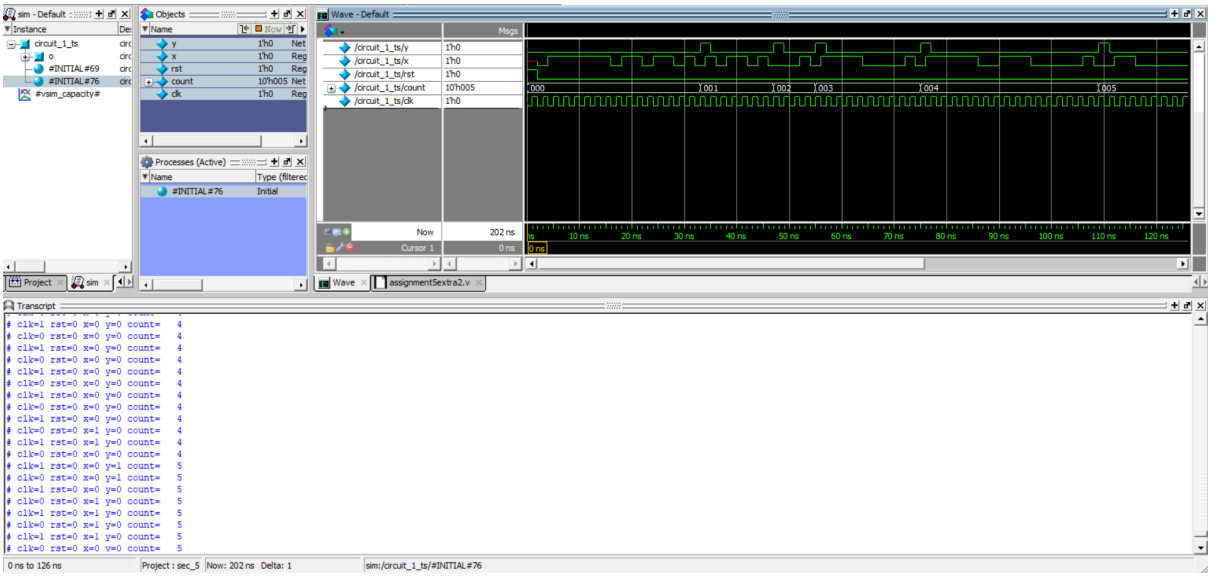
q2

```
module circuit_1 ( output reg y,
                   output reg [9:0] count,
                   input x,clk,rst);
parameter IDLE = 2'b00;
parameter ZERO = 2'b01;
parameter STORE = 2'b10;
parameter ONE = 2'b11;

reg [1:0] cs,ns;

always @(posedge clk , posedge rst) begin
    if (rst) begin
        cs <= IDLE;
        count <= 0;
    end
    else begin
        cs <= ns;

    end
end
always @(*) begin
    case (cs)
        IDLE   : begin
                    if (x)
                        ns = IDLE;
                    else
                        ns = ZERO;
                end
        ZERO   : begin
                    if (x)
                        ns = ONE;
                    else
                        ns = ZERO;
                end
        ONE    : begin
                    if (x)
                        ns = IDLE;
                    else
                        ns = STORE;
                end
        STORE  : begin
                    if (x)
                        ns = IDLE;
                    else
                        ns = ZERO;
                end
    endcase
end
```

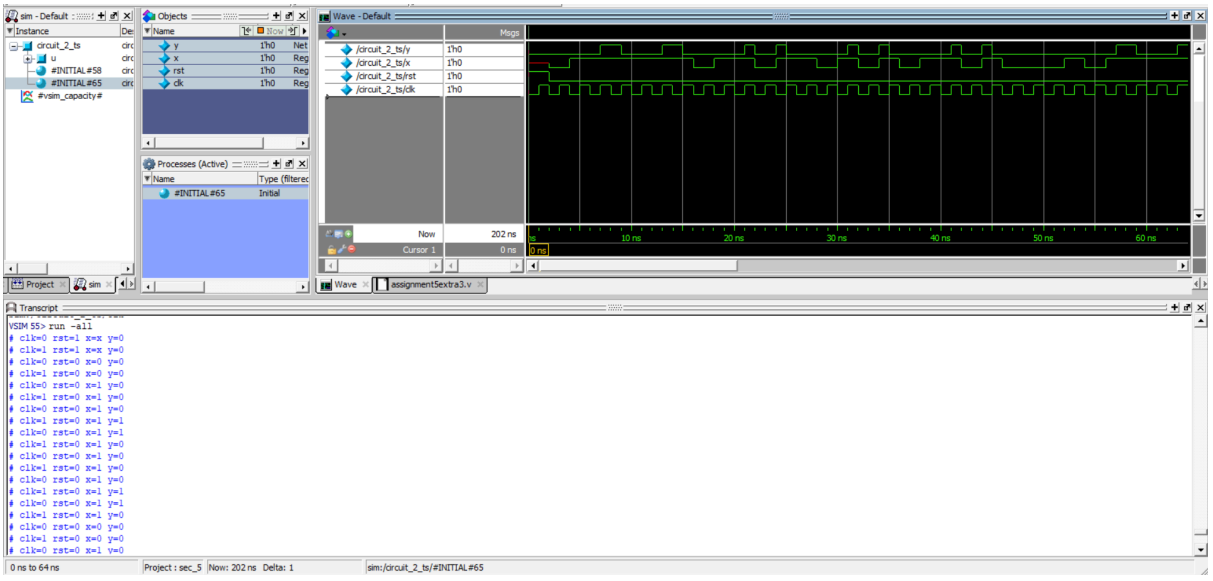
```
always @(*) begin
    case (cs)
        IDLE      : y = 0;
        ZERO      : y = 0;
        ONE       : y = 0;
        STORE     : begin
                        y = 1;
                        count = count + 1;
                    end
    endcase
end
endmodule

module circuit_1_ts ();
wire y;
wire [9:0] count;
reg x,clk,rst;

circuit_1 o (y,count,x,clk,rst);

initial begin
    clk = 0;
    forever begin
        #1 clk = ~clk;
    end
end

initial begin
    rst = 1;
    @(negedge clk)
    rst = 0;
    repeat (100) begin
    x = $random;
    @(negedge clk);
    end
    $stop;
end
initial begin
    $monitor ("clk=%b rst=%b x=%b y=%b count=%d",clk,rst,x,y,count);

end
endmodule
```


sim - Default
Objects
Wave - Default
Instance
circuit_1_ts
o
#INITIAL#69
#INITIAL#76
#vsim_capacity#
Name
y
x
rst
count
clk
Processes (Active)
#INITIAL#76
Initial
Msgs
/circuit_1_ts/y
/circuit_1_ts/x
/circuit_1_ts/rst
/circuit_1_ts/count
/circuit_1_ts/clk
Now
202 ns
Cursor 1
0 ns
10 ns
20 ns
30 ns
40 ns
50 ns
60 ns
70 ns
80 ns
90 ns
100 ns
110 ns
120 ns
Project
sim
Wave
assignment5extra2.v
Transcript
clk=1 rst=0 x=0 y=0 count= 4
clk=0 rst=0 x=0 y=0 count= 4
clk=1 rst=0 x=0 y=0 count= 4
clk=0 rst=0 x=0 y=0 count= 4
clk=1 rst=0 x=0 y=0 count= 4
clk=0 rst=0 x=0 y=0 count= 4
clk=1 rst=0 x=0 y=0 count= 4
clk=0 rst=0 x=0 y=0 count= 4
clk=1 rst=0 x=0 y=0 count= 4
clk=0 rst=0 x=1 y=0 count= 4
clk=1 rst=0 x=1 y=0 count= 4
clk=0 rst=0 x=0 y=1 count= 5
clk=1 rst=0 x=0 y=1 count= 5
clk=0 rst=0 x=0 y=0 count= 5
clk=1 rst=0 x=1 y=0 count= 5
clk=0 rst=0 x=1 y=0 count= 5
clk=1 rst=0 x=1 y=0 count= 5
clk=0 rst=0 x=0 y=0 count= 5
0 ns to 126 ns
Project : sec_5
Now: 202 ns
Delta: 1
sim:/circuit_1_ts/#INITIAL#76

q3

```
1   module circuit_2 ( output y,
2                      input x,clk,rst);
3   parameter S0 = 2'b00;
4   parameter S1 = 2'b01;
5   parameter S2 = 2'b10;
6   parameter S3 = 2'b11;
7
8   reg [1:0] cs,ns;
9
10  always @(posedge clk , posedge rst) begin
11      if (rst) begin
12          cs <= S0;
13      end
14      else begin
15          cs <= ns;
16
17      end
18  end
19  always @(*) begin
20      case (cs)
21          S0   : begin
22                      if (x)
23                          ns = S1;
24                      else
25                          ns = S0;
26                  end
27          S1   : begin
28                      if (x)
29                          ns = S2;
30                      else
31                          ns = S1;
32                  end
33          S2    : begin
34                      if (x)
35                          ns = S3;
36                      else
37                          ns = S2;
38                  end
39          S3   : begin
40                      if (x)
41                          ns = S1;
42                      else
43                          ns = S0;
44                  end
45      endcase
46  end
```

```
48  assign y = ((cs == S2) && (x==1)) ? 1'b1 : 1'b0;
49
50  endmodule
51
52  module circuit_2_ts ();
53  wire y;
54  reg x,clk,rst;
55
56  circuit_2 u (y,x,clk,rst);
57
58  initial begin
59      clk = 0;
60      forever begin
61          #1 clk = ~clk;
62      end
63  end
64
65  initial begin
66      rst = 1;
67      @(negedge clk)
68      rst = 0;
69      repeat (100) begin
70      x = $random;
71      @(negedge clk);
72      end
73      $stop;
74  end
75  initial begin
76      $monitor ("clk=%b rst=%b x=%b y=%b",clk,rst,x,y);
77
78  end
79  endmodule
```



q4

```
module rom_4x4 ( output reg [3:0] data_out,
                 input [1:0] address,
                 input clk,rst);

        reg [3:0] mem [3:0] ;

        always @(posedge clk) begin
            if (rst)
                data_out <= 0;
            else
                data_out <= mem[address];
        end

endmodule

module rom_4x4_ts ();

wire [3:0] data_out;
reg [1:0] address;
reg clk , rst;

rom_4x4 p (data_out,address,clk,rst);

initial begin
    clk = 0;
    forever #1 clk = ~clk;
end
initial begin
    $readmemb ("mem_extra_q4.txt", p.mem);
    rst = 1;
    repeat (5) @(negedge clk);
    rst = 0;
    repeat (100) begin
        address = $random;
        @(negedge clk);
    end
    $stop;
end

initial begin
    $monitor ("clk=%b rst=%b address=%b out=%b",clk,rst,address,data_out);
end
endmodule
```