CE/CZ1055 Digital Logic
Tutorial 10: Finite State Machines

1. Design a finite state machine that has a single input and single output. It outputs a 1 from the second consecutive high input, and only then outputs a zero after the second consecutive low input. Hence, two consecutive 1 inputs, get a high output, that stays until two consecutive low inputs are received.

   (a) Implement the finite state machine in Verilog using only assign statements for the state transition logic.

   (b) Redo the implementation using a combinational always block for the state transition logic.

   (c) Show how the state machine would respond to the following sequence of inputs:
   0-------1----------0----------1
   0,1,0,1,1,0,1,1,0,0,0,1,0,1,1

a)
```
module consecutive (input b,
                    input clk, rst,
                    output p);
  wire n2, n1, n0
  reg s2, s1, s0

  assign n0 = ~b;
  assign n1 = (~s0&~s1&b)|(s0&~s1&~b)
  assign n2 = 1'b0;

  always @ (posedge clk)
  begin
     if (rst) begin
        s0 <= 1'b0;
        s1 <= 1'b0;
     end else begin
        s0 <= n0;
        s1 <= n1;

  end

  always @ *
  begin
     if (s0 = 1) begin
        if (s1 = 0 & s2 = 0) begin
           p = 1;
        end if (s1 = 1 & s2 = 0) begin
           p = 0;
       end
  end
```

b)
```
module consecutive (input b,
                    input clk, rst,
                    output p);
  reg [1:0] nst, st
  parameter
  s1 = 2'b000;
  s2 = 2'b001;
  s3 = 2'b010;
  s4 = 2'b011;

  always @ (posedge clk)
  begin
     if (rst) begin
        st <= 2'b100
     end else begin
        st <= nst
  end

  always @ *
  nst = st;
  begin
     case (st)
        start: if (b) nst = s1;
               else nst = s3;
        s1: if (b) nst = s2;
               else nst = s3;
        s2: if (b) nst = s1;
               else nst = s3;
        s3: if (b) nst = s1;
               else nst = s4;
        s4: if (b) nst = s1;
               else nst = s3;
        default: nst = start;
  end
```