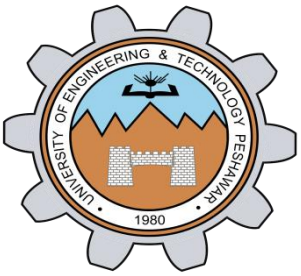
Lab Report #10



Digital System Design LAB

Submitted By: JAWAD IRFAN

Registration No: 21PWCSE2009

Section: B

Submitted To: Shahzada Fahim Jan

"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:

JAWAD IRFAN

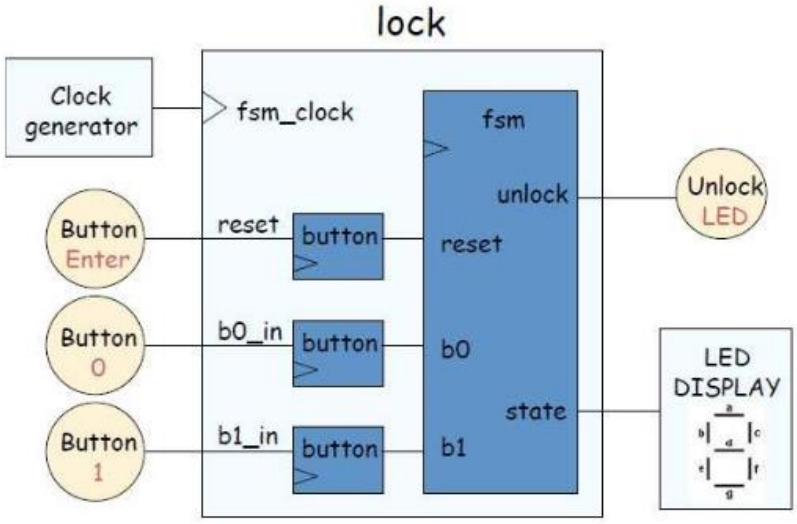
Department of Computer Systems Engineering

University of Engineering and Technology Peshawar

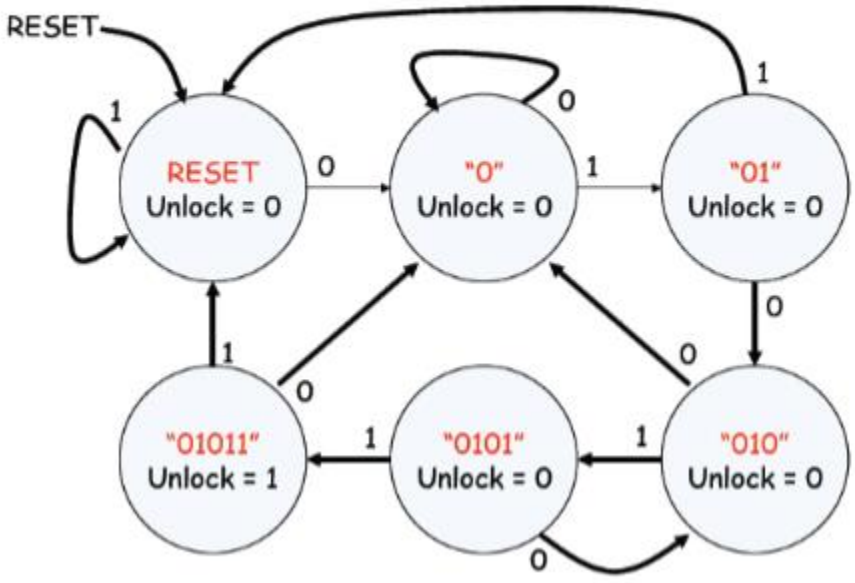
Lab 10

A Digital Lock

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



State Transition Diagram:



Lab Tasks:

1- Change the functionality of the lock such that it unlocks on the

sequence of 11011.

CODE :

```
module clock_divider(
    input clk_in,
    output reg clk_out
);
   reg [27:0] counter = 28'd0;
   parameter divisor = 28'd1000000;

    always @(posedge clk_in) begin
      counter <= counter + 28'd1;

      if (counter >= (divisor - 1))
            counter <= 28'd0;

      clk_out <=(counter < divisor/2)?1'b1:1'b0;
      end
endmodule



module D_FF(Q,D,clk,rst
    );
  input D,clk,rst;
  output reg Q;
  always @(posedge clk) begin
  if(rst)
    Q <= 1'b0;
    else
    Q<=D;
  end
```

```
module BCD_to_SevenSeg (bcd,SEVENSEG);
    input [2:0] bcd;
    output reg [10:0] SEVENSEG;


  always @(*) begin
    case(bcd)
      3'b000: SEVENSEG = 7'b1000000; // 0
      3'b001: SEVENSEG= 7'b1111001; // 1
      3'b010: SEVENSEG = 7'b0100100; // 2
      3'b011: SEVENSEG = 7'b0100000; // 3
      3'b100: SEVENSEG = 7'b1011001;// 4
      3'b101: SEVENSEG = 7'b0010010;//5

      default: SEVENSEG = 7'b1000000;  // 0
    endcase
  end

endmodule
```

```
module level_to_pulse(
    input synch_input,
    input clk,
    input rst,
    output pulse
     );

     wire Q;

     D_FF df(Q, synch_input, clk, rst);
     and a(pulse ,~Q, synch_input);


endmodule
```

```
////////////////////////////////////////////////////////////////////////////////
module synchronizer(
    input clk,
    input btn,
    input rst,
    output synch_btn
     );

    wire Q1;

    D_FF df1(Q1, btn, clk, rst);
    D_FF df2(synch_btn, Q1, clk, rst);

endmodule
```

```
//////////////////////////////////////////////////////////////////////////////////
module btn_module(
    input btn,
    input CLK,
    input RST,
    output pulse
     );
    wire synch_btn;
    synchronizer s1(CLK, btn, RST, synch_btn);
    level_to_pulse lp1(synch_btn, CLK, RST, pulse);

endmodule

//////////////////////////////////////////////////////////////////////////////////
module lock_fsm2(
   input btn0,
   input btn1,
   input clk,
   input RST_BTN,
   output reg led,
   output reg [3:0]bcd
);
   reg [2:0]state;

   parameter s0 = 3'b000;
   parameter s1 = 3'b001;
   parameter s2 = 3'b010;
   parameter s3 = 3'b011;
   parameter s4 = 3'b100;
   parameter s5 = 3'b101;

   always@(posedge clk) begin
      if(RST_BTN)
         state <= s0;

      else
         case(state)
            s0: begin
               if(btn0)
                  state <= s0;
               else if(btn1)
                  state <= s1;
```

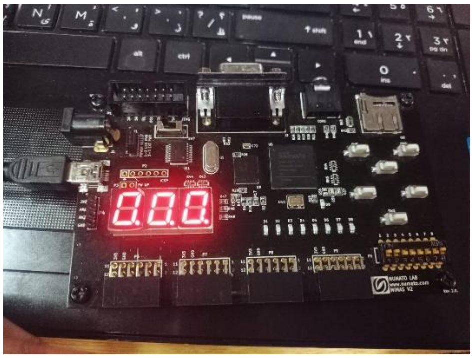
```
49                   else
50                      state <= s0;
51               end
52
53               s1: begin
54                   if(btn0)
55                      state <= s1;
56                   else if(btn1)
57                      state <= s2;
58                   else
59                      state <= s1;
60               end
61
62               s2: begin
63                   if(btn0)
64                      state <= s3;
65                   else if(btn1)
66                      state <= s2;
67                   else
68                      state <= s2;
69               end
70
71               s3: begin
72                   if(btn0)
73                      state <= s3;
74                   else if(btn1)
75                      state <= s4;
76                   else
77                      state <= s3;
78               end
```

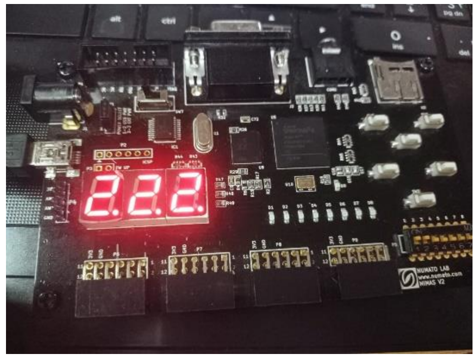
```
78               end
79
80               s4: begin
81                  if(btn0)
82                     state <= s4;
83                  else if(btn1)
84                     state <= s5;
85                  else
86                     state <= s4;
87               end
88
89               s5: begin
90                  if(btn0)
91                     state <= s5;
92                  else if(btn1)
93                     state <= s0;
94                  else
95                     state <= s5;
96               end
97           endcase
98
99       end
100
101
102      always@(*)
103         case(state)
104            s0: begin
105               led <= 0;
106               bcd <= 4'b0000;
107            end
```

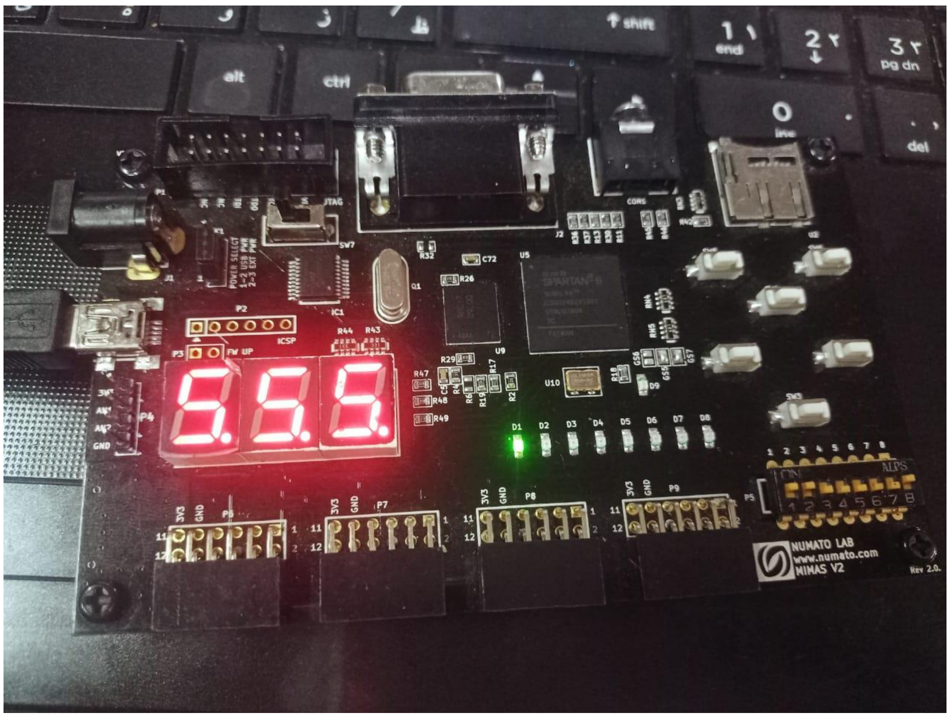
```
module Top( CLK, RST, RST_BTN, BTN0, BTN1, LED, SEVENSEG);

   input CLK, RST, BTN0, BTN1, RST_BTN;

   output LED;
   output [7:0]SEVENSEG;

   wire SLOW_CLOCK;
   //wire synch_btn0, synch_btn1, synch_rst;

   wire pulse0, pulse1, RST_Pulse;

   wire [3:0]bcd;

   clock_divider divider(CLK, SLOW_CLOCK);
   lock_fsm2 my_fsm( pulse0, pulse1, SLOW_CLOCK, RST_Pulse, LED, bcd);

   btn_module b0(BTN0, SLOW_CLOCK, RST,  pulse0);
   btn_module b1(BTN1, SLOW_CLOCK, RST,  pulse1);
   btn_module b2(RST_BTN, SLOW_CLOCK, RST,  RST_Pulse);

   BCD_to_SevenSeg(bcd, SEVENSEG);

endmodule
```

```
        end

        s1: begin
           led <= 0;
           bcd <= 4'b0001;
        end


        s2: begin
           led <= 0;
           bcd <= 4'b0010;
        end

        s3: begin
           led <= 0;
           bcd <= 4'b0011;
        end

        s4: begin
           led <= 0;
           bcd <= 4'b0100;
        end

        s5: begin
           led <= 1;
           bcd <= 4'b0101;
        end
     endcase

endmodule
```

OUTPUT:






NUMATO LAB
www.numato.com
MIMAS V2
Rev 2.0
SPARTAN 6
ICSP
JTAG
ALPS