Verilog Code

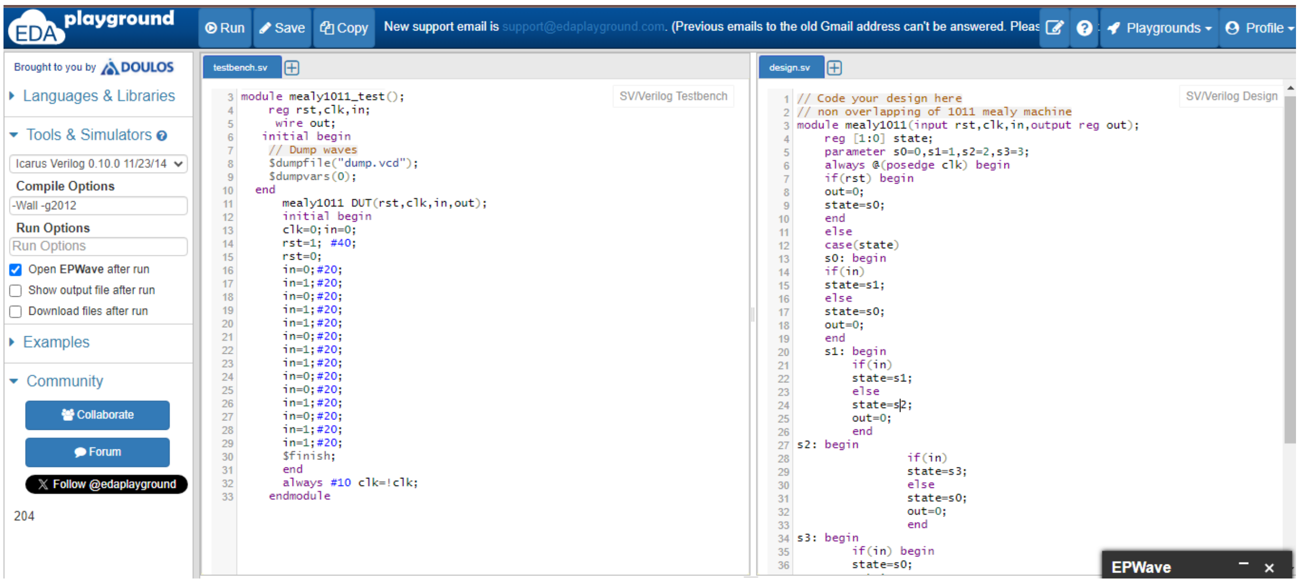
```
// non overlapping of 1011 mealy machine

module mealy1011(input rst,clk,in,output reg out);

   reg [1:0] state;

   parameter s0=0,s1=1,s2=2,s3=3;

   always @(posedge clk) begin

   if(rst) begin

   out=0;

   state=s0;

   end

   else

   case(state)

   s0: begin

   if(in)

   state=s1;

   else

   state=s0;

   out=0;

   end

   s1: begin

      if(in)

      state=s1;

      else

      state=s2;

      out=0;

      end

s2: begin
```

```
                if(in)
                state=s3;
                else
                state=s0;
                out=0;
                end
s3: begin
        if(in) begin
        state=s0;
        out=1;
        end
        else begin
        state=s2;
        out=0;
        end
        end

    endcase
    end
    endmodule
```

Testbench

```
module mealy1011_test();
    reg rst,clk,in;
     wire out;
   initial begin
    // Dump waves
```

```
  $dumpfile("dump.vcd");
  $dumpvars(0);
 end
    mealy1011 DUT(rst,clk,in,out);
    initial begin
    clk=0;in=0;
    rst=1; #40;
    rst=0;
    in=0;#20;
    in=1;#20;
    in=0;#20;
    in=1;#20;
    in=1;#20;
    in=0;#20;
    in=1;#20;
    in=1;#20;
    in=0;#20;
    in=0;#20;
    in=1;#20;
    in=0;#20;
    in=1;#20;
    in=1;#20;
    $finish;
    end
    always #10 clk=!clk;
  endmodule
```



New support email is support@edaplayground.com. (Previous emails to the old Gmail address can't be answered. Pleas

Icarus Verilog 0.10.0 11/23/14

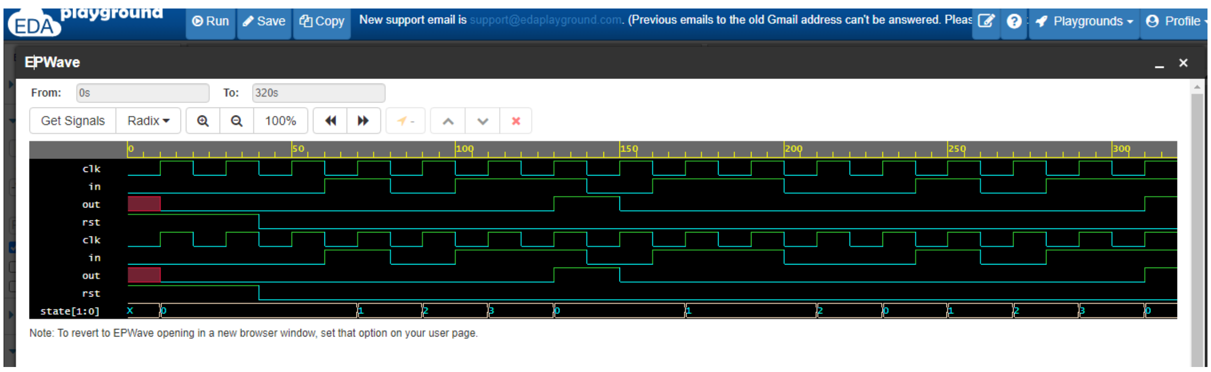
Compile Options: -Wall -g2012

Run Options

[✓] Open EPWave after run
[ ] Show output file after run
[ ] Download files after run

```
module mealy1011_test();
    reg rst,clk,in;
     wire out;
   initial begin
    // Dump waves
    $dumpfile("dump.vcd");
    $dumpvars(0);
  end
      mealy1011 DUT(rst,clk,in,out);
      initial begin
      clk=0;in=0;
      rst=1; #40;
      rst=0;
      in=0;#20;
      in=1;#20;
      in=0;#20;
      in=1;#20;
      in=1;#20;
      in=0;#20;
      in=1;#20;
      in=1;#20;
      in=0;#20;
      in=0;#20;
      in=1;#20;
      in=0;#20;
      in=1;#20;
      in=1;#20;
      $finish;
      end
      always #10 clk=!clk;
    endmodule
```

```
// Code your design here
// non overlapping of 1011 mealy machine
module mealy1011(input rst,clk,in,output reg out);
    reg [1:0] state;
    parameter s0=0,s1=1,s2=2,s3=3;
    always @(posedge clk) begin
    if(rst) begin
    out=0;
    state=s0;
    end
    else
    case(state)
    s0: begin
    if(in)
    state=s1;
    else
    state=s0;
    out=0;
    end
    s1: begin
        if(in)
        state=s1;
        else
        state=s2;
        out=0;
        end
s2: begin
                if(in)
                state=s3;
                else
                state=s0;
                out=0;
                end
s3: begin
        if(in) begin
        state=s0;
```



New support email is support@edaplayground.com. (Previous emails to the old Gmail address can't be answered. Pleas

From: 0s To: 320s

Note: To revert to EPWave opening in a new browser window, set that option on your user page.

Verilog code for Moore sequence of 1011

module seq_detector(

input x,clk,reset,

output reg z

);

parameter S0 = 0 , S1 = 1 , S2 = 2 , S3 = 3 , S4 = 4;

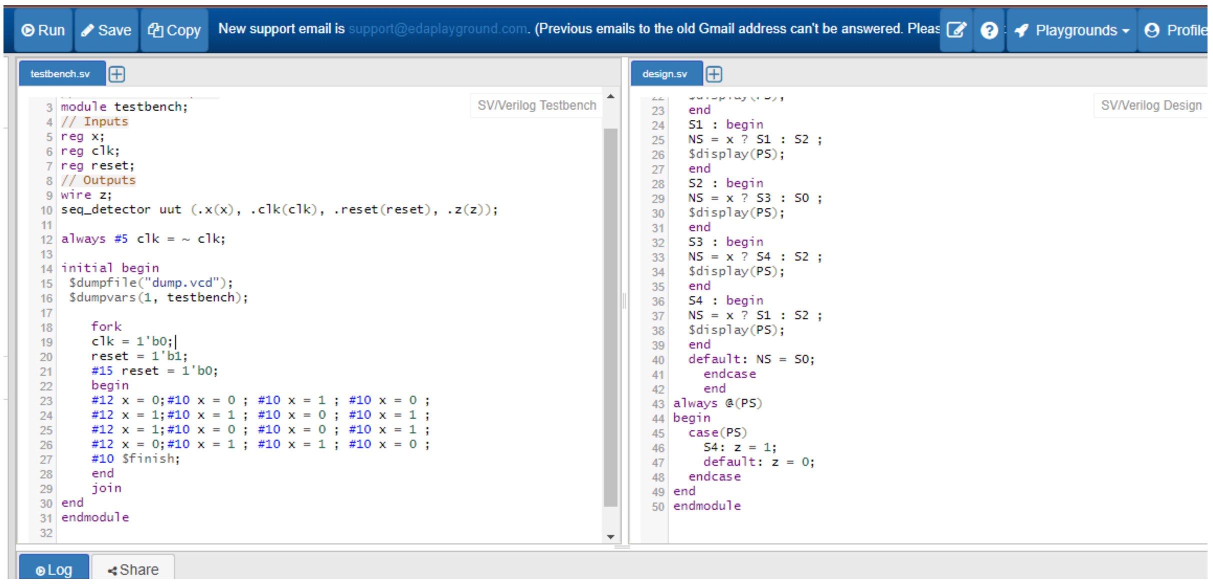
reg [2:0] PS,NS ;

always @(posedge clk or posedge reset)

```
    begin
        if(reset)
            PS <= S0;
        else
            PS <= NS ;
    end
always @(PS, x)
begin
case(PS)
  S0 : begin
  NS = x ? S1 : S0 ;
  $display(PS);
  end
  S1 : begin
  NS = x ? S1 : S2 ;
  $display(PS);
  end
  S2 : begin
  NS = x ? S3 : S0 ;
  $display(PS);
  end
  S3 : begin
  NS = x ? S4 : S2 ;
  $display(PS);
  end
  S4 : begin
  NS = x ? S1 : S2 ;
  $display(PS);
  end
  default: NS = S0;
    endcase
    end
always @(PS)
begin
  case(PS)
    S4: z = 1;
    default: z = 0;
  endcase
end
endmodule
```

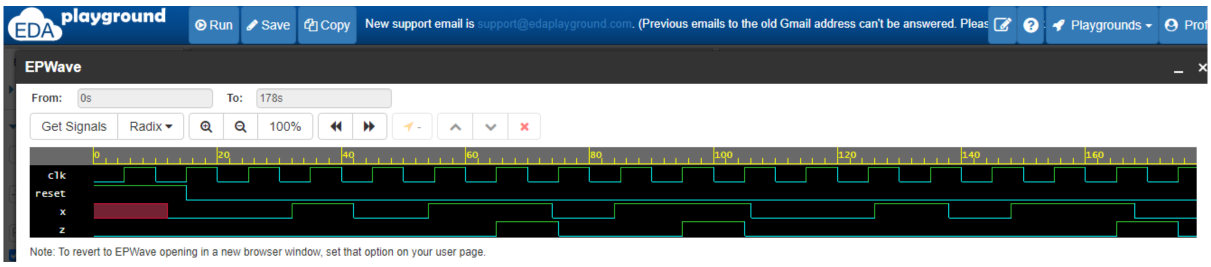
Testbench

```
module testbench;
// Inputs
reg x;
reg clk;
reg reset;
// Outputs
wire z;
seq_detector uut (.x(x), .clk(clk), .reset(reset), .z(z));

always #5 clk = ~ clk;

initial begin
 $dumpfile("dump.vcd");
 $dumpvars(1, testbench);


   fork
   clk = 1'b0;
   reset = 1'b1;
   #15 reset = 1'b0;
   begin
   #12 x = 0;#10 x = 0 ; #10 x = 1 ; #10 x = 0 ;
   #12 x = 1;#10 x = 1 ; #10 x = 0 ; #10 x = 1 ;
   #12 x = 1;#10 x = 0 ; #10 x = 0 ; #10 x = 1 ;
   #12 x = 0;#10 x = 1 ; #10 x = 1 ; #10 x = 0 ;
   #10 $finish;
   end
   join
end
endmodule
```


Run
Save
Copy
New support email is support@edaplayground.com. (Previous emails to the old Gmail address can't be answered. Pleas
Playgrounds
Profile
testbench.sv
SV/Verilog Testbench
design.sv
SV/Verilog Design
Log
Share

```
module testbench;
// Inputs
reg x;
reg clk;
reg reset;
// Outputs
wire z;
seq_detector uut (.x(x), .clk(clk), .reset(reset), .z(z));

always #5 clk = ~ clk;

initial begin
 $dumpfile("dump.vcd");
 $dumpvars(1, testbench);

    fork
    clk = 1'b0;
    reset = 1'b1;
    #15 reset = 1'b0;
    begin
    #12 x = 0;#10 x = 0 ; #10 x = 1 ; #10 x = 0 ;
    #12 x = 1;#10 x = 1 ; #10 x = 0 ; #10 x = 1 ;
    #12 x = 1;#10 x = 0 ; #10 x = 0 ; #10 x = 1 ;
    #12 x = 0;#10 x = 1 ; #10 x = 1 ; #10 x = 0 ;
    #10 $finish;
    end
    join
end
endmodule
```

```
  end
  S1 : begin
  NS = x ? S1 : S2 ;
  $display(PS);
  end
  S2 : begin
  NS = x ? S3 : S0 ;
  $display(PS);
  end
  S3 : begin
  NS = x ? S4 : S2 ;
  $display(PS);
  end
  S4 : begin
  NS = x ? S1 : S2 ;
  $display(PS);
  end
  default: NS = S0;
    endcase
    end
always @(PS)
begin
  case(PS)
    S4: z = 1;
    default: z = 0;
  endcase
end
endmodule
```


playground
Run
Save
Copy
New support email is support@edaplayground.com. (Previous emails to the old Gmail address can't be answered. Pleas
Playgrounds
EPWave
From: 0s
To: 178s
Get Signals
Radix
100%
0
20
40
60
80
100
120
140
160
clk
reset
x
z

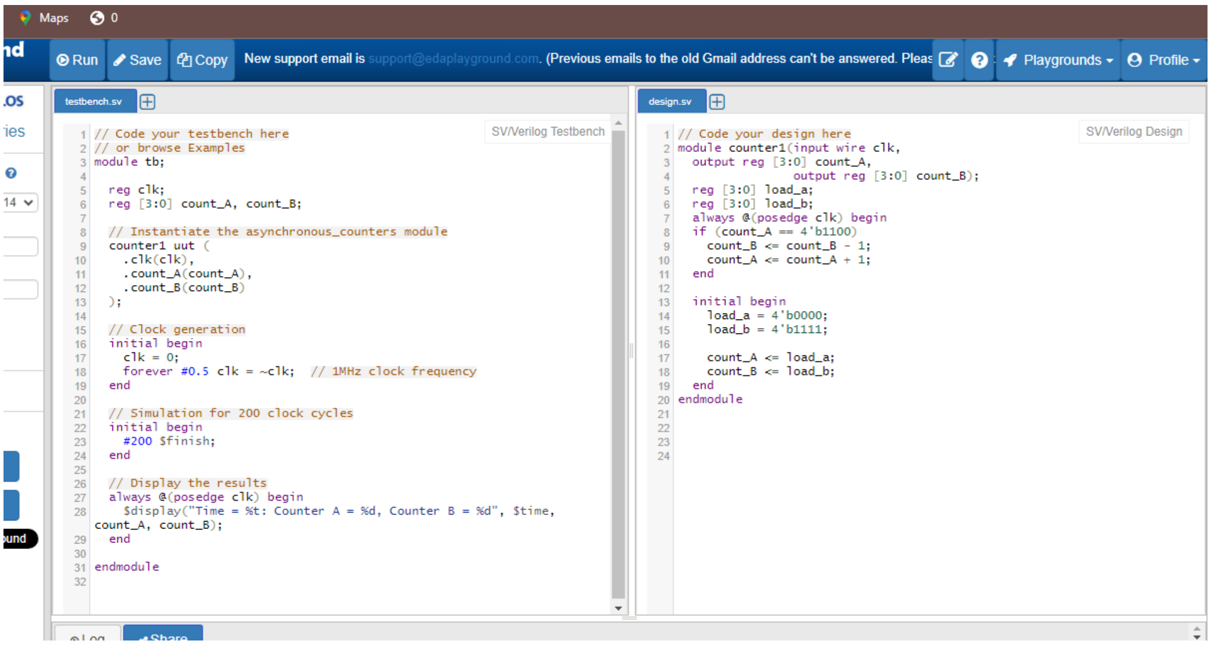
Note: To revert to EPWave opening in a new browser window, set that option on your user page.

Verilog Code for Asynchronous Counters

```
module counter1(input wire clk,
  output reg [3:0] count_A,
              output reg [3:0] count_B);
  reg [3:0] load_a;
  reg [3:0] load_b;
  always @(posedge clk) begin
  if (count_A == 4'b1100)
    count_B <= count_B - 1;
    count_A <= count_A + 1;
  end

  initial begin
    load_a = 4'b0000;
    load_b = 4'b1111;

    count_A <= load_a;
    count_B <= load_b;
  end
endmodule
```

Testbench

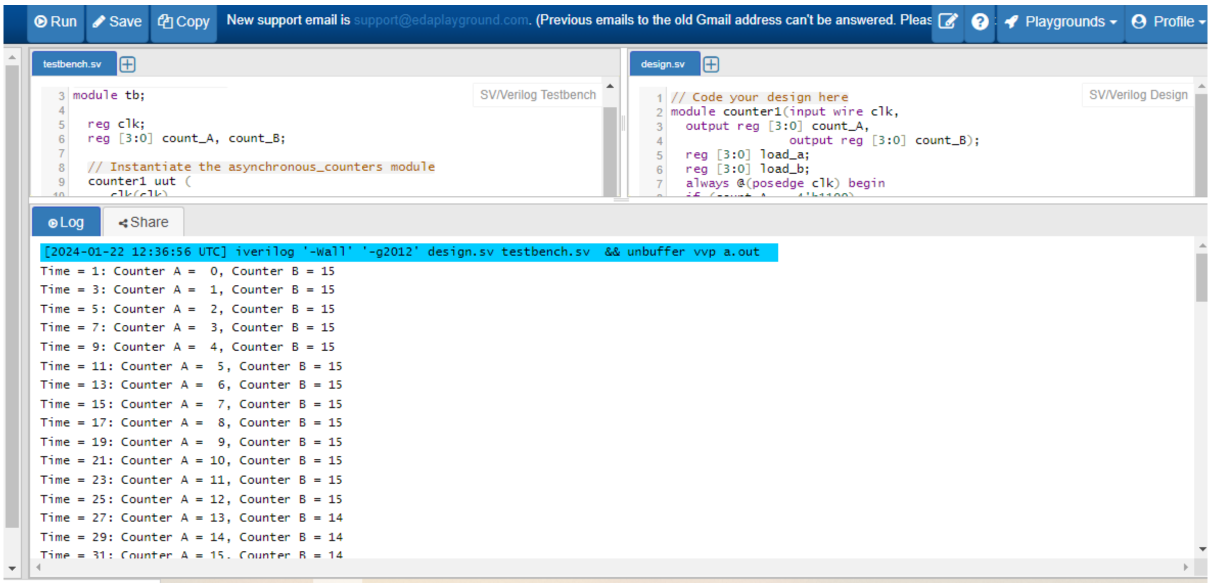
```
module tb;
 reg clk;
  reg [3:0] count_A, count_B;
  // Instantiate the asynchronous_counters module
  counter1 uut (
    .clk(clk),
    .count_A(count_A),
    .count_B(count_B)
  );

  // Clock generation
  initial begin
    clk = 0;
    forever #0.5 clk = ~clk;  // 1MHz clock frequency
  end

  // Simulation for 200 clock cycles
  initial begin
    #200 $finish;
  end

  // Display the results
  always @(posedge clk) begin
    $display("Time = %0d :Counter A = %d, Counter B = %d", $time, count_A, count_B);
  end
endmodule
```



Maps
Run
Save
Copy
New support email is support@edaplayground.com. (Previous emails to the old Gmail address can't be answered. Pleas
Playgrounds
Profile
testbench.sv
design.sv
SV/Verilog Testbench
SV/Verilog Design
Log
Share

```verilog
// Code your testbench here
// or browse Examples
module tb;

  reg clk;
  reg [3:0] count_A, count_B;

  // Instantiate the asynchronous_counters module
  counter1 uut (
    .clk(clk),
    .count_A(count_A),
    .count_B(count_B)
  );

  // Clock generation
  initial begin
    clk = 0;
    forever #0.5 clk = ~clk;  // 1MHz clock frequency
  end

  // Simulation for 200 clock cycles
  initial begin
    #200 $finish;
  end

  // Display the results
  always @(posedge clk) begin
    $display("Time = %t: Counter A = %d, Counter B = %d", $time,
count_A, count_B);
  end

endmodule
```

```verilog
// Code your design here
module counter1(input wire clk,
  output reg [3:0] count_A,
              output reg [3:0] count_B);
  reg [3:0] load_a;
  reg [3:0] load_b;
  always @(posedge clk) begin
  if (count_A == 4'b1100)
    count_B <= count_B - 1;
    count_A <= count_A + 1;
  end

  initial begin
    load_a = 4'b0000;
    load_b = 4'b1111;

    count_A <= load_a;
    count_B <= load_b;
  end
endmodule
```



Run
Save
Copy
New support email is support@edaplayground.com. (Previous emails to the old Gmail address can't be answered. Pleas
Playgrounds
Profile
testbench.sv
design.sv
SV/Verilog Testbench
SV/Verilog Design
Log
Share

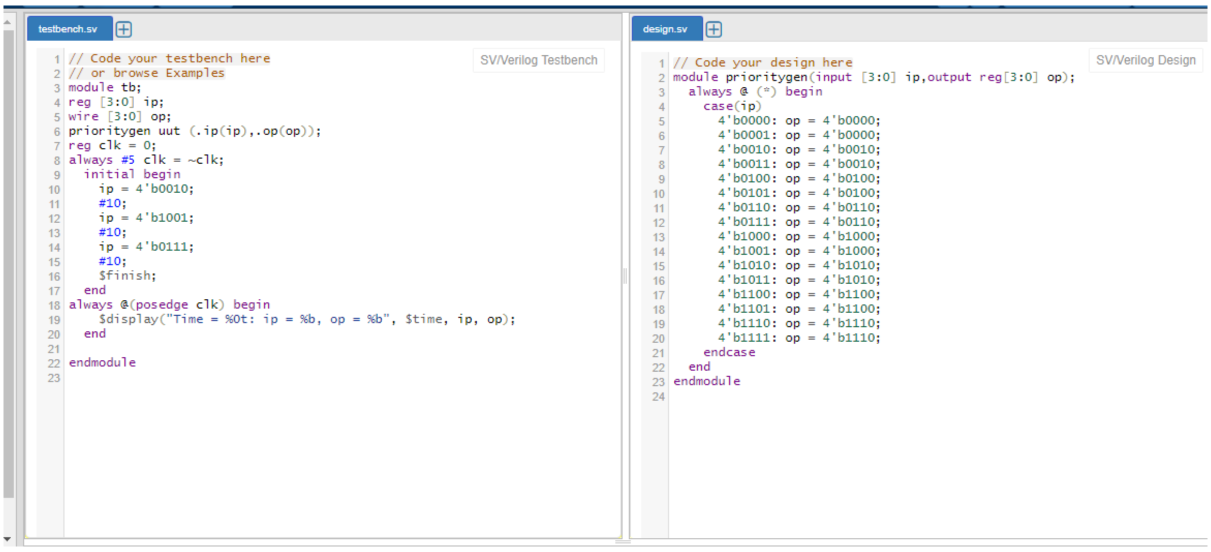
```
[2024-01-22 12:36:56 UTC] iverilog '-Wall' '-g2012' design.sv testbench.sv  && unbuffer vvp a.out
Time = 1: Counter A =  0, Counter B = 15
Time = 3: Counter A =  1, Counter B = 15
Time = 5: Counter A =  2, Counter B = 15
Time = 7: Counter A =  3, Counter B = 15
Time = 9: Counter A =  4, Counter B = 15
Time = 11: Counter A =  5, Counter B = 15
Time = 13: Counter A =  6, Counter B = 15
Time = 15: Counter A =  7, Counter B = 15
Time = 17: Counter A =  8, Counter B = 15
Time = 19: Counter A =  9, Counter B = 15
Time = 21: Counter A = 10, Counter B = 15
Time = 23: Counter A = 11, Counter B = 15
Time = 25: Counter A = 12, Counter B = 15
Time = 27: Counter A = 13, Counter B = 14
Time = 29: Counter A = 14, Counter B = 14
Time = 31: Counter A = 15, Counter B = 14
```

<u>Verilog Code for Even priority encoder</u>

```verilog
module prioritygen(input [3:0] ip,output reg[3:0] op);

  always @ (*) begin

    case(ip)

      4'b0000: op = 4'b0000;

      4'b0001: op = 4'b0000;

      4'b0010: op = 4'b0010;

      4'b0011: op = 4'b0010;
      4'b0100: op = 4'b0100;
      4'b0101: op = 4'b0100;
      4'b0110: op = 4'b0110;
      4'b0111: op = 4'b0110;
      4'b1000: op = 4'b1000;
      4'b1001: op = 4'b1000;
      4'b1010: op = 4'b1010;
      4'b1011: op = 4'b1010;
      4'b1100: op = 4'b1100;
      4'b1101: op = 4'b1100;
      4'b1110: op = 4'b1110;
      4'b1111: op = 4'b1110;
    endcase
  end
endmodule
```

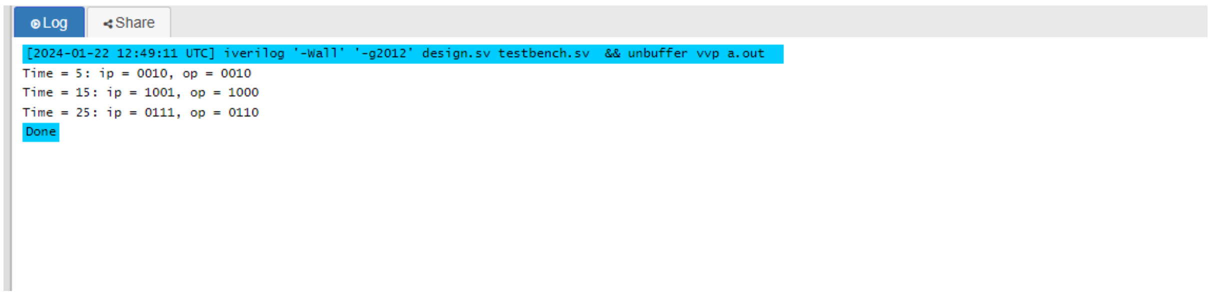
<u>Testbench</u>

```
module tb;
reg [3:0] ip;
wire [3:0] op;
prioritygen uut (.ip(ip),.op(op));
reg clk = 0;
always #5 clk = ~clk;
  initial begin
    ip = 4'b0010;
    #10;
    ip = 4'b1001;
    #10;
    ip = 4'b0111;

    #10;

    $finish;

  end

always @(posedge clk) begin

    $display("Time = %0t: ip = %b, op = %b", $time, ip, op);

  end

endmodule
```



```
// Code your testbench here
// or browse Examples
module tb;
reg [3:0] ip;
wire [3:0] op;
prioritygen uut (.ip(ip),.op(op));
reg clk = 0;
always #5 clk = ~clk;
  initial begin
    ip = 4'b0010;
    #10;
    ip = 4'b1001;
    #10;
    ip = 4'b0111;
    #10;
    $finish;
  end
always @(posedge clk) begin
    $display("Time = %0t: ip = %b, op = %b", $time, ip, op);
  end

endmodule
```

```
// Code your design here
module prioritygen(input [3:0] ip,output reg[3:0] op);
  always @ (*) begin
    case(ip)
      4'b0000: op = 4'b0000;
      4'b0001: op = 4'b0000;
      4'b0010: op = 4'b0010;
      4'b0011: op = 4'b0010;
      4'b0100: op = 4'b0100;
      4'b0101: op = 4'b0100;
      4'b0110: op = 4'b0110;
      4'b0111: op = 4'b0110;
      4'b1000: op = 4'b1000;
      4'b1001: op = 4'b1000;
      4'b1010: op = 4'b1010;
      4'b1011: op = 4'b1010;
      4'b1100: op = 4'b1100;
      4'b1101: op = 4'b1100;
      4'b1110: op = 4'b1110;
      4'b1111: op = 4'b1110;
    endcase
  end
endmodule
```



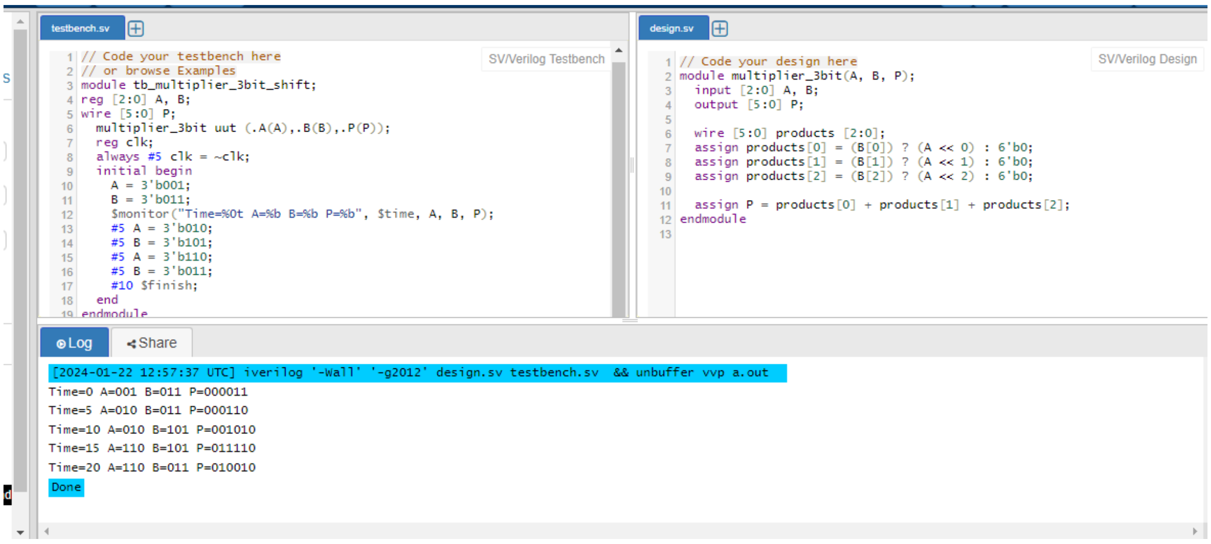
```
[2024-01-22 12:49:11 UTC] iverilog '-Wall' '-g2012' design.sv testbench.sv  && unbuffer vvp a.out
Time = 5: ip = 0010, op = 0010
Time = 15: ip = 1001, op = 1000
Time = 25: ip = 0111, op = 0110
Done
```

<u>Verilog Code for 3-bit Multiplier using Shift register</u>

```
module multiplier_3bit(A, B, P);

  input [2:0] A, B;

  output [5:0] P;
  wire [5:0] products [2:0];

  assign products[0] = (B[0]) ? (A << 0) : 6'b0;

  assign products[1] = (B[1]) ? (A << 1) : 6'b0;

  assign products[2] = (B[2]) ? (A << 2) : 6'b0;


  assign P = products[0] + products[1] + products[2];

endmodule
```

Testbench

```
module tb_multiplier_3bit_shift;

reg [2:0] A, B;

wire [5:0] P;

  multiplier_3bit uut (.A(A),.B(B),.P(P));

  reg clk;

  always #5 clk = ~clk;

  initial begin

    A = 3'b001;

    B = 3'b011;

    $monitor("Time=%0t A=%b B=%b P=%b", $time, A, B, P);

    #5 A = 3'b010;

    #5 B = 3'b101;

    #5 A = 3'b110;

    #5 B = 3'b011;

    #10 $finish;

  end

endmodule
```



```
// Code your testbench here
// or browse Examples
module tb_multiplier_3bit_shift;
reg [2:0] A, B;
wire [5:0] P;
  multiplier_3bit uut (.A(A),.B(B),.P(P));
  reg clk;
  always #5 clk = ~clk;
  initial begin
    A = 3'b001;
    B = 3'b011;
    $monitor("Time=%0t A=%b B=%b P=%b", $time, A, B, P);
    #5 A = 3'b010;
    #5 B = 3'b101;
    #5 A = 3'b110;
    #5 B = 3'b011;
    #10 $finish;
  end
endmodule
```

```
// Code your design here
module multiplier_3bit(A, B, P);
  input [2:0] A, B;
  output [5:0] P;

  wire [5:0] products [2:0];
  assign products[0] = (B[0]) ? (A << 0) : 6'b0;
  assign products[1] = (B[1]) ? (A << 1) : 6'b0;
  assign products[2] = (B[2]) ? (A << 2) : 6'b0;

  assign P = products[0] + products[1] + products[2];
endmodule
```

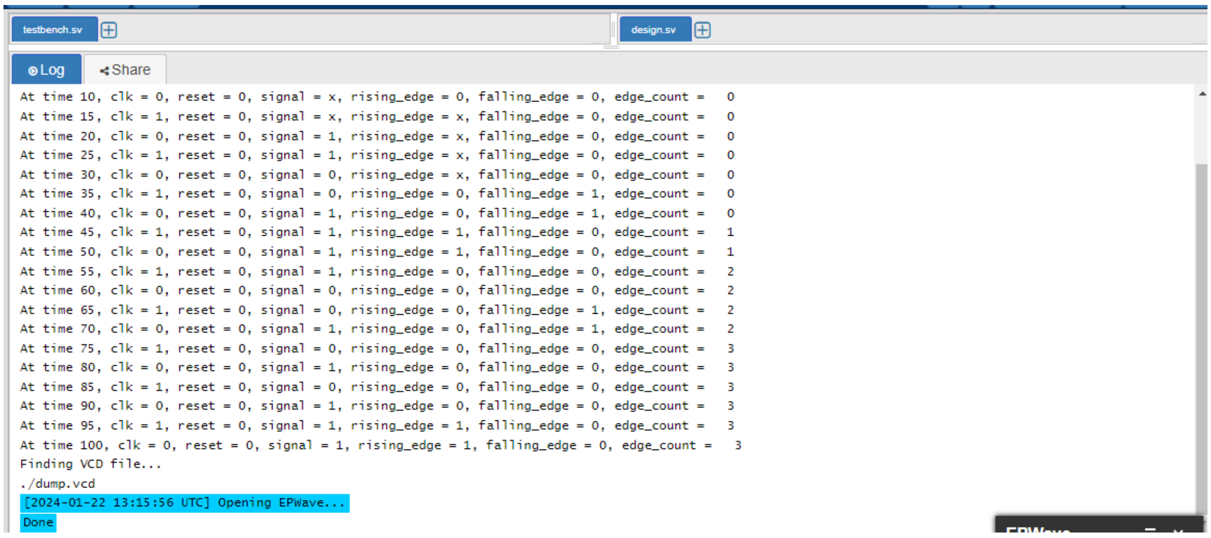
```
[2024-01-22 12:57:37 UTC] iverilog '-Wall' '-g2012' design.sv testbench.sv  && unbuffer vvp a.out
Time=0 A=001 B=011 P=000011
Time=5 A=010 B=011 P=000110
Time=10 A=010 B=101 P=001010
Time=15 A=110 B=101 P=011110
Time=20 A=110 B=011 P=010010
Done
```

<u>Verilog Code for counting of rising and falling edges and total count</u>

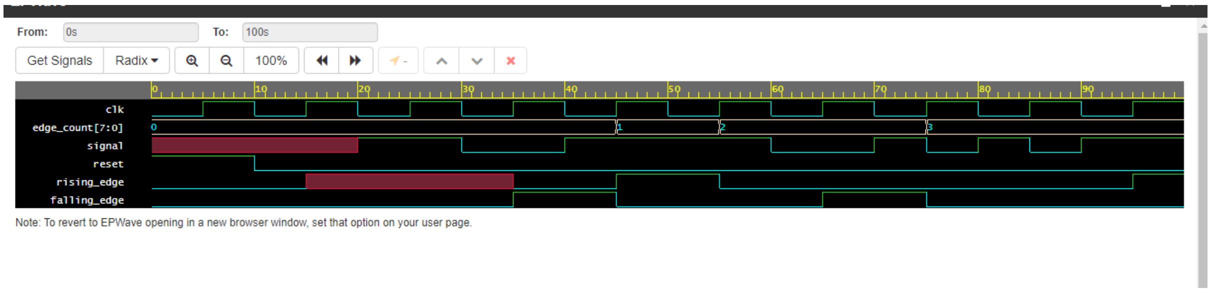
```
module risingandfalling  (
  input clk,
  input reset,
  input signal,
  output reg rising_edge,
  output reg falling_edge,
  output reg [7:0] edge_count
);
reg prev_signal;
always @(posedge clk or posedge reset) begin
    if (reset) begin
      prev_signal <= 0;
      rising_edge <= 0;
      falling_edge <= 0;
      edge_count <= 0;
    end else begin
      prev_signal <= signal;
    rising_edge <= (signal & ~prev_signal);
    falling_edge <= (~signal & prev_signal);
     if (rising_edge | falling_edge) begin
       edge_count <= edge_count + 1;
     end
    end
  end
endmodule
```

<u>Testbench</u>

```
module tb;

  reg clk, reset, signal;
  wire rising_edge, falling_edge;
  wire [7:0] edge_count;

  risingandfalling dut (.clk(clk),.reset(reset),.signal(signal),
    .rising_edge(rising_edge),.falling_edge(falling_edge),
    .edge_count(edge_count));
initial begin
  $dumpfile("dump.vcd");
  $dumpvars;
end
  initial begin
    clk = 1'b0;
    forever #5 clk = ~clk;
  end
  initial begin
    reset = 1'b1;
    #10 reset = 1'b0;

    #10 signal = 1'b1;
    #10 signal = 1'b0;
    #10 signal = 1'b1;
    #20 signal = 1'b0;
    #10 signal = 1'b1;
    #5 signal = 1'b0;
    #5 signal = 1'b1;
    #5 signal = 1'b0;
    #5 signal = 1'b1;



    #10 $finish;
  end

  initial begin
    $monitor("At time %t, clk = %b, reset = %b, signal = %b, rising_edge = %b, falling_edge = %b, 
edge_count = %d",
             $time, clk, reset, signal, rising_edge, falling_edge, edge_count);

  end

endmodule
```



```
At time 10, clk = 0, reset = 0, signal = x, rising_edge = 0, falling_edge = 0, edge_count =   0
At time 15, clk = 1, reset = 0, signal = x, rising_edge = x, falling_edge = 0, edge_count =   0
At time 20, clk = 0, reset = 0, signal = 1, rising_edge = x, falling_edge = 0, edge_count =   0
At time 25, clk = 1, reset = 0, signal = 1, rising_edge = x, falling_edge = 0, edge_count =   0
At time 30, clk = 0, reset = 0, signal = 0, rising_edge = x, falling_edge = 0, edge_count =   0
At time 35, clk = 1, reset = 0, signal = 0, rising_edge = 0, falling_edge = 1, edge_count =   0
At time 40, clk = 0, reset = 0, signal = 1, rising_edge = 0, falling_edge = 1, edge_count =   0
At time 45, clk = 1, reset = 0, signal = 1, rising_edge = 1, falling_edge = 0, edge_count =   1
At time 50, clk = 0, reset = 0, signal = 1, rising_edge = 1, falling_edge = 0, edge_count =   1
At time 55, clk = 1, reset = 0, signal = 1, rising_edge = 0, falling_edge = 0, edge_count =   2
At time 60, clk = 0, reset = 0, signal = 0, rising_edge = 0, falling_edge = 0, edge_count =   2
At time 65, clk = 1, reset = 0, signal = 0, rising_edge = 0, falling_edge = 1, edge_count =   2
At time 70, clk = 0, reset = 0, signal = 1, rising_edge = 0, falling_edge = 1, edge_count =   2
At time 75, clk = 1, reset = 0, signal = 0, rising_edge = 0, falling_edge = 0, edge_count =   3
At time 80, clk = 0, reset = 0, signal = 1, rising_edge = 0, falling_edge = 0, edge_count =   3
At time 85, clk = 1, reset = 0, signal = 0, rising_edge = 0, falling_edge = 0, edge_count =   3
At time 90, clk = 0, reset = 0, signal = 1, rising_edge = 0, falling_edge = 0, edge_count =   3
At time 95, clk = 1, reset = 0, signal = 1, rising_edge = 1, falling_edge = 0, edge_count =   3
At time 100, clk = 0, reset = 0, signal = 1, rising_edge = 1, falling_edge = 0, edge_count =   3
Finding VCD file...
./dump.vcd
[2024-01-22 13:15:56 UTC] Opening EPWave...
Done
```



Note: To revert to EPWave opening in a new browser window, set that option on your user page.