Lab 10

A Digital Lock

Spring 2025

Submitted by: **Mohsin Sajjad**

Registration No: **22pwsce2149**

Class Section: **A**

"On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work."

Student Signature: _____

Submitted to:

Engr. Faheem Jan

Month Day, Year (18 05, 2025)

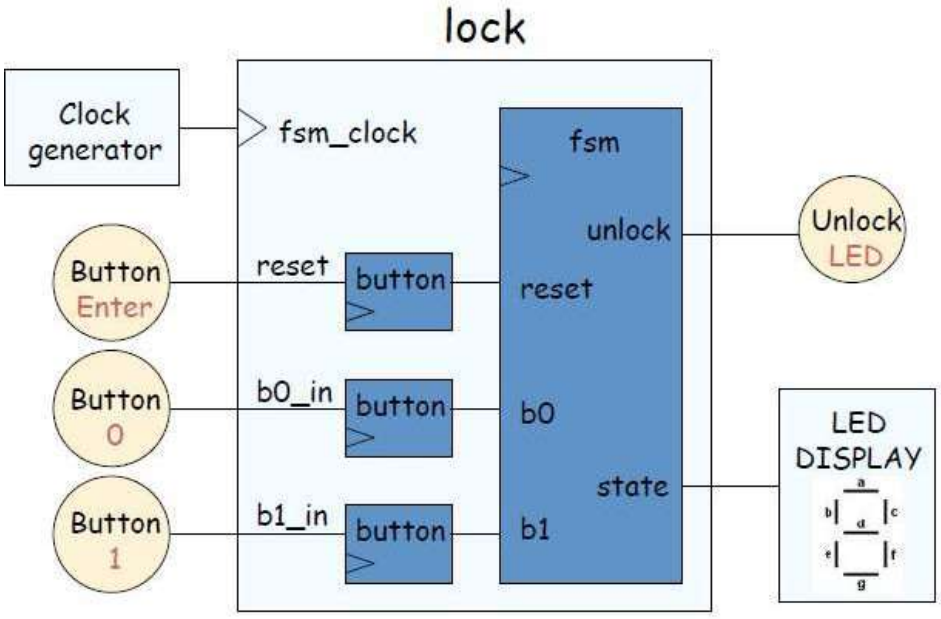
Department of Computer Systems Engineering

University of Engineering and Technology, Peshawar

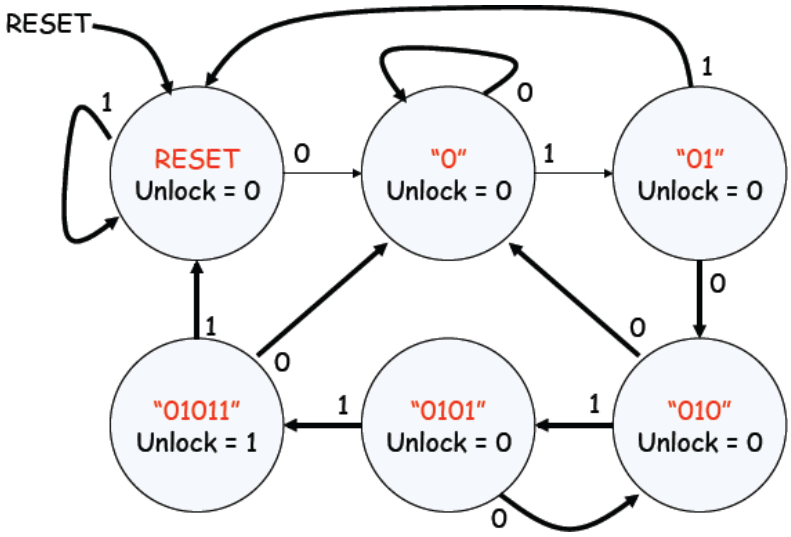
A Digital Lock

Objective: Build an electronic combination lock with reset button, two number buttons (0 and 1), and an unlock output. The combination should be “01011”

Block Diagram: A combinational digital lock has three input buttons for Reset, entering a “0” and entering a “1” and output button UNLOCK and state which shows in which state the machine is currently in. The state output is connected to the seven segment display on the S6BOARD. The Module button in the below diagram is an abstraction of the synchronizer and level to pulse converter from the previous lab. The state transition diagram is given in the following figures.



State Transition Diagram:



Lab Tasks:

1- Build an electronic combination lock with reset button, two number buttons (0 and 1), and an unlock output. The combination should be "01011"

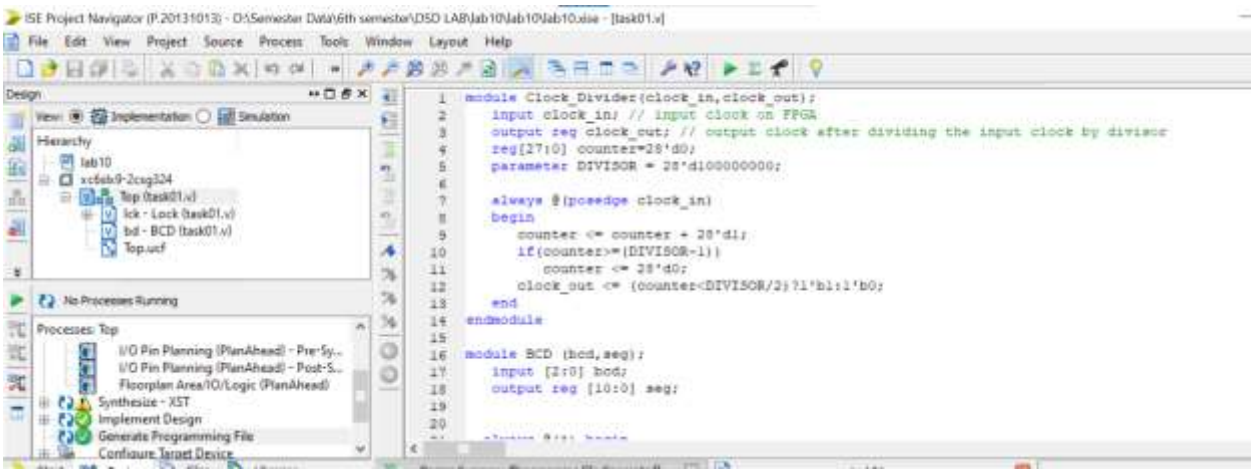
CODE:

```
module Lock(button0, button1, clk_in, rst, openLock, PS);

   input button0, button1, clk_in, rst;

   output openLock;

   output reg [2:0]PS;

   reg openLock;


   parameter      S0 = 3'b000,

                  S1 = 3'b001,
```

```
            S2 = 3'b010,
            S3 = 3'b011,
            S4 = 3'b100,
            S5 = 3'b101;

reg [2:0] state;

wire slowclock;
Clock_Divider cd(clk_in, slowclock);

always @(posedge slowclock) begin
   if (rst) begin
      openLock = 0;
      state = S0;
   end else begin
      case (state)
         S0: begin
            if (!button0)      // input 0
               state = S1;
            else if (!button1)
               state = S0;
         end

         S1: begin
```

```
   if (!button1)      // input 1 => 01
      state = S2;
   else if (!button0)
      state = S1;
end

S2: begin
   if (!button0)      // input 0 => 010
      state = S3;
   else if (!button1)
      state = S0;
end

S3: begin
   if (!button1)      // input 1 => 0101
      state = S4;
   else if (!button0)
      state = S1;
end

S4: begin
   if (!button1)      // input 1 => 01011
      state = S5;
   else if (!button0)
```

```
                state = S1;
            end

            S5: begin
               // Once unlocked, reset or wait for reset
               if (!button0 || !button1)
                  state = S0;
            end
         endcase
      end

      PS <= state;

      if (state == S5)
         openLock = 1;
      else
         openLock = 0;
   end
endmodule
```



output:

Conclusion:

This Verilog code implements a digital lock system that unlocks when the button sequence `01011` is entered. It uses a finite state machine (FSM) to track the input pattern, a clock divider to slow down the clock, and a 7-segment display module to show the current FSM state. When the correct sequence is entered, the `Unlock` output goes high.

TASk 02:

Change the functionality of the lock such that it unlocks on the sequence of 11011.

CODE:

```
module Clock_Divider(clock_in,clock_out);

	input clock_in; // input clock on FPGA

	output reg clock_out; // output clock after dividing the input clock by divisor

	reg[27:0] counter=28'd0;

	parameter DIVISOR = 28'd100000000;


	always @(posedge clock_in)

	begin

		counter <= counter + 28'd1;

		if(counter>=(DIVISOR-1))

			counter <= 28'd0;

		clock_out <= (counter<DIVISOR/2)?1'b1:1'b0;

	end

endmodule


module BCD (bcd,seg);
```

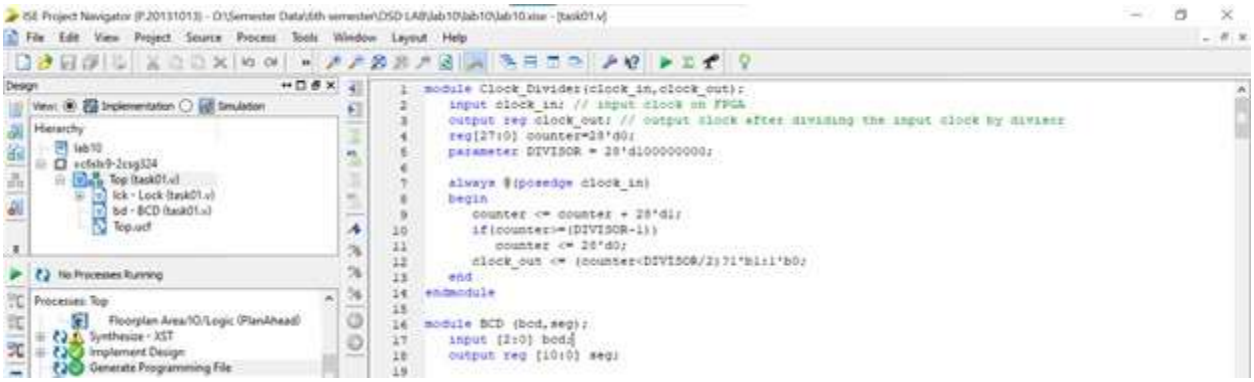
```
        input [2:0] bcd;
        output reg [10:0] seg;



  always @(*) begin
    case(bcd)
      3'b000: seg = 11'b10000000011; // 0
      3'b001: seg = 11'b11110010011; // 1
      3'b010: seg = 11'b01001000011; // 2
      3'b011: seg = 11'b01100000011; // 3
      3'b100: seg = 11'b00110010011; // 4
      3'b101: seg = 11'b00100100011; // 5

      default: seg = 11'b00000000011; // Error or Blank A
    endcase
  end

endmodule
module Lock(button0, button1, clk_in, rst, openLock, PS);
    input button0, button1, clk_in, rst;
    output openLock;
    output reg [2:0]PS;
    reg openLock;
```

```
parameter               S0 = 3'b000,
                        S1 = 3'b001,
                        S2 = 3'b010,
                        S3 = 3'b011,
                        S4 = 3'b100,
                        S5 = 3'b101;

reg [2:0] state;

wire slowclock;
                        Clock_Divider (clk_in,slowclock);

always @(posedge slowclock) begin
   if (rst) begin
      openLock = 0;
      state = S0;
   end

   else begin
      case (state)
S0: begin
   if (!button1) // first 1
```

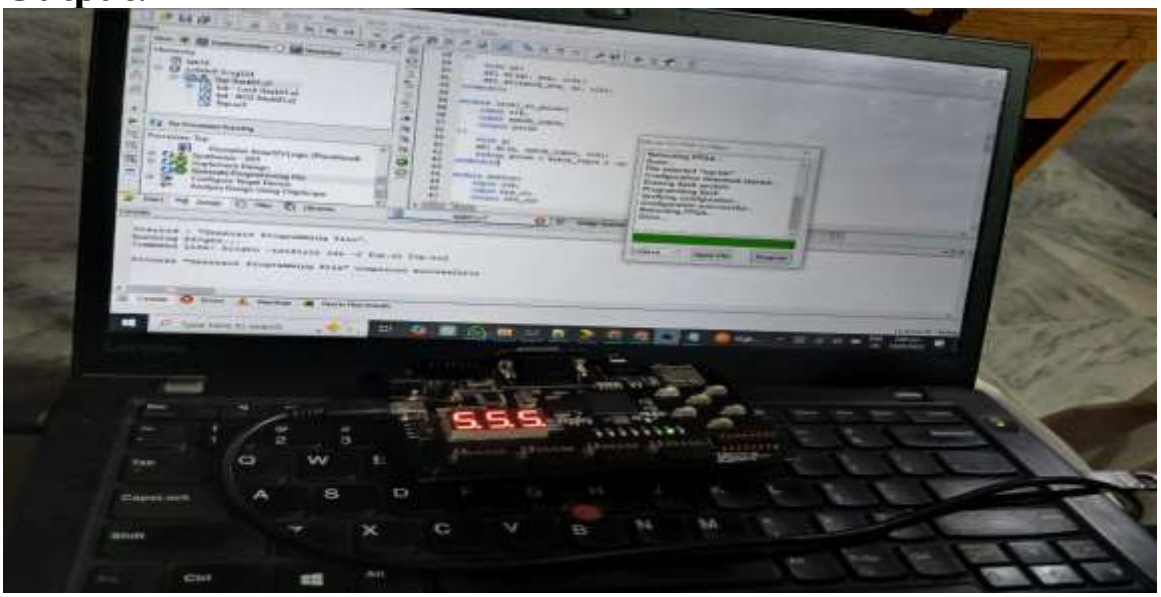
```
        state = S1;
    else if (!button0)
        state = S0;
end

S1: begin
    if (!button1) // second 1 ? 11
        state = S2;
    else if (!button0)
        state = S0;
end

S2: begin
    if (!button0) // zero ? 110
        state = S3;
    else if (!button1)
        state = S0;
end

S3: begin
    if (!button1) // fourth 1 ? 1101
        state = S4;
    else if (!button0)
```

```
            state = S0;
        end

        S4: begin
            if (!button1) // fifth 1 ? 11011
                state = S5;
            else if (!button0)
                state = S0;
        end

        S5: begin
            if (!button1)
                state = S0;
            else if (!button0)
                state = S0;
        end
    endcase

            end                                 //else
                PS <= state;
                if(PS==S5)
                        openLock = 1;
                else
```

```
            openLock = 0;

    end                                 // always end


endmodule
module Top(BT0, BT1, clk, rst,  Unlock, seg);


        input BT0, BT1, rst, clk;
        output Unlock;
        wire [2:0]PS;
        output [10:0] seg;

        Lock lck(BT0, BT1, clk, rst, Unlock, PS);
        BCD bd(PS,seg);

endmodule
```



Output:



Conclusion:

This Verilog code implements a digital lock system that unlocks when the button sequence `01011` is entered. It uses a finite state machine (FSM) to track the input pattern, a clock divider to slow down the clock, and a 7-segment display module to show the current FSM state. When the correct sequence is entered, the `Unlock` output goes high.