FAMU- FSU College of Engineering

Department of Electrical and Computer Engineering Spring 2024 Semester

EEL3705L Digital Logic Lab Report

Section No: 0002

Lab Instructor: Bruce Harvey

Lab No: Lab Project

Lab Title: Digital Combination Lock

Name: Ruth Massock

Partner's Name: Christopher Balthazar

Date Performed: 4/24/2024

Date Delivered: 4/25/2024

Contents

1 Introduction

In this lab project, we were assign to design and implement digital combination lock using the DE1-SoC board. Module locks are fundamental components in digital systems used for security and access control. They require users to input a specific sequence or password to gain access to a system or module. In this experiment, we implemented a module lock using Verilog HDL, targeting an FPGA or ASIC platform. The module lock design comprised state machines and combinational logic to validate the password input and control the lock's behavior accordingly. Our password was 11-17-46.

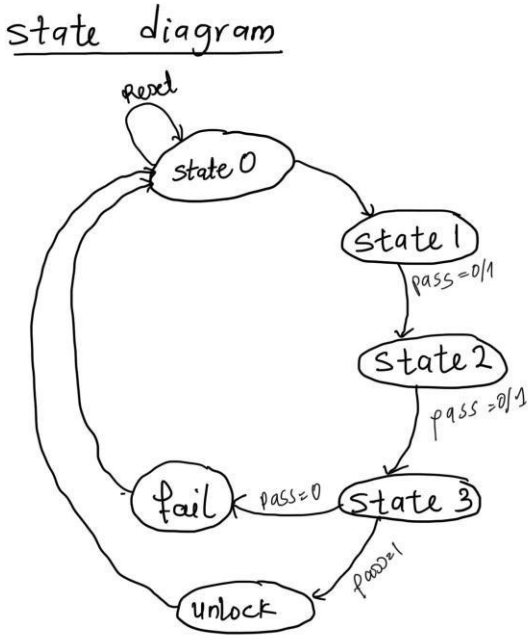
2 Requirements

This lab required some of the following to make this digital combinational lock The lock should have the following states: STATE_s0, STATE_s1, STATE_s2, STATE_s3, STATE_unlock, and STATE_invalid. The lab had the following input ports: Clk: Clock input. Reset: Reset signal to initialize or reset the lock. pass1: 4-bit input for the first part of the password.pass2: 4-bit input for the second part of the password. The lab had the following output ports: out: Output signal indicating whether the lock is unlocked (1) or locked (0). hex0, hex1, hex2, hex3, hex4, hex5: Seven-segment display outputs to visualize the lock status.

3 Theoretical Design

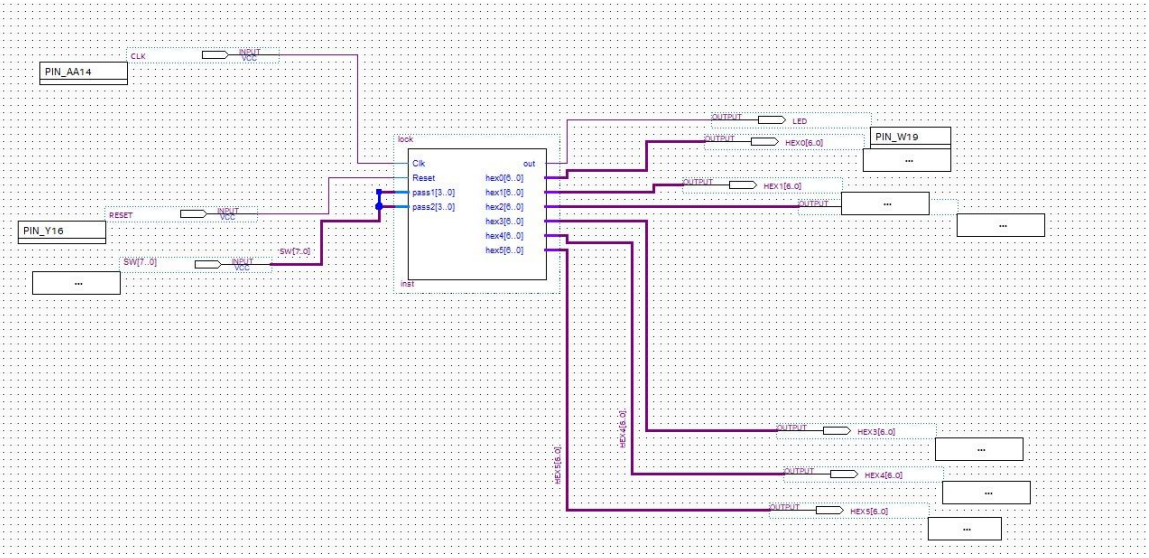
Design Narrative

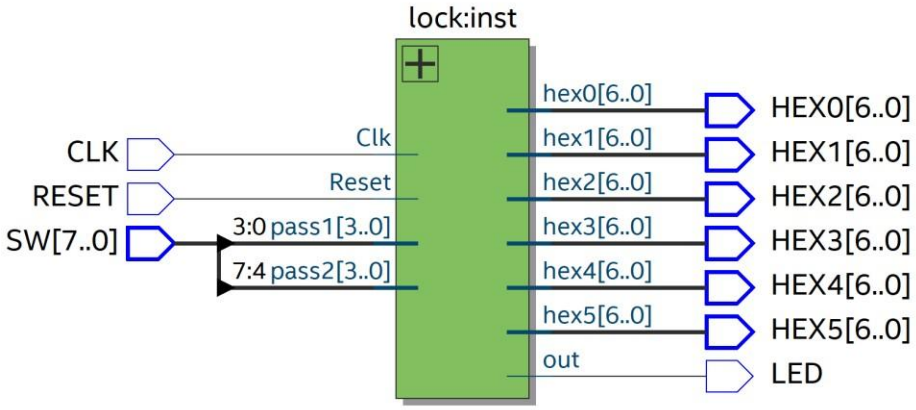
Our Verilog HDL module lock employs state machine logic and password verification for secure access control. It transitions between six states, unlocking upon correct password input. Synchronous logic ensures accurate transitions, while seven-segment displays offer visual feedback. Validation through simulation ensures reliable operation under diverse conditions, making our design a dependable solution for digital access control.



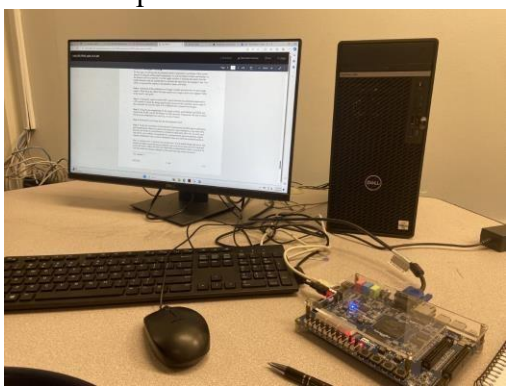
Top-level design

Top level design





Hardware setup



Functional description of components

LED5 on the DE1 Soc Board serves as output to show lock is unlocked by lighting up(extra credit). Hex5 to Hex0 display entered password , unlock results of ||||| and failed results of ------. Switches from SW7 to SW0 are used to enter a pair of password number, key0 serves as a clock and key3 serves as the reset button

4 Synthesized Design

Verilog code for Combinational lock

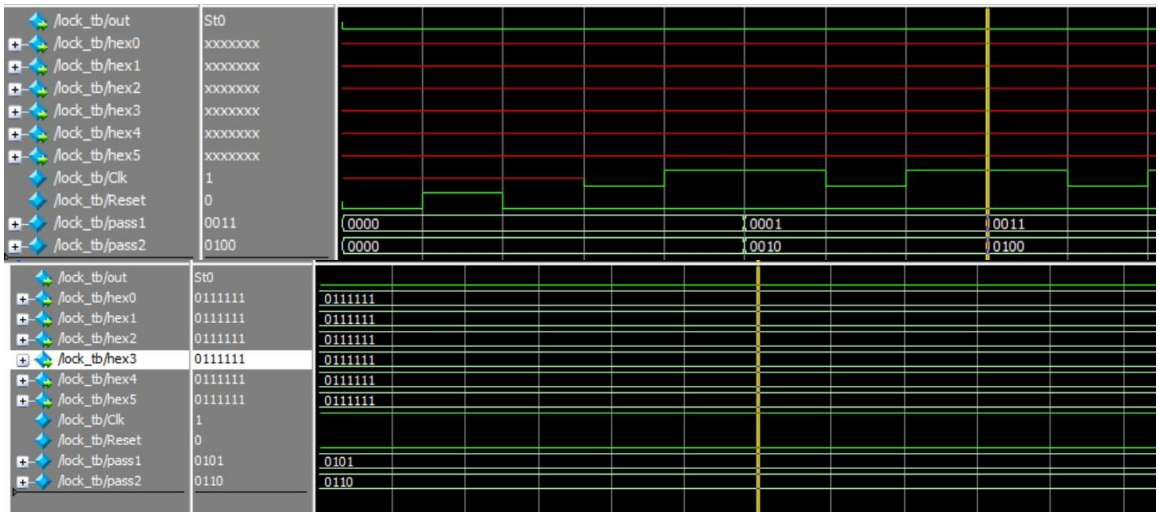
```
1   //Names: Christopher Balthazar and Ruth Massock
2   //Semester: Spring 2024
3   //Lab Section:section 0002 Friday at 9pm
4   //Date Created: 04/24/2024
5   //Description: this code is lock that displays
6   //|||||| for unlock ----- for failed and shows numbers been entered
7   module lock (
8       input wire Clk,
9       input wire Reset,
10      input wire [3:0] pass1,
11      input wire [3:0] pass2,
12      output reg out,
13      output reg [6:0] hex0,     // Seven-segment display outputs
14      output reg [6:0] hex1,
15      output reg [6:0] hex2,
16      output reg [6:0] hex3,
17      output reg [6:0] hex4,
18      output reg [6:0] hex5
19  );
20
21  // Define states
22  localparam STATE_s0=3'b000,
23  STATE_s1=3'b001,
24  STATE_s2=3'b010,
25  STATE_s3=3'b011,
26  STATE_unlock = 3'b100,
27  STATE_invalid = 3'b101;
28
29  // Define state registers
30  reg [2:0] CurrentState,NextState;
31
32  // Define password array
33  reg [3:0] password [0:5];
34
35  // Seven-segment display outputs
36  reg [6:0] z;
37  reg [6:0] zz;
38
39  // Initialize password array: 11-17-46
40  initial begin
41      password[0] = 4'b0001; //1
42      password[1] = 4'b0001;  //1
43      password[2] = 4'b0001; //1
44      password[3] = 4'b0111; //7
45      password[4] = 4'b0100;  //4
46      password[5] = 4'b0110;  //6
47  end
48
```

```
49  // State transition logic
50  always @(posedge Clk) begin
51      if (Reset)
52          CurrentState <= STATE_s0;
53
54      else
55      CurrentState <= NextState;
56  end
57  always @(*)  begin
58
59  NextState=CurrentState;
60      case (CurrentState)
61          STATE_s0: begin
62              NextState = STATE_s1;
63          end
64          STATE_s1: begin
65          if(pass1 == password[0] && pass2 == password[1])
66              NextState = STATE_s2;
67          end
68          STATE_s2: begin
69          if(pass1 == password[2] && pass2 == password[3])
70              NextState = STATE_s3;
71          end
72          STATE_s3: begin
73              if (pass1 == password[4] && pass2 == password[5])
74                  NextState = STATE_unlock;
75            else
76                NextState = STATE_invalid;
77          end
78          STATE_unlock: begin
79              if (pass1 == 4'b0000 && pass2 == 4'b0000)
80                  NextState = STATE_s0;
81              else
82                  NextState = STATE_invalid;
83          end
84          STATE_invalid: begin
85              if (pass1 == 4'b0000 && pass2 == 4'b0000)
86                  NextState = STATE_s0;
87  //            else
88  //                NextState = STATE_invalid;
89          end
90      endcase
91  end
92
93  // Output logic
94  always @(*) begin
95  out=0;
96      case (CurrentState)
97
98          STATE_s1: begin
```

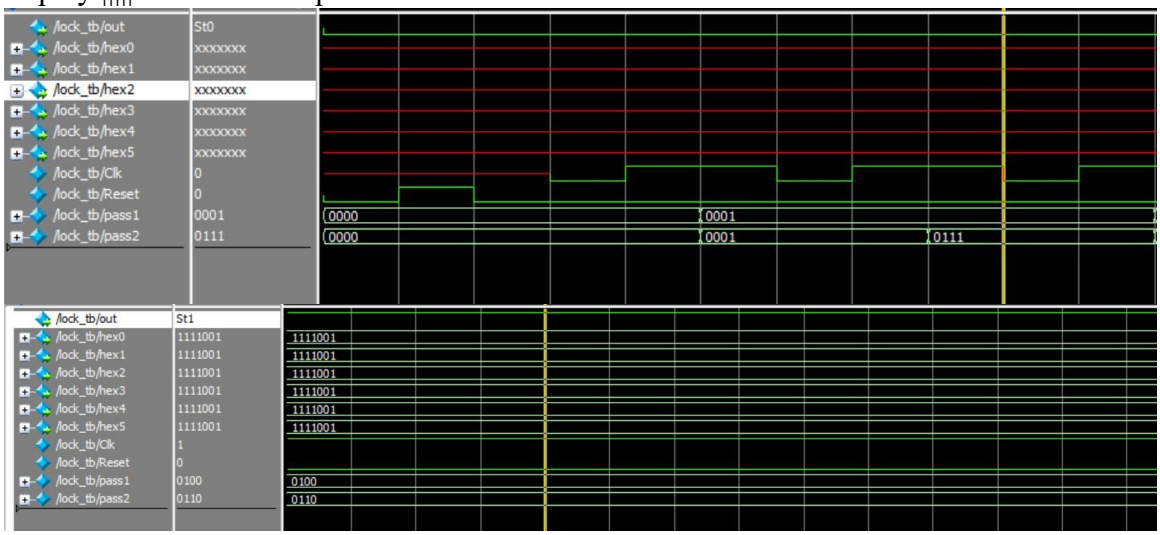
```
97          STATE_s1: begin
98              out = 0;
99              // Define outputs for STATE_s1
100           hex5 = ~z;
101           hex4 = ~zz;
102         end
103         STATE_s2: begin
104             out = 0;
105             // Define outputs for STATE_s2
106             hex3 = ~z;
107           hex2 = ~zz;
108         end
109         STATE_s3: begin
110             out = 0;
111             // Define outputs for STATE_s3
112             hex1 = ~z;
113           hex0 = ~zz;
114         end
115         STATE_unlock: begin
116             out = 1;
117             // Define outputs for STATE_unlock
118             hex0 = 7'b1111001;
119              hex1 =  7'b1111001;
120             hex2 = 7'b1111001;
121              hex3 = 7'b1111001;
122              hex4 = 7'b1111001;
123              hex5 = 7'b1111001;
124         end
125         STATE_invalid: begin
126             out = 0;
127             // Define outputs for STATE_invalid
128
129             hex0 = 7'b0111111;
130              hex1 =  7'b0111111;
131             hex2 = 7'b0111111;
132              hex3 = 7'b0111111;
133              hex4 = 7'b0111111;
134              hex5 = 7'b0111111;
135         end
136         default:begin
137         out=0;
138             hex0 = 7'b111111; //blank display
139              hex1 =7'b111111;
140             hex2 = 7'b111111;
141              hex3 =7'b111111;
142              hex4 =7'b111111;
143              hex5 =7'b111111;
144         end
145     endcase
146 end
147
148 // Seven-segment display logic
149 always @(*) begin
150     case (pass1)
151         4'h0: z = 7'h3F;
152         4'h1: z = 7'h06;
153         4'h2: z = 7'h5B;
154         4'h3: z = 7'h4F;
155         4'h4: z = 7'h66;
156         4'h5: z = 7'h6D;
157         4'h6: z = 7'h7D;
158         4'h7: z = 7'h07;
159         4'h8: z = 7'h7F;
160         4'h9: z = 7'h67;
161         4'hA: z = 7'h77;
162         4'hB: z = 7'h7C;
163         4'hC: z = 7'h39;
164         4'hD: z = 7'h5E;
165         4'hE: z = 7'h79;
166         4'hF: z = 7'h71;
167         default: z = 7'h00;
168     endcase
169
170     case (pass2)
171         4'h0: zz = 7'h3F;
172         4'h1: zz = 7'h06;
173         4'h2: zz = 7'h5B;
174         4'h3: zz = 7'h4F;
175         4'h4: zz = 7'h66;
176         4'h5: zz = 7'h6D;
177         4'h6: zz = 7'h7D;
178         4'h7: zz = 7'h07;
179         4'h8: zz = 7'h7F;
180         4'h9: zz = 7'h67;
181         4'hA: zz = 7'h77;
182         4'hB: zz = 7'h7C;
183         4'hC: zz = 7'h39;
184         4'hD: zz = 7'h5E;
185         4'hE: zz = 7'h79;
186         4'hF: zz = 7'h71;
187         default: zz = 7'h00;
188     endcase
189 end
190
191
192 endmodule
```

5 Simulation Results

Displaying wrong password entered 123456 (--------) which is 0111111

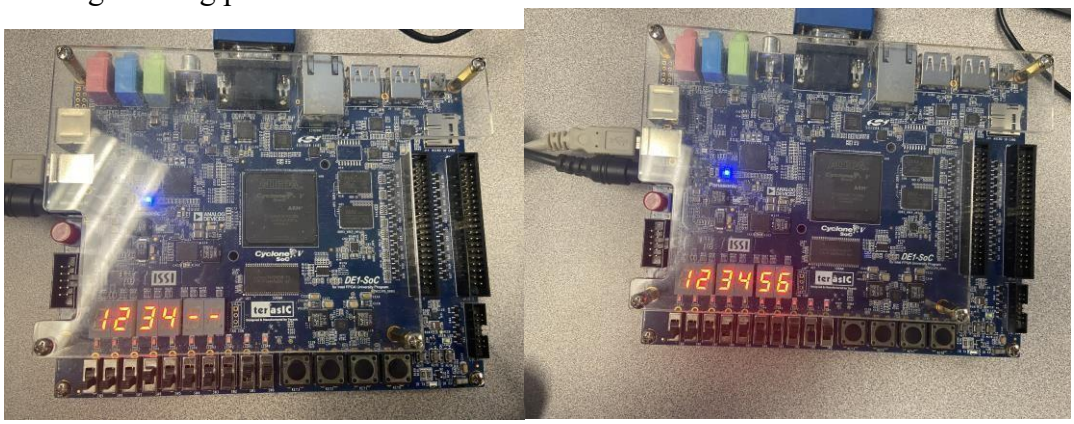


Display ||||| when correct password is entered 111746 which is 1111001

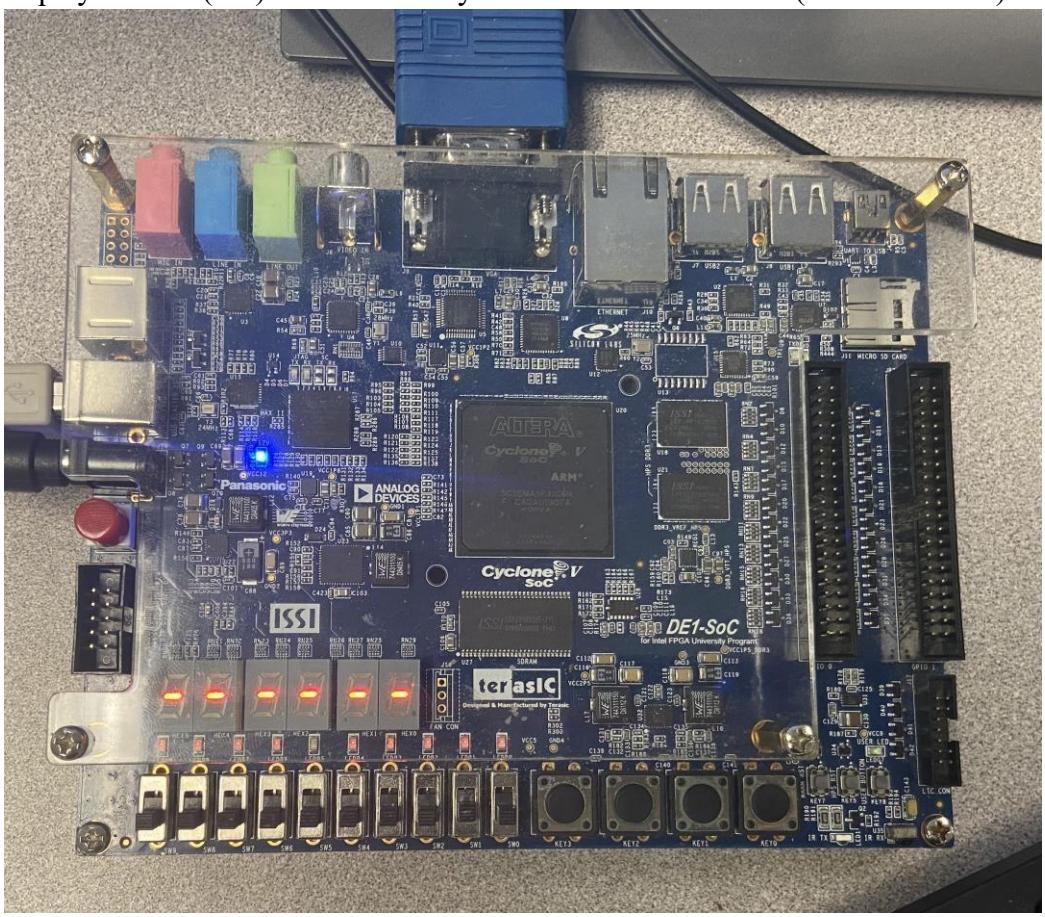


6 Experimental Results

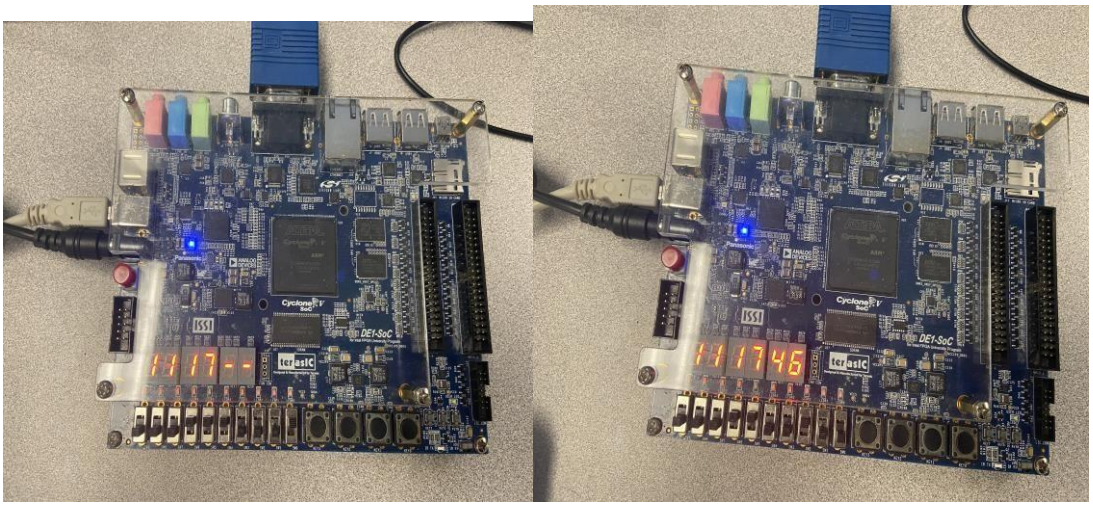
Entering a wrong password



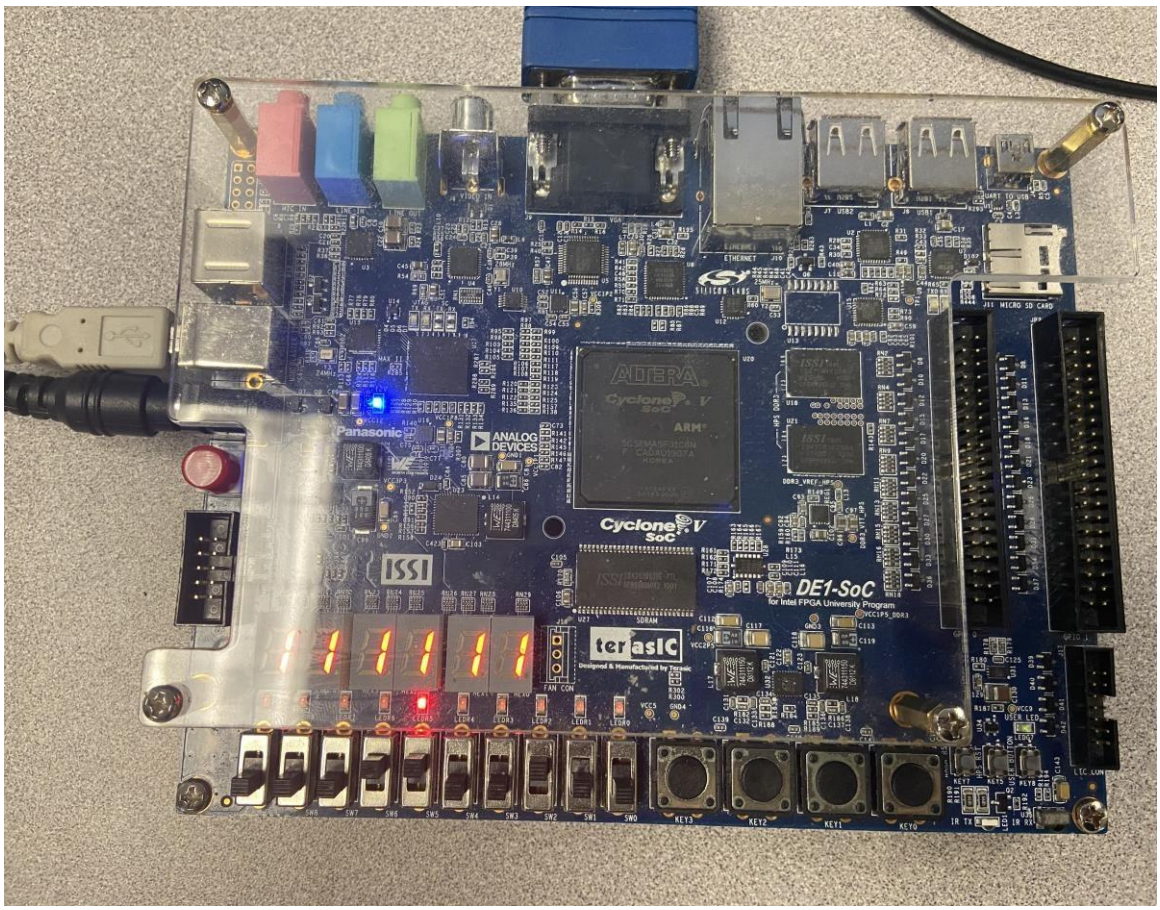
Displays Failed (----) and LED 5 stays off to show failed state (for extra credit)



Entering the correct password (111746)



Displays Unlock with (|||||) also for extra credit added an LED5 to show it is unlocj



7 Summary

Our lab Project was to design a combinational digital lock that takes a password of 8bit of 3 pairs of numbers that displays on the DE1 SoC Board and tells user if the of its wrong by ------ and ||||||| and LED ON if the correct password(11-17-46) is entered in this results are display on the board by the help of seven segment code. This experiment involved the design and implementation of a module lock using Verilog hardware description language. The module lock is a digital circuit designed to control access to a module or system by requiring the input of a specific password. The password is entered via two 4-bit inputs, and the lock transitions through different states based on the correctness of the password input. The design utilized D flip-flops for state storage and combinational logic for password verification and output generation.

8 Lessons Learned

I had problems with this project a lot like I had a hard time storing 3 pairs of numbers I could do one and it will display if wrong or not then decided to use register to store values later couldn't put it together, I knew it could work but decided finally to take the

state transition approach. I also learned the significance of proper state machine design and combinational logic in implementing access control mechanisms. Understanding how to define states, transitions, and outputs in Verilog code allows for the creation of complex digital systems with specific functionalities.

END OF DOCUMENT