## Ideas for implementing the Project on FPGA:

1. **Modelsim simulation:**

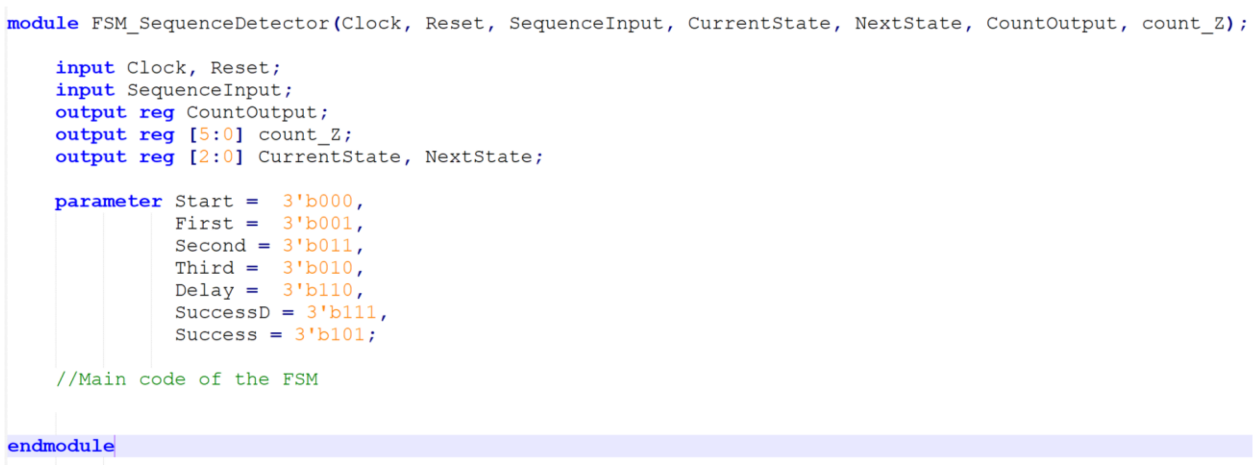
Input to the FSM will be 1-bit Clock, 1-bit Active-low synchronous Reset, 1-bit Input data (input data sequence will come through this variable).

The output of the FSM will be 3-bit CurrentState, 3-bit NextState, where the three bits are gray coded binary numbers found from part 1 of the project.

There are two more outputs:

Each time a correct sequence is found, the 1-bit output of the FSM will be set to 1. And each time the 1-bit output of the FSM is 1, the counter variable will increase by 1.

Verilog code of the proposed I/O scheme for the FSM in Modelsim:

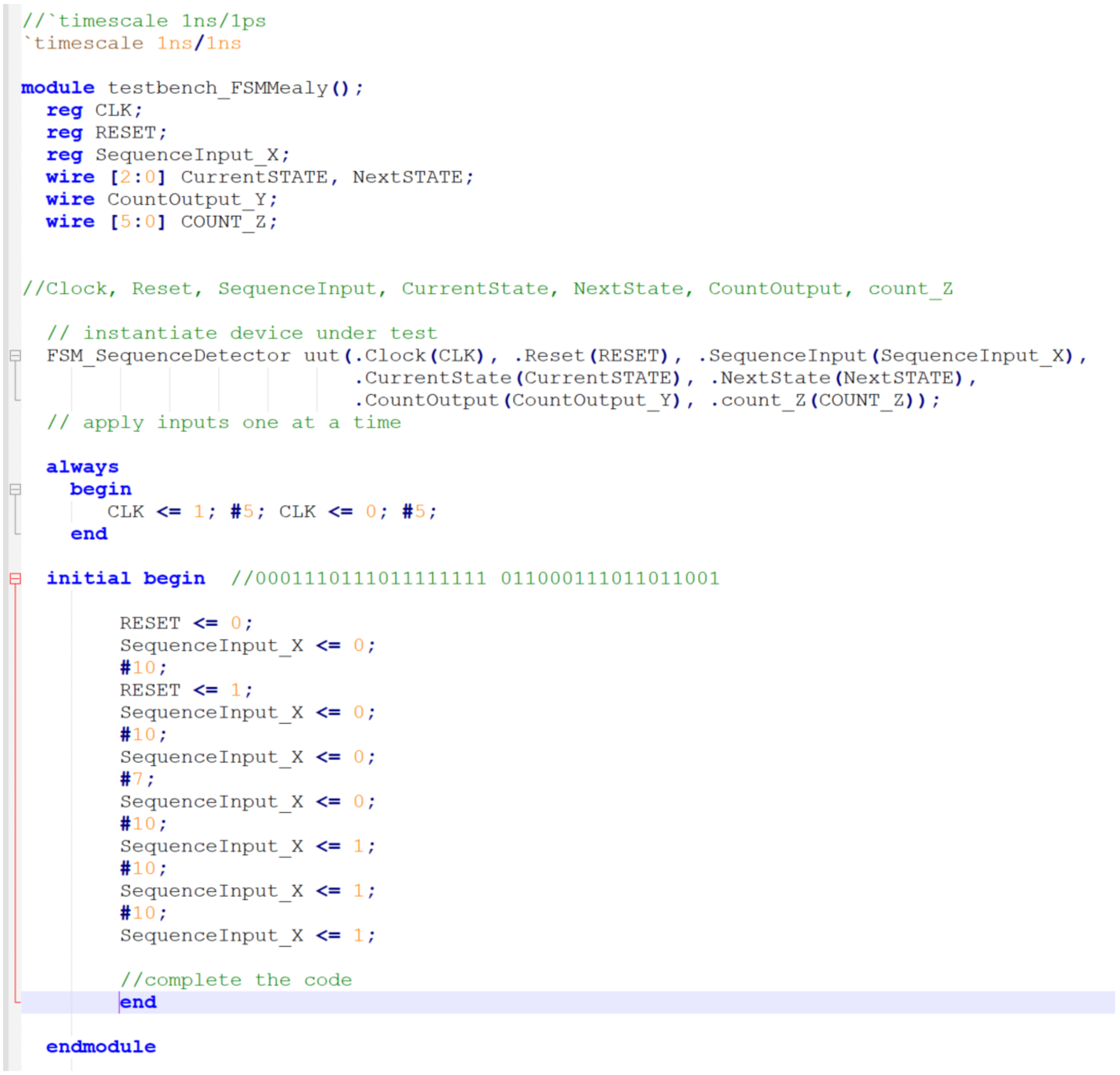


```
module FSM_SequenceDetector(Clock, Reset, SequenceInput, CurrentState, NextState, CountOutput, count_Z);

    input Clock, Reset;
    input SequenceInput;
    output reg CountOutput;
    output reg [5:0] count_Z;
    output reg [2:0] CurrentState, NextState;

    parameter Start =  3'b000,
              First =  3'b001,
              Second = 3'b011,
              Third =  3'b010,
              Delay =  3'b110,
              SuccessD = 3'b111,
              Success = 3'b101;

    //Main code of the FSM

endmodule
```

Assume you are giving this input sequence to the FSM:

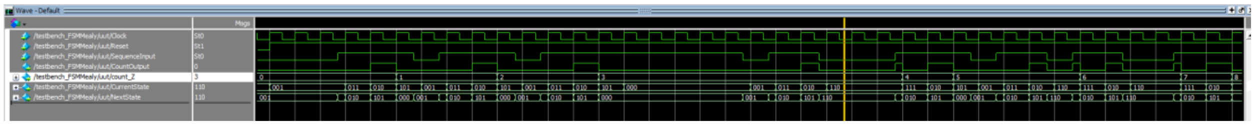
00011101110111111111 011000111011011001

In the testbench, if you want to give 3 consecutive 0s as input (000), you can give one ‘0’ as input, and then apply three clock pulses. In the same way, if you want to give five consecutive 1s as input (11111), you can give one ‘1’ as input, and then apply five clock pulses. In the screenshot of the testbench below, I give each input explicitly to make the code more readable:



```
//`timescale 1ns/1ps
`timescale 1ns/1ns

module testbench_FSMMealy();
  reg CLK;
  reg RESET;
  reg SequenceInput_X;
  wire [2:0] CurrentSTATE, NextSTATE;
  wire CountOutput_Y;
  wire [5:0] COUNT_Z;


//Clock, Reset, SequenceInput, CurrentState, NextState, CountOutput, count_Z

  // instantiate device under test
  FSM_SequenceDetector uut(.Clock(CLK), .Reset(RESET), .SequenceInput(SequenceInput_X),
                          .CurrentState(CurrentSTATE), .NextState(NextSTATE),
                          .CountOutput(CountOutput_Y), .count_Z(COUNT_Z));
  // apply inputs one at a time

  always
    begin
      CLK <= 1; #5; CLK <= 0; #5;
    end

  initial begin  //0001110111011111111 011000111011011001

        RESET <= 0;
        SequenceInput_X <= 0;
        #10;
        RESET <= 1;
        SequenceInput_X <= 0;
        #10;
        SequenceInput_X <= 0;
        #7;
        SequenceInput_X <= 0;
        #10;
        SequenceInput_X <= 1;
        #10;
        SequenceInput_X <= 1;
        #10;
        SequenceInput_X <= 1;

        //complete the code
        end

  endmodule
```

The wave simulation result of the proposed FSM will be as below:


Wave - Default
Msgs
/testbench_FSMMealy/uut/Clock
/testbench_FSMMealy/uut/Reset
/testbench_FSMMealy/uut/SequenceInput
/testbench_FSMMealy/uut/CountOutput
/testbench_FSMMealy/uut/count_Z
/testbench_FSMMealy/uut/CurrentState
/testbench_FSMMealy/uut/NextState

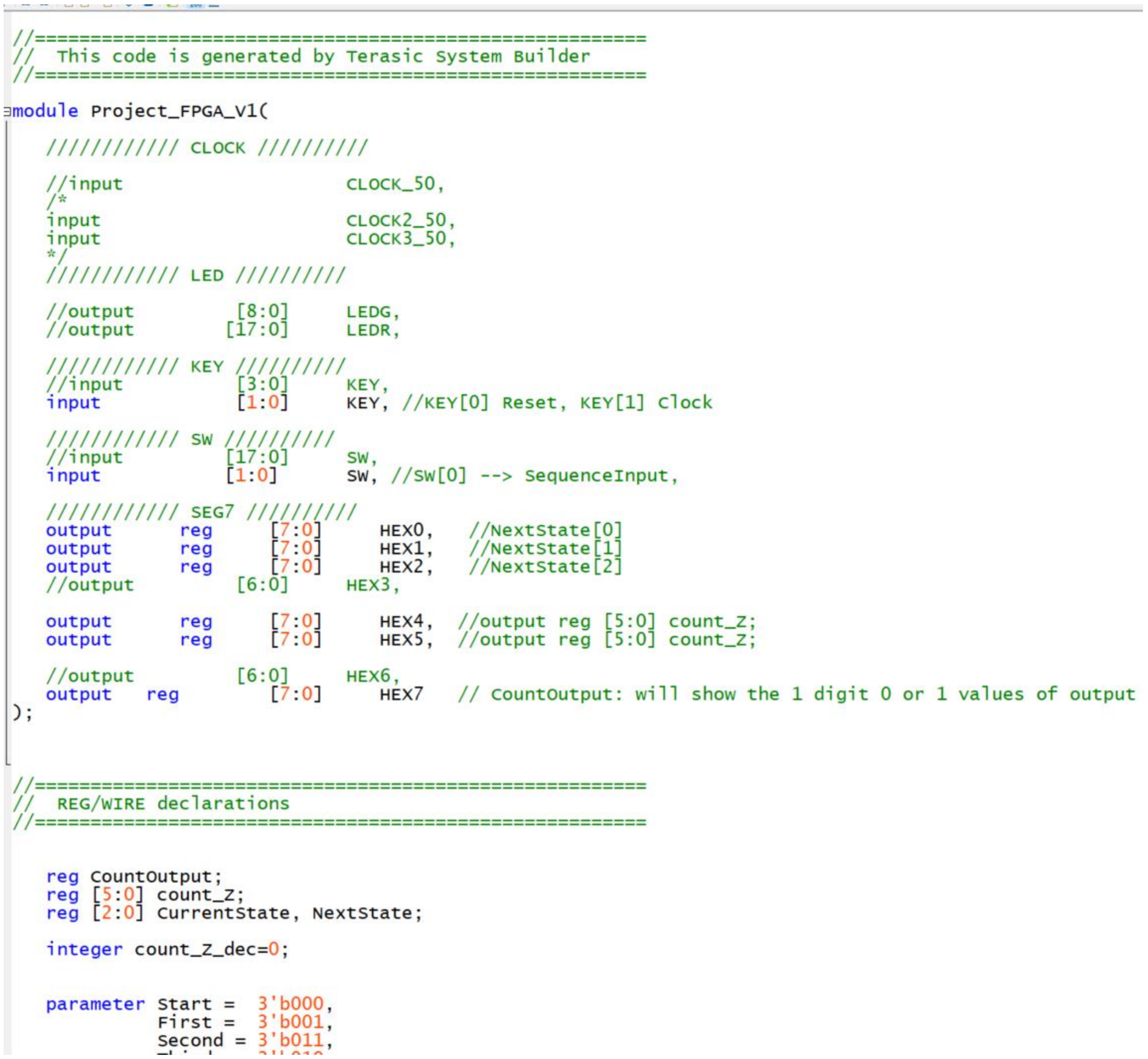
## 2. Implementing the FSM on FPGA:

Follow the procedure in lab 4 to create the Verilog code using Systembuilder and Altera Quartus.

On the FPGA, you can use KEY[0] and KEY[1] as Reset and Clock, respectively. KEY in the FPGA is a push-button (PB) switch, so by pressing the KEY[1] PB, you are manually giving clock pulses to the FPGA. Use SW[0] slide switch (SS) to give the input sequence. So as described before, if you want to give five consecutive 1s as input (11111) to the FPGA, you can give one '1' as input by setting SW[0] to ON position, and then apply five clock pulses using KEY[1] PB.

For output, you can show 3 gray coded bits of NextState to HEX0, HEX1 and HEX2 displays of the FPGA. You can show the 1-bit output of the FSM using HEX7, and the 2-digit decimal counter variable (count_z_dec) can be shown using HEX4 and HEX5.

Verilog code of the proposed I/O scheme for the FSM in Quartus:



```
//=======================================================
//  This code is generated by Terasic System Builder
//=======================================================

module Project_FPGA_V1(

    /////////// CLOCK //////////

    //input                     CLOCK_50,
    /*
    input                       CLOCK2_50,
    input                       CLOCK3_50,
    */
    /////////// LED //////////

    //output          [8:0]     LEDG,
    //output         [17:0]     LEDR,

    /////////// KEY //////////
    //input           [3:0]     KEY,
    input             [1:0]     KEY, //KEY[0] Reset, KEY[1] Clock

    /////////// SW //////////
    //input          [17:0]     SW,
    input            [1:0]      SW, //SW[0] --> SequenceInput,

    /////////// SEG7 //////////
    output      reg     [7:0]     HEX0,    //NextState[0]
    output      reg     [7:0]     HEX1,    //NextState[1]
    output      reg     [7:0]     HEX2,    //NextState[2]
    //output          [6:0]     HEX3,

    output      reg     [7:0]     HEX4,   //output reg [5:0] count_Z;
    output      reg     [7:0]     HEX5,   //output reg [5:0] count_Z;

    //output          [6:0]     HEX6,
    output   reg        [7:0]     HEX7   // CountOutput: will show the 1 digit 0 or 1 values of output
);


//=======================================================
//  REG/WIRE declarations
//=======================================================


    reg CountOutput;
    reg [5:0] count_Z;
    reg [2:0] CurrentState, NextState;

    integer count_Z_dec=0;


    parameter Start =  3'b000,
              First =  3'b001,
              Second = 3'b011,
          Third =  3'b010,
          Delay =  3'b110,
          SuccessD = 3'b111,
          Success = 3'b101;


//write your code here.

endmodule
```