University of Engineering and Technology, Peshawar
Department of Computer Systems Engineering.
Course Lab: CSE-308 Digital System Design

Zubair Khan **19** PWCSE **1797**
Section A
Batch 21 (Spring_2022)
Submitted to Engr. Madiha Sher

LAB # 8 TITLE

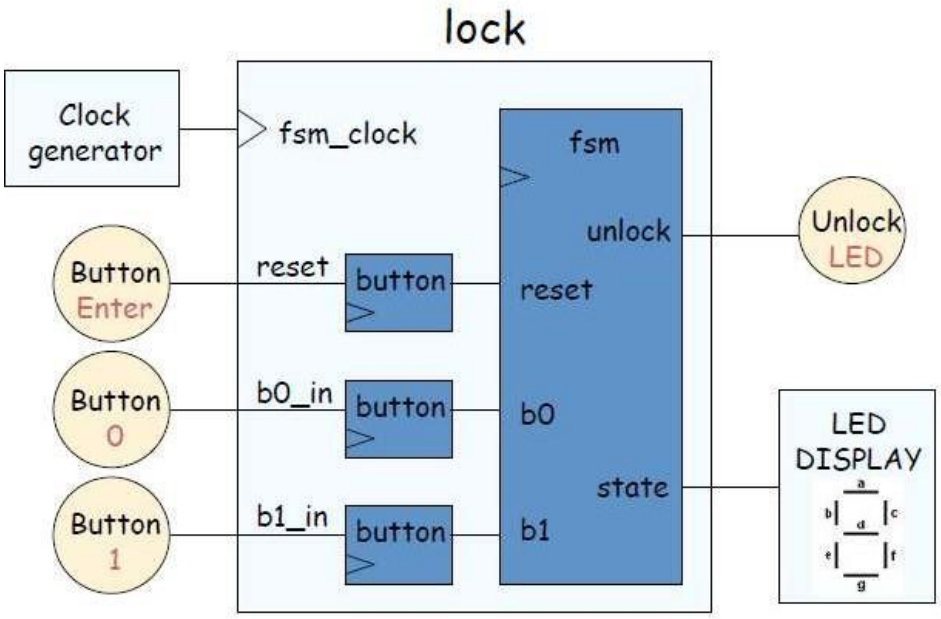
A DIGITAL LOCK

OBJECTIVE:

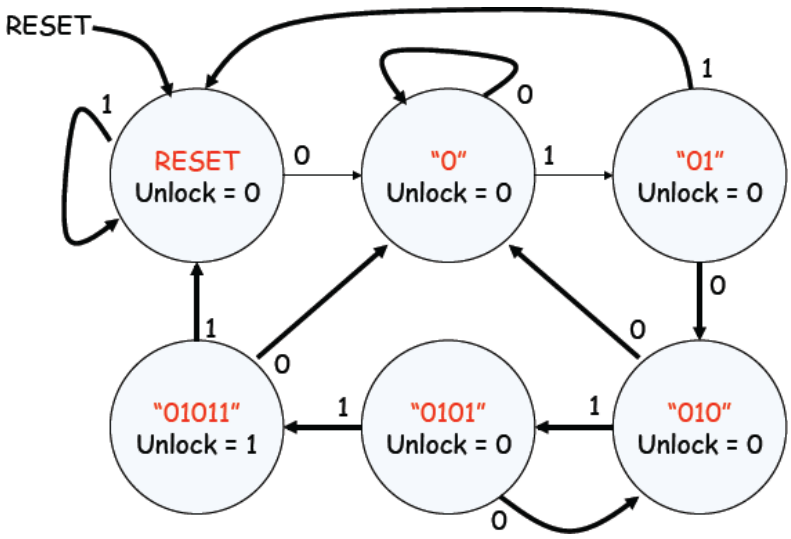
Build an electronic combination lock with a reset button, two number buttons (0 and 1), and an unlock output. The combination should be “01011”

BLOCK DIAGRAM:

A combinational digital lock has three input buttons for Reset, entering a "0" and entering a "1" and the 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. **Change the functionality of the lock such that it unlocks on the sequence of 11011.**

CODE:

```
module lock(clk,rst,on,off,out,one,zero,segcode,en);

output reg out;
input clk,rst,on,off;
output one,zero,en;
output segcode;
wire synin, CLK_1hz;
reg [2:0]state, next_state;
wire [6:0]segcode;

CLOCK_Divider CD1(clk,rst,CLK_1hz);
synchronizer ss1(on,CLK_1hz,rst,synon);
synchronizer ss2(off,CLK_1hz,rst,synoff);
level2pulse t1(CLK_1hz,rst,synon,one);
level2pulse t2(CLK_1hz,rst,synoff,zero);
sevensegment seg(state,segcode);
```

```
assign en = 0;

parameter s0=0;
parameter s1=1;
parameter s2=2;
parameter s3=3;
parameter s4=4;
parameter s5=5;

always @(posedge CLK_1hz)
begin
                if(~rst)
                 begin
                        state=s0;
                 end
                else
                 state=next_state;
end

always @(*)
begin
                case(state)
                 s0:
                 begin
                        if(one)
                        begin
                                next_state=s0;
                                out=0;
                        end
                        else if (zero)
                        begin
                                next_state=s1;
                                out=0;
                        end
                        else
                        begin
                                next_state = state;
                                out = out;
                        end
                 end
```

```
s1:
begin
        if(one)
        begin
                next_state=s2;
                out=0;
        end
        else if(zero)
        begin
                next_state=s1;
                out=0;
        end
        else
        begin
                next_state = state;
                out = out;
        end
end
s2:
begin
        if(one)
        begin
                next_state=s0;
                out=0;
        end
        else if(zero)
        begin
                next_state=s3;
                out=0;
        end
        else
        begin
                next_state = state;
                out = out;
        end
end
s3:
begin
        if(one)
        begin
```

```
                next_state=s4;
                out=0;
        end
        else if(zero)
        begin
                next_state=s1;
                out=0;
        end
        else
        begin
                next_state = state;
                out = out;
        end
end
s4:
begin
        if(one)
        begin
                next_state=s5;
                out=1;
        end
        else if(zero)
        begin
                next_state=s1;
                out=0;
        end
        else
        begin
                next_state = state;
                out = out;
        end
end
s5:
begin
        if(one)
        begin
                next_state=s0;
                out=1;
        end
        else if(zero)
```

```
                        begin
                                next_state=s1;
                                out=1;
                        end
                        else
                        begin
                                next_state = state;
                                out = 0;
                        end
                end
                default:
                begin
                        next_state = s0;
                        out = 0;
                end
                endcase
end
endmodule

module level2pulse(CLK_1hz,rst,synin,out);
input rst,CLK_1hz,synin;
output out;

parameter s0=0;
parameter s1=1;
reg state, next_state,out;

always @(posedge CLK_1hz)
begin
                if(~rst)
                begin
                 state=s0;
                 end
                else
                 state=next_state;
end
always @(*)
begin
                case(state)
                 s0:
```

```
                begin
                        if(synin==0)
                        begin
                                next_state=s0;
                                out=0;
                                end
                        else
                        begin
                                next_state=s1;
                                out=1;
                                end
                end
                s1:
                begin
                        if(synin==1)
                        begin
                                next_state=s1;
                                out=0;
                        end
                        else
                        begin
                                next_state=s0;
                                out=0;
                        end
                end
               endcase
end
endmodule

module synchronizer(in, clk, rst,synin);

input in,clk,rst;
output synin;

wire out;

D_FF ff1(out,in,clk,rst);
D_FF ff2(synin,out,clk,rst);

endmodule
```

```
module D_FF (q, d, clock, reset);

		output q;
		input d, clock, reset;

		reg q;

		always @(posedge clock)
		begin
		 if (~reset)
			q = 1'b0;
		 else
			q = d;
		end

endmodule

module CLOCK_Divider(input CLK_100MHz,input RESET,output reg CLK_1hz);

		integer c=0;
		always @(posedge CLK_100MHz)
		begin
		 if(~RESET)
		 begin
			c = 0;
			CLK_1hz=1;
		 end
		 else
		 begin
			c = c+1;
			if(c==10000000)
			begin
				CLK_1hz = ~CLK_1hz;
				c=0;
			end
		 end
		end
endmodule
```

```
module sevensegment(in, out);

output [6:0] out;
input [2:0] in;
assign out =
  (in==0)?(7'b1000000):
    (in==1)?(7'b1111001):
    (in==2)?(7'b0100100):
    (in==3)?(7'b0110000):
    (in==4)?(7'b0011001):
    (in==5)?(7'b0010010):
    (in==6)?(7'b0000010):
    (in==7)?(7'b1111000):
                (in==8)?(7'b0000000):
                (in==9)?(7'b0010000): (7'b1111111);
endmodule
```

UCF FILE:

```
NET &quot;clk_100Mhz&quot; LOC = V10;
NET &quot;seg7[0]&quot; LOC = A3;
NET &quot;seg7[2]&quot; LOC = A4;
NET &quot;seg7[1]&quot; LOC = B4;
NET &quot;seg7[6]&quot; LOC = C6;
NET &quot;seg7[5]&quot; LOC = D6;
NET &quot;seg7[4]&quot; LOC = C5;
NET &quot;seg7[3]&quot; LOC = C4;
NET &quot;one&quot; LOC = F17;
NET &quot;reset&quot; LOC = E16;

NET &quot;zero&quot; LOC = F18;
NET &quot;clk_100Mhz&quot; PULLUP;
NET &quot;one&quot; PULLUP;
NET &quot;zero&quot; PULLUP;
NET &quot;reset&quot; PULLUP;
# PlanAhead Generated physical constraints
NET &quot;out&quot; LOC = P15;
```

OUTPUT: