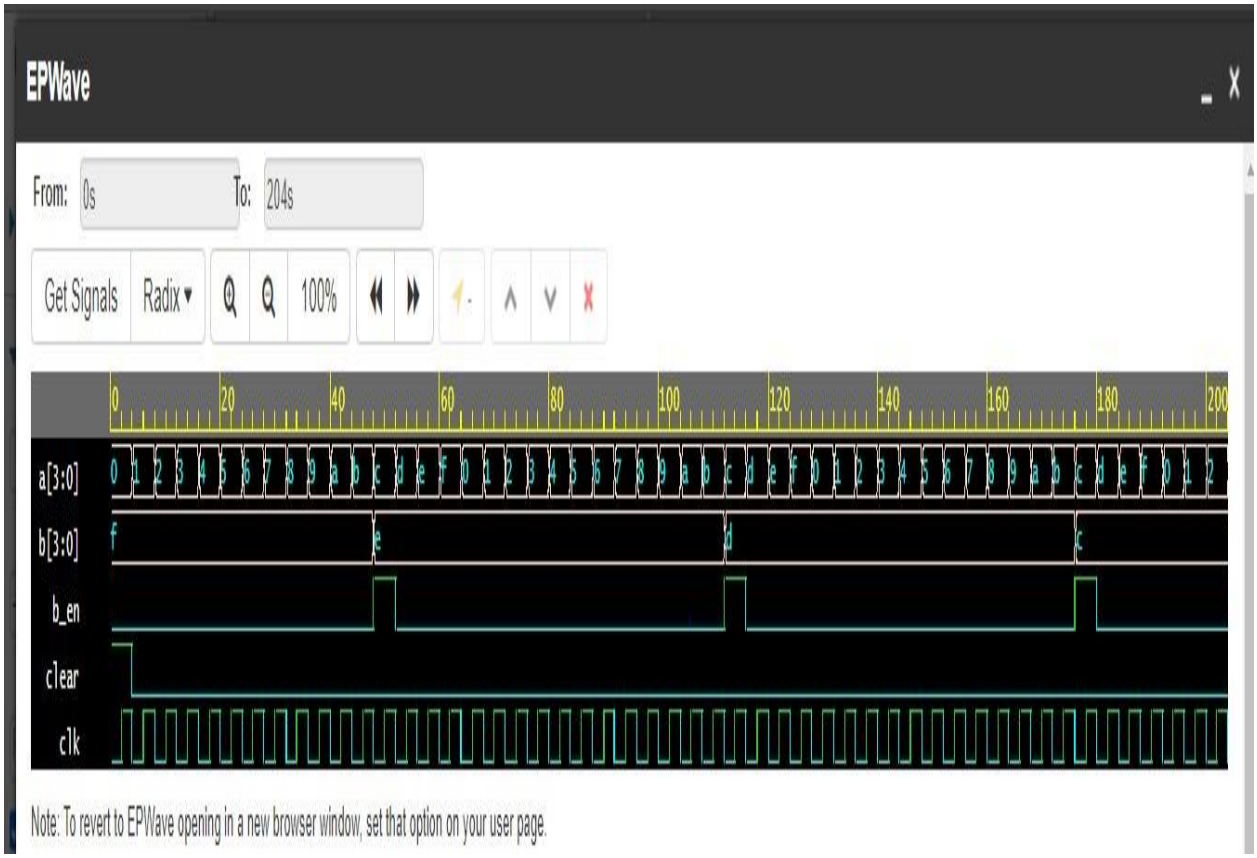
```
    #4 a = 1;
    #4 a = 0;
    #4 a = 0;



    $finish;
  end

  initial begin

    $dumpfile("dump.vcd");
    $dumpvars(0);
  end
endmodule
```



```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/////////////

//////////////Counter

// Code your design here

module counter(

 input clk,

 input clr,

 output reg [3:0] y1, y2,

 output reg enable

);

 reg o1, o2;


 always @(negedge clk)

  if (clr)

    begin

      y1 <= 4'b0000;

      y2 <= 4'b1111;

    end

  else

    begin

      y1[0] <= ~y1[0];

      if (y1[0]) begin

        y1[1] <= ~y1[1];

        if (y1[1]) begin

          y1[2] <= ~y1[2];
```

```
          if (y1[2]) begin
            y1[3] <= ~y1[3];
          end
        end
      end
    end

  assign o1 = ~(y1[0] | y1[1]);
  assign o2 = y1[2] & y1[3];
  assign enable = o1 & o2;

  always @(posedge enable)
    begin
      y2[0] <= ~y2[0];
      if (~y2[0]) begin
        y2[1] <= ~y2[1];
        if (~y2[1]) begin
          y2[2] <= ~y2[2];
          if (~y2[2]) begin
            y2[3] <= ~y2[3];
          end
        end
      end
    end
endmodule
```

```
////////////////TEST BENCH
// Code your testbench here
// or browse Examples
module tb();
 reg clk;
 reg clear;
 wire [3:0]a,b;
 wire b_en;
 counter UUT(clk,clear,a,b,b_en);
 initial
  begin
   clk=0;
   forever #2 clk=~clk;
  end
 initial
  begin
   $monitor("Time=%0t, a=%b,b=%b,b_en=%b", $time,a,b,b_en);
   clear=1;
   #4 clear=0;
   #200 $finish;
  end
endmodule
////////////////////////////////////////////////////////////////////////////////////////////////////
/////////
```



//

```
module Multiplier3bit(
  input wire [2:0] a1,
  input wire [2:0] b1,
  output reg [5:0] y1
);
  reg [5:0] y1_pro;
  initial begin
    y1_pro=0;
```

```
  y1=0;
 end
 always @(a1 or b1) begin
  for (int i = 0; i < 3; i = i + 1) begin
   y1_pro = y1_pro + (a1[i]*(b1<<i));
  end
  y1 = y1_pro;
 end
endmodule
```

TESET BENCH

```
module testbench;

 reg [2:0] a1, b1;
 wire [5:0] y1;

 Multiplier3bit  uut(
  .a1(a1),
  .b1(b1),
  .y1(y1)
 );

 initial begin
```

```
  $dumpfile("dump.vcd"); $dumpvars(1);
  a1 = 3'b101;
  b1 = 3'b011;
  $display("a1=%b b1=%b y1=%b", a1, b1, y1);

  a1 = 3'b110;
  b1 = 3'b010;
  $display("a1=%b b1=%b y1=%b", a1, b1, y1);

  #100 $finish();
 end

endmodule
```