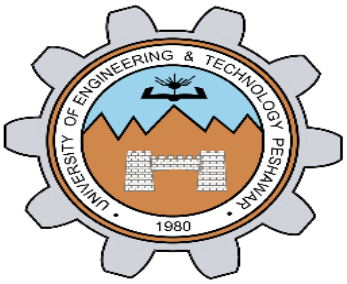
Lab report no 8



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;

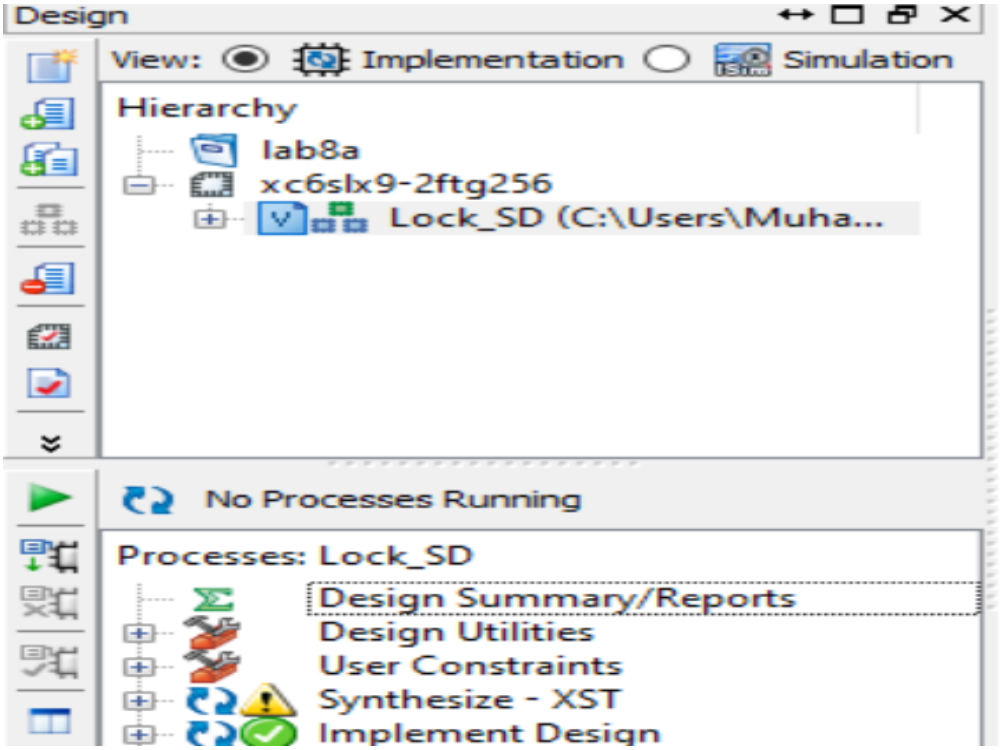
NET "zero" PULLUP;

NET "reset" PULLUP;

PlanAhead Generated physical constraints

NET "out" LOC = P15;

Output: -


Design
View:
Implementation
Simulation
Hierarchy
lab8a
xc6slx9-2ftg256
Lock_SD (C:\Users\Muha...
No Processes Running
Processes: Lock_SD
Design Summary/Reports
Design Utilities
User Constraints
Synthesize - XST
Implement Design