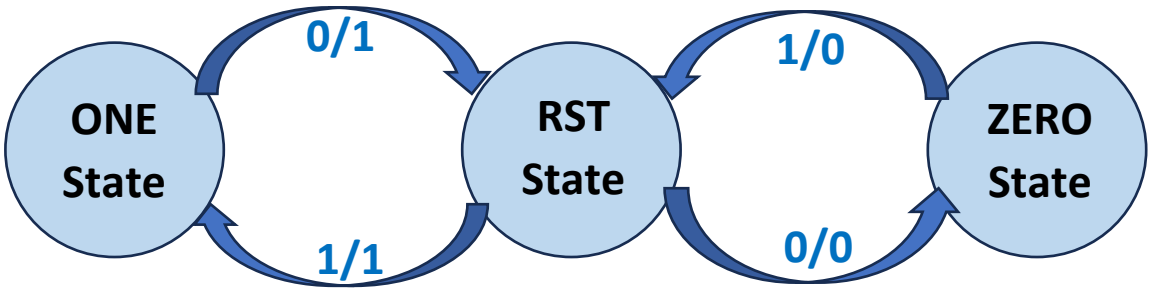
Problem 2: (25 points)

Design a glitch-free Manchester-to-NRZ Mealy FSM data converter. Your submission should include:

- A state graph for your FSM. **(5 points)**
- A screen shot of your Verilog code for the data converter. **(10 points)**
- A screen shot of your testbench. Make sure that you do not align your data_in transitions with the clock to show how you were able to implement a glitch-free design. **(4 points)**
- A screenshot of your simulation results. **(6 points)**




```
/*-------------------------------------------------------------
  Manchester_2_NRZ Serial Data Line converter:
  --------------------------------------------
  This design will have two examples, one with registered output
  to avoid glitches and one with an output directly connected
  to combinational logic. The data_in is not aligned with the
  positive clock edge. The registered output design is glitch free
  while the simple Mealy FSM will have glitches.
  The two designs will be combined in one top module and tested
  using the same testbench to compoare the simulation output.
  The modules included:
  - Manchester_NRZ_glitchy
  - Manchester_NRZ_nonglitchy
  - Manchester_2_NRZ            (top module)
  - Manchester_2_NRZ_tb         (testbench)
-------------------------------------------------------------*/

module Manchester_2_NRZ (input  reset_in,
                         input  clk_in,
                         input  Manchester_in,
                         output NRZ_glitchy,
                         output NRZ_nonglitchy);

Manchester_NRZ_glitchy     M1  (.reset      (reset_in),
                                .clk        (clk_in),
                                .Manchester (Manchester_in),
                                .NRZ_out    (NRZ_glitchy)    );

Manchester_NRZ_nonglitchy  M2  (.reset      (reset_in),
                                .clk        (clk_in),
                                .Manchester (Manchester_in),
                                .NRZ_out    (NRZ_nonglitchy)    );
endmodule
```

```
module Manchester_NRZ_glitchy (reset, clk, Manchester, NRZ_out);

 input        reset, clk, Manchester;
 output reg   NRZ_out;                  // to assign it values within the always block
 reg [1:0]    state, next_state;

 parameter RSt  = 2'b11;
 parameter ONE  = 2'b01;
 parameter ZERO = 2'b00;

 // Sequential logic to update the state:
 always @ (posedge clk)
    if (reset) state <= RSt;
    else       state <= next_state;

 // Combinational logic to update the next_state and NRZ_out:
 always @ *
   case (state)
        RSt    :    if  (Manchester) begin
                          next_state = ONE;    NRZ_out = 1'b1;  end
                               else  begin
                          next_state = ZERO;   NRZ_out = 1'b0;  end
        ONE    :    begin next_state = RSt;    NRZ_out = 1'b1;  end
        ZERO   :    begin next_state = RSt;    NRZ_out = 1'b0;  end
       default :    begin next_state = RSt;    NRZ_out = 1'b0;  end
   endcase

 endmodule


module Manchester_NRZ_nonglitchy (reset, clk, Manchester, NRZ_out);

 input        reset, clk, Manchester;
 output reg   NRZ_out;
 reg          next_NRZ;
 reg [1:0]    state, next_state;

 parameter RSt  = 2'b11;
 parameter ONE  = 2'b01;
 parameter ZERO = 2'b00;

 // Sequential logic to update the state and the NRZ output:
 always @ (posedge clk)
    if (reset) begin state <= RSt;              NRZ_out <= 1'b0;       end
    else       begin state <= next_state;       NRZ_out <= next_NRZ;   end

 // Combinational logic to update the next_state and next_NRZ:
 always @ *
   case (state)
        RSt    :    if  (Manchester) begin
                          next_state = ONE;    next_NRZ = 1'b1;  end
                               else  begin
                          next_state = ZERO;   next_NRZ = 1'b0;  end
        ONE    :    begin next_state = RSt;    next_NRZ = 1'b1;  end
        ZERO   :    begin next_state = RSt;    next_NRZ = 1'b0;  end
       default :    begin next_state = RSt;    next_NRZ = 1'b0;  end
   endcase

 endmodule
```

```
module Manchester_2_NRZ_tb ();

reg  reset_in;
reg  clk_in;
reg  Manchester_in;
wire NRZ_glitchy;
wire NRZ_nonglitchy;

Manchester_2_NRZ UUT (reset_in, clk_in, Manchester_in, NRZ_glitchy, NRZ_nonglitchy);

initial
    begin         clk_in = 1'b0;
    forever  begin
             #10  clk_in = ~clk_in;    end    end

initial
    begin          reset_in = 1'b1;
             #95   reset_in = 1'b0;    end

initial
    begin       Manchester_in = 1'b0;
          #5    Manchester_in = 1'b1;      // 10 represents 1
          #20   Manchester_in = 1'b0;
    forever
          begin
          #20   Manchester_in = 1'b1;      // 10 represents 1
          #20   Manchester_in = 1'b0;

          #20   Manchester_in = 1'b0;      // 01 represents 0
          #20   Manchester_in = 1'b1;

          #20   Manchester_in = 1'b0;      // 01 represents 0
          #20   Manchester_in = 1'b1;

          #20   Manchester_in = 1'b1;      // 10 represents 1
          #20   Manchester_in = 1'b0;

          #20   Manchester_in = 1'b0;      // 01 represents 0
          #20   Manchester_in = 1'b1;  end
   end

endmodule
```