**Lab report no 8**



**Fall 2022**

**CSE-308L Digital Systems Design Lab**

**Submitted By**

**Names Registration No**

# Muhammad Ali 19pwcse1801

Section: **A**

**Date**:22,6,22

**Submitted To: MAM. Madiha Sher**

Department of Computer Systems Engineering

University of Engineering and Technology, Peshawar

**Lab Tasks:**

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

**sequence of 11011.**

**Code: -**

module Lock\_SD(seg7,out,zero,one,clk\_100Mhz,reset);

input zero,one,clk\_100Mhz,reset;

output reg out;

output [6:0] seg7;

wire clk\_1Mhz;

wire syn\_zero,syn\_one;

reg [2:0]state,next\_state;

CLOCK\_Divider cd(clk\_1Mhz,clk\_100Mhz,reset);

synchronizer inst1(syn\_zero,clk\_100Mhz,reset,zero);

synchronizer inst2(syn\_one,clk\_100Mhz,reset,one);

level2pulse l2p1(pulse\_zero,clk\_1Mhz,reset,syn\_zero);

level2pulse l2p2(pulse\_one,clk\_1Mhz,reset,syn\_one);

seven\_seg\_Dec seg1(seg7,state);

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

always @(posedge clk\_1Mhz)

if(~reset)

state=s0;

else

state=next\_state;

always @(\*)

begin

case(state)

s0:

if(pulse\_zero)

begin

next\_state=s1;

out=0;

end

else if(pulse\_one)

begin

next\_state=s0;

out=0;

end

else

begin

next\_state=state;

out=out;

end

s1:

if(pulse\_zero)

begin

next\_state=s1;

out=0;

end

else if(pulse\_one)

begin

next\_state=s2;

out=0;

end

else

begin

next\_state=state;

out=out;

end

s2:

if(pulse\_zero)

begin

next\_state=s3;

out=0;

end

else if(pulse\_one)

begin

next\_state=s0;

out=0;

end

else

begin

next\_state=state;

out=out;

end

s3:

if(pulse\_zero)

begin

next\_state=s1;

out=0;

end

else if(pulse\_one)

begin

next\_state=s4;

out=0;

end

else

begin

next\_state=state;

out=out;

end

s4:

if(pulse\_zero)

begin

next\_state=s1;

out=0;

end

else if(pulse\_one)

begin

next\_state=s5;

out=0;

end

else

begin

next\_state=state;

out=out;

end

s5:

if(pulse\_zero)

begin

next\_state=s1;

out=0;

end

else if(pulse\_one)

begin

next\_state=s0;

out=1;

end

else

begin

next\_state=state;

out=out;

end

default:

begin

next\_state = s0;

out=0;

end

endcase

end

endmodule

module CLOCK\_Divider(clk\_1Mhz,clk\_100Mhz,reset);

output reg clk\_1Mhz;

input clk\_100Mhz,reset;

integer c=0;

always @(posedge clk\_100Mhz)

begin

if(~reset)

begin

c = 0;

clk\_1Mhz=1;

end

else

begin

c = c+1;

if(c==50000000)

begin

clk\_1Mhz = ~clk\_1Mhz;

c=0;

end

end

end

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 synchronizer(syn\_out,clk\_100Mhz,reset,in);

input in,clk\_100Mhz,reset;

output syn\_out;

wire out;

D\_FF ff0(out,in,clk\_100Mhz,reset);

D\_FF ff1(syn\_out,out,clk\_100Mhz,reset);

endmodule

module level2pulse(pulse\_out,clk\_1Mhz,reset,syn\_in);

input clk\_1Mhz, reset,syn\_in;

output reg pulse\_out;

parameter s0=0;

parameter s1=1;

reg state, next\_state,out;

always @(posedge clk\_1Mhz)

begin

if(~reset)

begin

state=s0;

end

else

state=next\_state;

end

always @(\*)

begin

case(state)

s0:

begin

if(syn\_in==0)

begin

next\_state=s0;

pulse\_out=0;

end

else

begin

next\_state=s1;

pulse\_out=1;

end

end

s1:

begin

if(syn\_in==1)

begin

next\_state=s1;

pulse\_out=0;

end

else

begin

next\_state=s0;

pulse\_out=0;

end

end

endcase

end

endmodule

module seven\_seg\_Dec(seg,in);

input [2:0]in;

output [6:0]seg;

assign seg=(in==3'b000)? 7'b1000000:

(in==3'b001)? 7'b1111001:

(in==3'b010)? 7'b0100100:

(in==3'b011)? 7'b0110000:

(in==3'b100)? 7'b0011001:

(in==3'b101)? 7'b0010010:

(in==3'b110)? 7'b0000010:

(in==3'b111)? 7'b1111000:7'b1111111;

Endmodule

**I/O Port of generating programing: -**

INPUT/OUTPUT PLANNING

NET "clk\_100Mhz" LOC = V10;

NET "seg7[0]" LOC = A3;

NET "seg7[2]" LOC = A4;

NET "seg7[1]" LOC = B4;

NET "seg7[6]" LOC = C6;

NET "seg7[5]" LOC = D6;

NET "seg7[4]" LOC = C5;

NET "seg7[3]" LOC = C4;

NET "one" LOC = F17;

NET "reset" LOC = E16;

NET "zero" LOC = F18;

NET "clk\_100Mhz" PULLUP;

NET "one" PULLUP;

NET "zero" PULLUP;

NET "reset" PULLUP;

# PlanAhead Generated physical constraints

NET "out" LOC = P15;

**Output: -**

