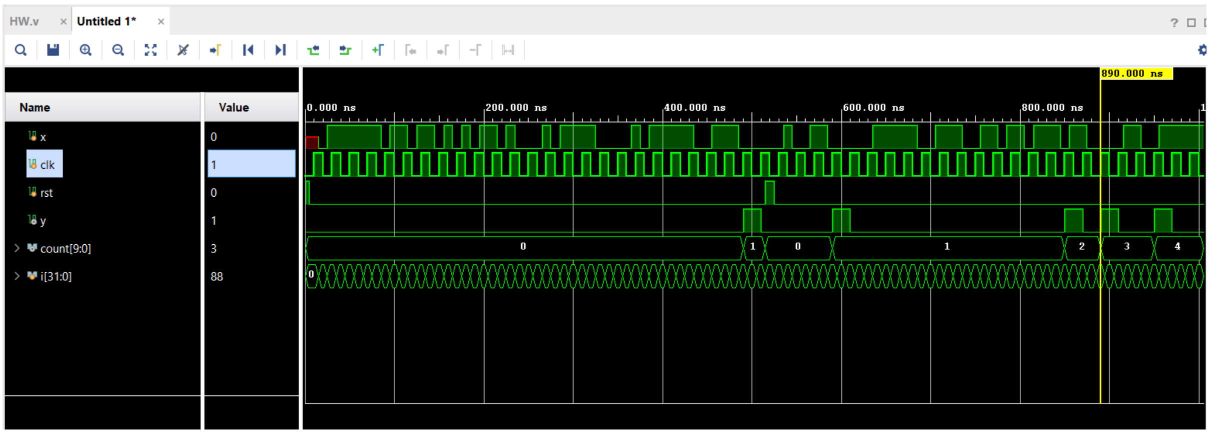
```
`timescale 1ns / 1ps

module Q1_FSM(x,clk,rst,y,count);

   input x,clk,rst;

   output reg y;

   output reg [9:0]count;

   reg [1:0]cs,ns;

   parameter A=2'b00;

   parameter B=2'b01;

   parameter C=2'b10;

   parameter D=2'b11;

   always @(posedge clk, posedge rst)begin

      if(rst)begin

         cs<=A;

         count<=0;

      end

      else

         cs<=ns;

   end

   always @(cs,x)begin

      case (cs)

         A:if(x)

            ns=B;

         else

            ns=A;

         B:if(x)

            ns=D;

         else

            ns=C;

         C:if(x)

            ns=B;

         else
```

```
            ns=A;
        D:if(x)
            ns=D;
        else
            ns=A;
        default:
        ns=A;
      endcase
   end
   always @(cs)begin
      case(cs)
         A:y=0;
         B:y=0;
         C:begin
            y=1;
            count=count+1;
         end
         D:y=0;
         default:begin
            y=0;
            count=0;
         end
      endcase
   end
endmodule

module Q1_FSM_tb();
   reg x,clk,rst;
   wire y;
   wire [9:0]count;
   integer i=0;
```

```
  Q1_FSM d1(x,clk,rst,y,count);
  initial begin
    clk=0;
    forever
      #10 clk=~clk;
  end
  initial begin
    rst=1;
    #5 rst=0;
    for(i=0;i<100;i=i+1)begin
      #10
      x=$random;
      if(i==50)
        rst=1;
      else
        rst=0;
    end
    $stop;
  end
endmodule
```

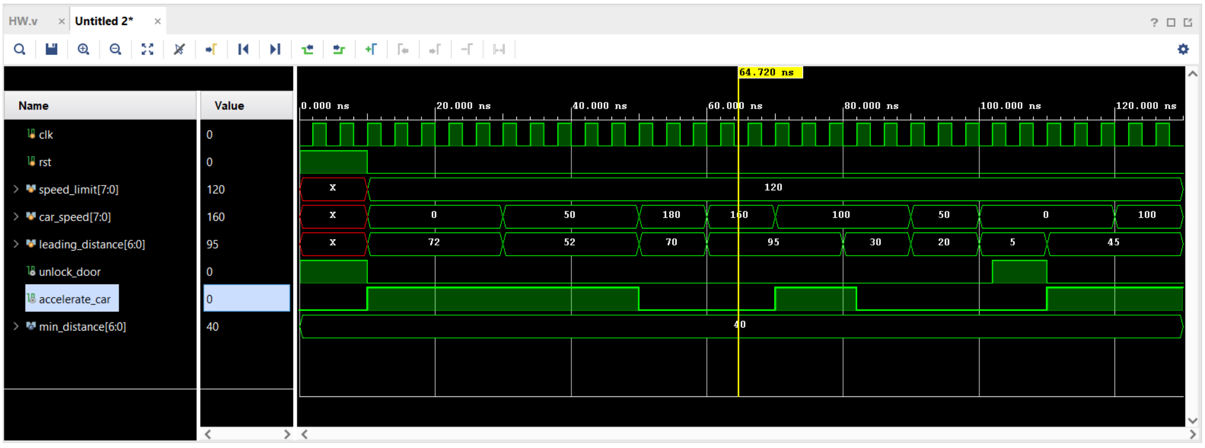


```
/****************************************************************************
****************************************************************************/
module Q2_FSM(speed_limit,car_speed,leading_distance,clk,rst,unlock_door,accelerate_car);
  input clk,rst;
  input [7:0]speed_limit,car_speed;
  input [6:0]leading_distance;
  output reg unlock_door,accelerate_car;
  parameter min_distance=7'd40;
  reg [1:0]cs,ns;
  parameter stop=2'b00;
  parameter accelerate=2'b01;
  parameter decelerate=2'b11;
  always @(posedge clk, posedge rst)begin
     if(rst)begin
        cs<=stop;
     end
     else
        cs<=ns;
  end
  always @(cs,speed_limit,car_speed,leading_distance)begin
     case (cs)
        stop:if(leading_distance>=min_distance)
           ns=accelerate;
        else
           ns=stop;
        accelerate:if( leading_distance<min_distance || car_speed > speed_limit)
           ns=decelerate;
        else
           ns=accelerate;
        decelerate:if(leading_distance>=min_distance && car_speed < speed_limit)
           ns=accelerate;
        else if(car_speed==0)
           ns=stop;
        else
           ns=decelerate;
```

```
          default:
          ns=stop;
       endcase
    end
    always @(cs)begin
       case(cs)
          stop:begin
             unlock_door=1;
             accelerate_car=0;
          end
          accelerate:begin
             unlock_door=0;
             accelerate_car=1;
          end
          decelerate:begin
             unlock_door=0;
             accelerate_car=0;
          end
          default:begin
             unlock_door=1;
             accelerate_car=0;
          end
       endcase
    end
endmodule

module Q2_FSM_tb();
    reg clk,rst;
    reg [7:0]speed_limit,car_speed;
    reg [6:0]leading_distance;
    wire unlock_door,accelerate_car;
```

```
  parameter min_distance=7'd40;


  Q2_FSM #(.min_distance(min_distance))
dut(speed_limit,car_speed,leading_distance,clk,rst,unlock_door,accelerate_car);
  initial begin
    clk=0;
    forever
      #2 clk=~clk;
  end


  initial begin
    rst=1;
    #10
    rst=0;
    //take a road of speed limit 120
    speed_limit=8'd120;
    leading_distance=7'd200;
    // start speed 0
    car_speed=0;
    // then goto accelerate state
    #20 // after 20 time speed increase
    car_speed=8'd50;
    leading_distance=7'd180;
    #20
    car_speed=8'd180;
    leading_distance=7'd70;
    //the speed cross the limit then we go decelerate
    #10
    car_speed=8'd160;
    leading_distance=7'd95;
    #10
```

```
        car_speed=8'd100;

        leading_distance=7'd95;

        #10 // crouded place

        car_speed=8'd100;

        leading_distance=7'd30;

        #10 //slowing down

        car_speed=8'd50;

        leading_distance=7'd20;

        #10 //still croude

        car_speed=8'd0;

        leading_distance=7'd5;

        #10 //  road starts to move again acc state

        car_speed=8'd0;

        leading_distance=7'd45;

        #10

        car_speed=8'd100;

        leading_distance=7'd45;

        #10

        $stop;

    end

endmodule
```



Name	Value
clk	0
rst	0
speed_limit[7:0]	120
car_speed[7:0]	160
leading_distance[6:0]	95
unlock_door	0
accelerate_car	0
min_distance[6:0]	40

/***/

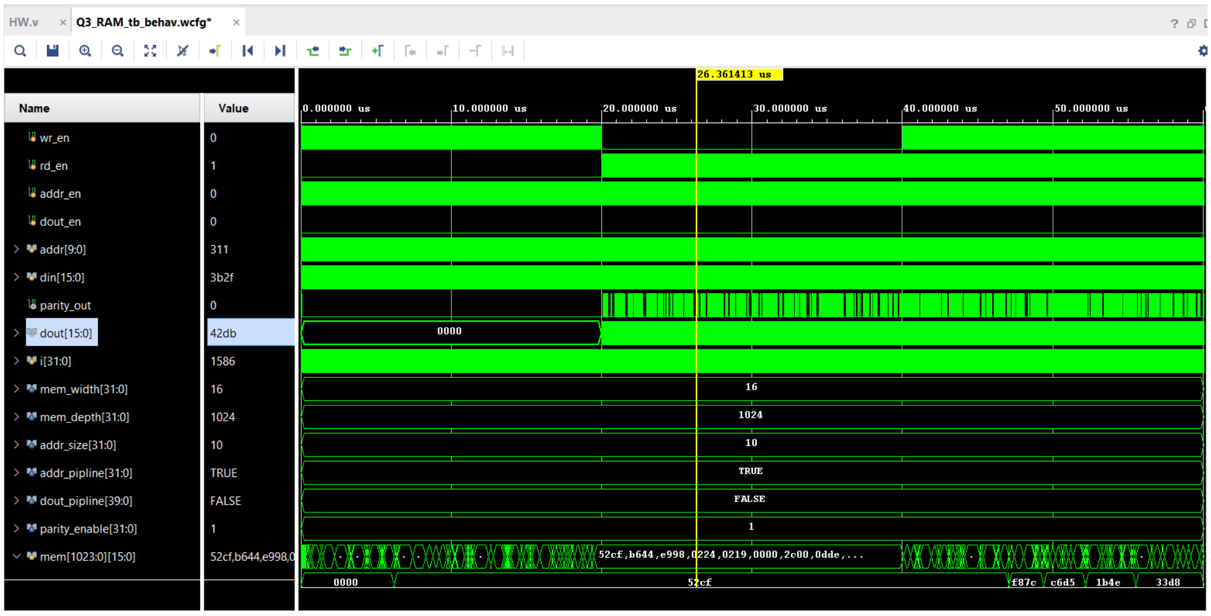
```
module Q3_RAM(clk,rst,blk_select,wr_en,rd_en,addr_en,dout_en,addr,din,parity_out,dout);

   parameter mem_width=16;

   parameter mem_depth=1024;

   parameter addr_size=10;

   parameter addr_pipline="TRUE";

   parameter dout_pipline="FALSE";

   parameter parity_enable=1;


   input clk,rst,blk_select,wr_en,rd_en,addr_en,dout_en;

   input[ addr_size-1:0]addr;

   input [mem_width-1:0]din;

   output reg parity_out;

   output reg [ mem_width-1:0]dout;

   wire [ addr_size-1:0]addr_pip;

   reg [ addr_size-1:0]addr_m;

   wire [mem_width-1:0]dout_M,dout_pip;

   wire par_out;

   mem #(.mem_width(mem_width),.mem_depth(mem_depth),.add_size(addr_size))
m1(clk,rst,blk_select,wr_en,addr_m,din,rd_en,dout_M,par_out);

   D_FlIP_FLOP #(addr_size) ad(addr,addr_en,clk,addr_pip);

   D_FlIP_FLOP #(mem_width) out(dout_M, dout_en ,clk, dout_pip);

   always @(posedge clk ) begin

      if (rst) begin

         // reset


         dout<=0;

         parity_out<=0;

      end

      else begin


         if (addr_pipline=="TRUE")
```

```
            addr_m<=addr_pip;
          else
            addr_m<=addr;
          if (dout_pipline=="FALSE")
            dout<=dout_M;
          else
            dout<=dout_pip;
          if(parity_enable)
            parity_out<=par_out;
          else
            parity_out<=0;
      end
   end




endmodule
module D_FlIP_FLOP(d,e,clk,q);
   parameter N=1;
   input e,clk;
   input [N-1:0]d;
   output reg [N-1:0]q;
   always @(posedge clk ) begin
      if (e)
         q<=d;
   end
endmodule
module mem (clk,rst,blk_slect,wr_en,addr,din,rd_en,dout,par_out);
   parameter mem_width=16;
   parameter mem_depth=1024;
   parameter add_size=10;
```

```
    input clk,rst,blk_slect,wr_en,rd_en;
    input [add_size-1:0]addr;
    input [mem_width-1:0]din;
    output reg [mem_width-1:0]dout;
    output reg par_out;
    reg [mem_width-1:0]mem[mem_depth-1:0];

    always @(posedge clk ) begin
        if (rst) begin
            // reset
            dout<=0;
            par_out<=0;
        end
        else if (blk_slect) begin
            if(wr_en)
                mem[addr]<=din;
            else if(rd_en)begin
                dout<=mem[addr];
                par_out<=^mem[addr];
            end
        end
    end
endmodule

module Q3_RAM_tb();

reg clk,rst,blk_select,wr_en,rd_en,addr_en,dout_en;

    parameter mem_width=16;
    parameter mem_depth=1024;
    parameter addr_size=10;
    parameter addr_pipline="TRUE";

    parameter dout_pipline="FALSE";

    parameter parity_enable=1;


    reg[ addr_size-1:0]addr;

    reg [mem_width-1:0]din;

    wire parity_out;

    wire [ mem_width-1:0]dout;


Q3_RAM
#(.mem_width(mem_width),.mem_depth(mem_depth),.addr_size(addr_size),.addr_pipline(addr_pi
pline),.dout_pipline(dout_pipline),.parity_enable(parity_enable))

 dut(clk,rst,blk_select,wr_en,rd_en,addr_en,dout_en,addr,din,parity_out,dout);

initial begin

clk=0;

forever

#2 clk=~clk;

end

integer i=0;

initial begin

$readmemh("mem.dat",dut.m1.mem);

rst =1;

blk_select=0;

rd_en=0;

addr_en=0;

wr_en=0;

dout_en=0;

addr=0;

din=0;

#10

rst=0;

rd_en=0;
```

```
for(i=0;i<5000;i=i+1)begin
@ (negedge clk);
addr_en=$random;
blk_select=$random;
addr=$random;
din=$random;
wr_en=$random;
end
#10
wr_en=0;
for(i=0;i<5000;i=i+1)begin
@ (negedge clk);
addr_en=$random;
blk_select=$random;
addr=$random;
din=$random;
rd_en=$random;
end
#10;
for(i=0;i<5000;i=i+1)begin
@ (negedge clk);
addr_en=$random;
blk_select=$random;
addr=$random;
din=$random;
rd_en=$random;
wr_en=~rd_en;
end
#10
$stop;
end
```

```
endmodule
```



Name	Value
wr_en	0
rd_en	1
addr_en	0
dout_en	0
addr[9:0]	311
din[15:0]	3b2f
parity_out	0
dout[15:0]	42db
i[31:0]	1586
mem_width[31:0]	16
mem_depth[31:0]	1024
addr_size[31:0]	10
addr_pipline[31:0]	TRUE
dout_pipline[39:0]	FALSE
parity_enable[31:0]	1
mem[1023:0][15:0]	52cf,b644,e998,0