CSE241 Final Project Documentation

Seongjae Shin

May 6, 2021

CSE241, Spring 2021

Table of Contents

1 SECURITY SYSTEM

1.1 SYSTEM DESCRIPTION

This system is a security system for buildings. This can be locked and unlocked when the correct password is entered into the system. And it can be turned off and has an alarm, which will go off when entered password is wrong. In this report, I will talk about a state map and how it works. Then, I will show the result of the circuit and code on System Verilog.

1.2 SYSTEM SPECIFICATIONS

Inputs: S, X, U

Outputs: Z

S is switch, X is user input, U is update input

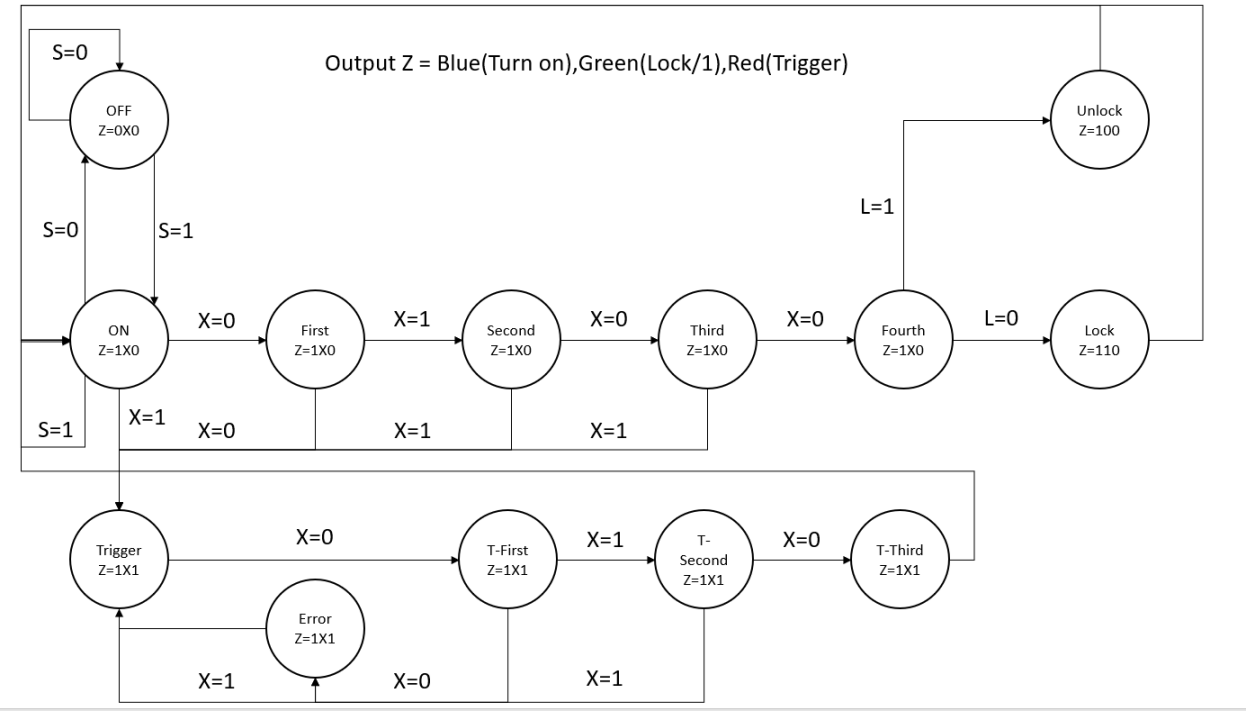
This system will have 13 states and each state would be transferred to next state based on the inputs. There is output of each state and each LED would be turned on or off based on input and variables in the system

1.3 SYSTEM METHODOLOGY

1. Try to turn on and off by switch (red light is turned on when it is on)
2. Put 0 or 1 when it is turned on
3. If it is wrong, blue light is turned on
4. After correct password is entered, the system locks or unlocks the machine. (green light is turned on when it is locked)
5. This system runs forever before it is broken by external damage.

1.4 SYSTEM DEVELOPMENT

State Diagram:



- Each state will transfer to next state when there is a corresponding input.
- There are some state without needed input. These always go to next state and only U, which is update input, is changed.

State Table:

PS	NS(S)		NS(X)		NS(L)		Output
	S = 0	S = 1	X = 0	X = 1	L = 0	L = 1	
OFF	OFF	ON	X	X	X	X	000
ON	OFF	ON	First	Trigger	X	X	100
First	X	X	Trigger	Second	X	X	1X0
Second	X	X	Third	Trigger	X	X	1X0
Third	X	X	Fourth	Trigger	X	X	1X0
Fourth	X	X	X	X	Lock	Unlock	1X0
Unlock	ON	ON	ON	ON	ON	ON	100
Lock	ON	ON	ON	ON	ON	ON	110
Trigger	X	X	T-First	Error	X	X	1X1
T-First	X	X	Error	T-Second	X	X	1X1
T-Second	X	X	T-Third	Error	X	X	1X1
T-Third	ON	ON	ON	ON	ON	ON	1X1
Error	Trigger	Trigger	Trigger	Trigger	Trigger	Trigger	1X1

- The Unlock, Lock, T-Third, and Error states always go to next state.
- X value in the output will be filled by L state. (Unlock=0, Lock=1)

State Assignment:

State	Encoding (Q3Q2Q1Q0)
OFF	0000
ON	0001
First	0010
Second	0011
Third	0100
Fourth	0101
Unlock	0110
Lock	0111
Trigger	1000
T-First	1001
T-Second	1010
T-Third	1011
Error	1100

- Each state is assigned from 0000 to 1100

Transition Table:

PS	NS(S)		NS(X)		NS(L)		Output
	S = 0	S = 1	X = 0	X = 1	L = 0	L = 1	
0000	0000	0001	XXXX	XXXX	XXXX	XXXX	000
0001	0000	0001	0010	1000	XXXX	XXXX	100
0010	XXXX	XXXX	1000	0011	XXXX	XXXX	1X0
0011	XXXX	XXXX	0100	1000	XXXX	XXXX	1X0
0100	XXXX	XXXX	0101	1000	XXXX	XXXX	1X0
0101	XXXX	XXXX	XXXX	XXXX	0111	0110	1X0
0110	0001	0001	0001	0001	0001	0001	100
0111	0001	0001	0001	0001	0001	0001	110
1000	XXXX	XXXX	1001	1100	XXXX	XXXX	1X1
1001	XXXX	XXXX	1100	1010	XXXX	XXXX	1X1
1010	XXXX	XXXX	1011	1100	XXXX	XXXX	1X1
1011	0001	0001	0001	0001	0001	0001	1X1
1100	1000	1000	1000	1000	1000	1000	1X1
1101	XXXX	XXXX	XXXX	XXXX	XXXX	XXXX	XXX
1110	XXXX	XXXX	XXXX	XXXX	XXXX	XXXX	XXX
1111	XXXX	XXXX	XXXX	XXXX	XXXX	XXXX	XXX

- XXXX and last three states are “don’t care”. This system will not care about that.

Equations:

Q3* = Q3Q1’Q0’ + Q2Q1’Q0’X + Q3Q2’Q1Q0’ + Q2’Q1Q0’X’ + Q3’Q2’Q1Q0 + Q3Q2’Q1’Q0 + Q2’Q1’Q0X[1]

[1] Appendix A

Q2* = Q3Q2'Q1'Q0'X + Q3'Q2Q1'Q0'X' + Q3Q2'Q1Q0'X + Q3'Q2'Q1Q0X' + Q3'Q2Q1'Q0 + Q3Q2'Q1'Q0X'[2]

Q1* = Q3Q2'Q1Q0'X' + Q3'Q2'Q1Q0'X + Q3'Q2Q1'Q0 + Q3Q2'Q1'Q0X + Q3'Q2'Q1'Q0X'[3][4]

Q0* = Q3'Q2'Q1'S + Q3Q2'Q1'Q0'X' + Q3'Q2Q0'X' + Q3'Q2Q1Q0' + Q3Q2'Q1Q0'X' + Q3'Q2'Q1Q0'X + Q3Q2'Q1Q0 + Q3'Q2Q1Q0 + Q3'Q2Q0L'

Z* = Q3' + Q3Q2' + Q1'Q0'[5]

- Each equation's K-map is provided at the appendix section.

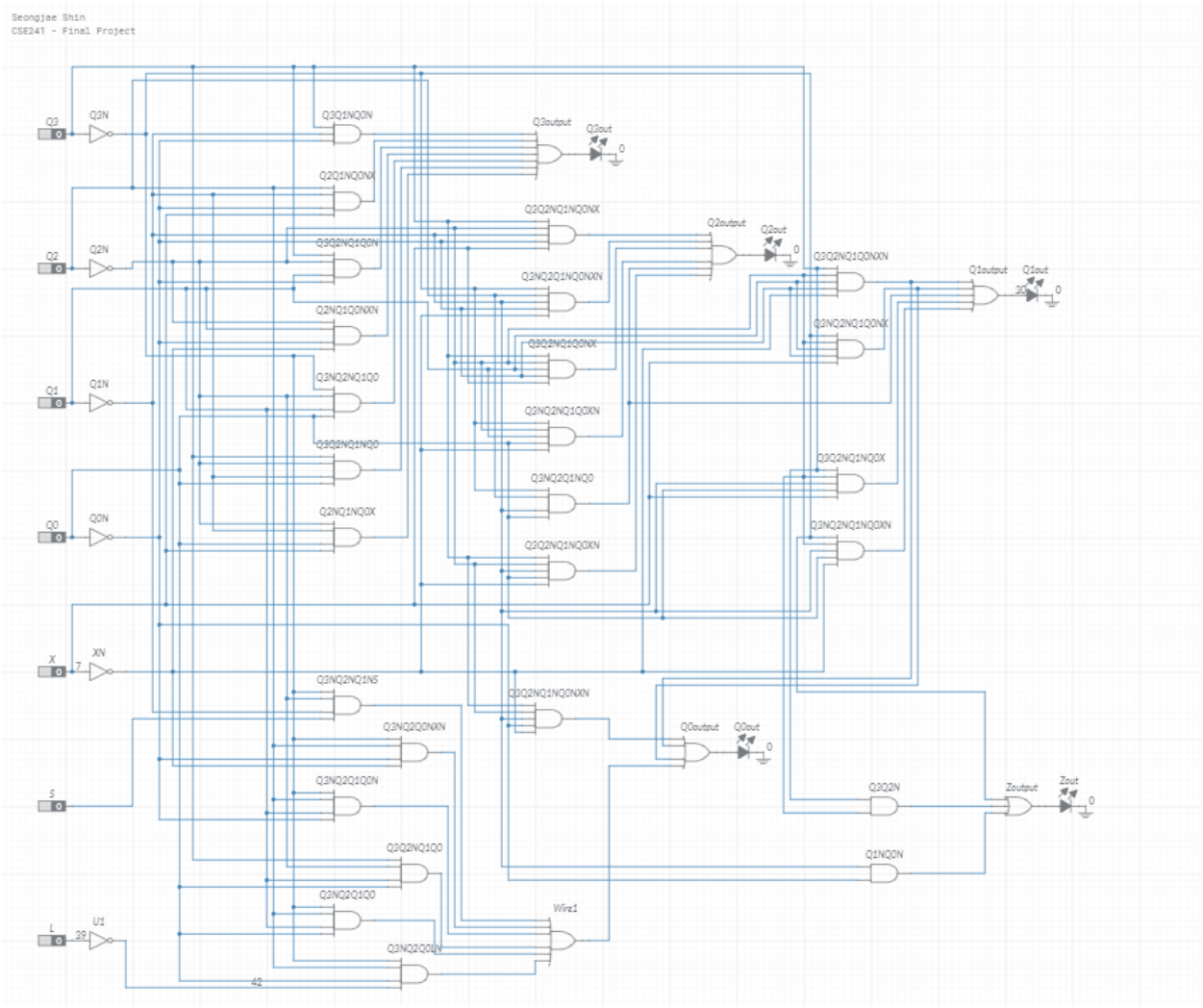
1.5 TEST CASE

Test #	PS	Input/Variable	NS	Reason
1	OFF	S = 1	ON	This can check input S
2	ON	X = 0	First	This can check input X
3	First	X = 1	Second	Process transferring to fourth with correct X
4	Second	X = 0	Third	Process transferring to fourth with correct X
5	Third	X = 0	Fourth	Process transferring to fourth with correct X
6	Fourth	L = 0	Lock	This can check variable L
7	Lock	U updated	ON	This can check it works without S and X change
8	ON	X = 1	Trigger	This can check to transfer to trigger state
9	Trigger	X = 1	Error	This can check to transfer to Error state
10	Error	U updated	Trigger	This can check it works without S and X change
11	Trigger	X = 0	T-First	Process transferring to T-Third with correct X
12	T-First	X = 1	T-Second	Process transferring to T-Third with correct X
13	T-Second	X = 0	T-Third	Process transferring to T-Third with correct X
14	T-Third	U updated	ON	This can check it works without S and X change
15	ON	X = 0	First	Process transferring to fourth when it is locked
16	First	X = 1	Second	Process transferring to fourth when it is locked
17	Second	X = 0	Third	Process transferring to fourth when it is locked
18	Third	X = 0	Fourth	Process transferring to fourth when it is locked
19	Fourth	L = 1	Unlock	This can check variable L is worked or not
20	Unlock	U updated	ON	This can check it works without S and X change

[2] Appendix B
[3] Appendix C
[4] Appendix D
[5] Appendix E

1.6 IMPLEMENTATION

Schematic Diagram:



- User can test this diagram by changing inputs.
- User will change S, X, and L from 000 to 111 for every input, Q3, Q2, Q1, and Q0.
- For example, test from 000 to 111 of SXL when 0000 of Q3Q2Q1Q0. Then, test from 000 to 111 of SXL when 0001 of Q3Q2Q1Q0. This step will be done until 1111 of Q3Q2Q1Q0 is tested except don't care parts.

System Verilog:

I choose behavioral Verilog style and two big functions; one is main and the other is deciding output.

There two big functions which are significant in this program. The main function runs case statement, which runs when update input is changed and changes present state to next state based on the inputs. After this function, the second function, which is deciding output, checks the present state, which can be changed at the main function, and decides to correspond to current variables, inputs, and state.

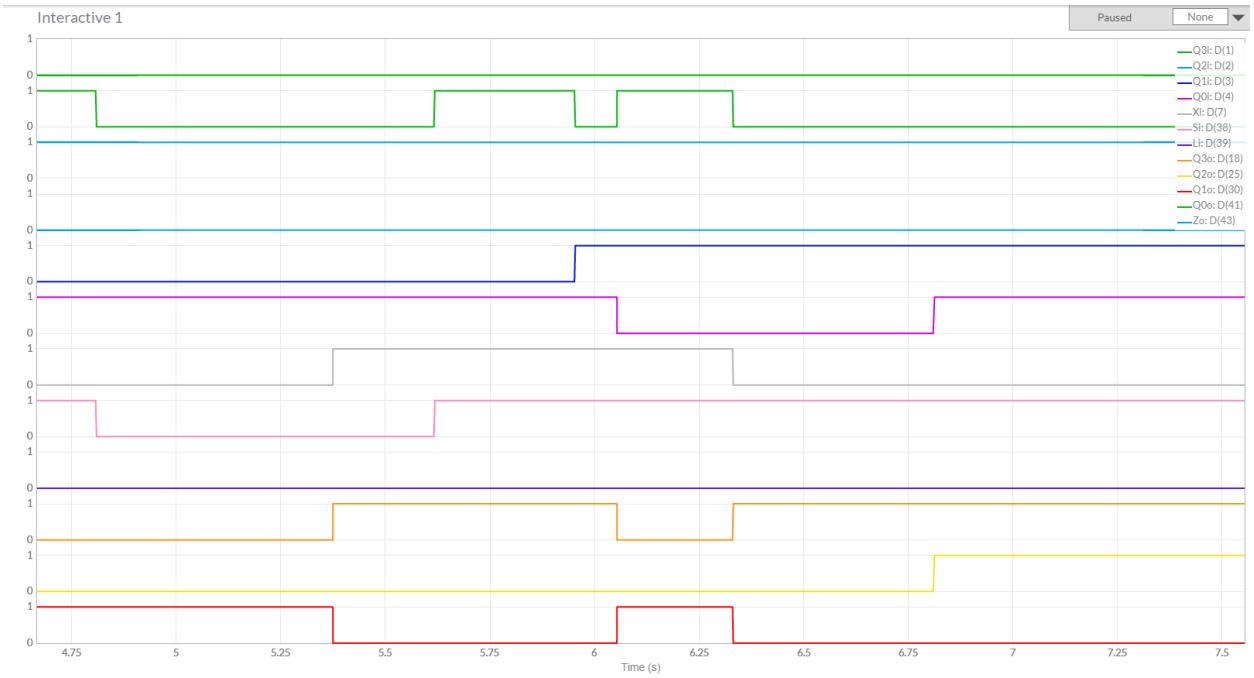
On the testbench, it will change U every second and change or stay inputs to follow to 20 test cases.

Instruction for System Verilog:

1. It is initially OFF state, so test with S = 1 at first.
2. Check output. (it should be 000->100 at first)
3. Test that it can reach to unlock or lock state with correct X value.
4. When it reaches unlock or lock state, middle value of output should be changed.
5. Test trigger state with wrong password.
6. Check right side output value. (when it is unlocked, it changes 100->101).

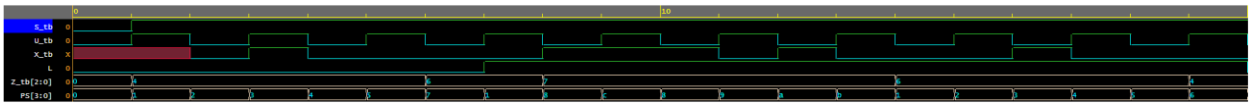
1.7 RESULT

Circuit Implementation:



- The second green line with 3 ups is Q0*. It is up when 0001 input and S is 1, and up again when 0010 input and X is 1.

System Verilog Implementation:



- At 0 second, it is OFF state. Thus, Z is 0x0 and PS is also 0x0.
- At 1 second, it is ON state and unlocked. Thus, Z is 0x4(100) and PS is 0x1(0001).
- At 6 seconds, it is Lock state. Thus, Z is 0x6(110) and PS is 0x7(0111).
- At 8 seconds, it is Trigger state and locked. Thus, Z is 0x7(111) and PS is 0x8(1000).
- At 14 seconds, it is ON state again and locked. Thus, Z is 0x6(110) and PS ix 0x1(0001).

1.8 System Improvement

1. There is a switch input and OFF state. In this system, switch input works at the OFF state and ON state. However, it needs to be worked at any state except states related to the trigger. Because users can turn off it for reset when they put 1 through 3 inputs. But trigger states are error part, it should block turning off the system for security
2. Enter and backspace keys are needed. This is a password security system and if a user puts the wrong number, it will go to the error part. However, programmers should consider the user's mistake. Thus, if a user puts the wrong number, there should be a chance to delete that number. Then, it will go to the error part when the user presses enter key, and the password is wrong.

1.9 Experience Reflection

I have some challenges in this project. One is understanding the design problem. There are a few rules to create this program and check-in 1. I get some wrong answers in check-in 1 but I can see the answer and understand the design problem. Thus, I believe check-in 1 is helpful to me. Another challenge is to create a circuit. There are long equations. I need to create a circuit readable. But it is too complex and there are many lines. This makes me spend more time on the circuit part. I can overcome it by spending more time and effort.

There is an easy part, too. Creating a system Verilog is easy. Because professor had shown how to create code, which addresses states during lecture and in the lecture slide. These provided codes help me to address states.

If I will be assigned this project again, I will apply more design problems, which is referred at the system improvement part. And I hope to test all cases.

1.10 Conclusion

I create a schematic diagram and System Verilog following figures, state table, transition table, and equations. Then, the outcomes of the two parts are matched with my transition tables and expectation. I think this project is successful.

2 Appendix

2.1 Work for 1.4

Appendix A:

Q3*

Q3Q2Q1Q0 / SXL	00 00	10 00	11 00	01 00	01 10	11 10	10 10	00 10	00 11	10 11	11 11	01 11	01 01	11 01	10 01	00 01
S=0	0	X	1	X	0	X	X	X	X	0	X	0	X	X	X	0
S=1	0	x	1	x	0	X	X	X	X	0	X	0	X	X	X	0
X=0	X	1	1	0	0	X	1	1	0	0	X	0	X	X	1	0
X=1	X	1	1	1	0	X	1	0	1	0	X	0	X	X	1	1

L=0	X	X	1	X	0	X	X	X	X	0	X	0	0	X	X	X
L=1	X	X	1	X	0	X	X	X	X	0	X	0	0	X	X	X

Appendix B:

Q2*

Q3Q2Q1Q0 / SXL	00 00	10 00	11 00	01 00	01 10	11 10	10 10	00 10	00 11	10 11	11 11	01 11	01 01	11 01	10 01	00 01
S=0	0	X	0	X	0	X	X	X	X	0	X	0	X	X	X	0
S=1	0	X	0	X	0	X	X	X	X	0	X	0	X	X	X	0
X=0	X	0	0	1	0	X	0	0	1	0	X	0	X	X	1	0
X=1	X	1	0	0	0	X	1	0	0	0	X	0	X	X	0	0
L=0	X	X	0	X	0	X	X	X	X	0	X	0	1	X	X	X
L=1	X	X	0	X	0	X	X	X	X	0	X	0	1	X	X	X

Appendix C:

Q1*

Q3Q2Q1Q0 / SXL	00 00	10 00	11 00	01 00	01 10	11 10	10 10	00 10	00 11	10 11	11 11	01 11	01 01	11 01	10 01	00 01
S=0	0	X	0	X	0	X	X	X	X	0	X	0	X	X	X	0
S=1	0	X	0	X	0	X	X	X	X	0	X	0	X	X	X	0
X=0	X	0	0	0	0	X	1	0	0	0	X	0	X	X	0	1
X=1	X	0	0	0	0	X	0	1	0	0	X	0	X	X	1	0
L=0	X	X	0	X	0	X	X	X	X	0	X	0	1	X	X	X
L=1	X	X	0	X	0	X	X	X	X	0	X	0	1	X	X	X

Appendix D:

Q0*

Q3Q2Q1Q0 / SXL	00 00	10 00	11 00	01 00	01 10	11 10	10 10	00 10	00 11	10 11	11 11	01 11	01 01	11 01	10 01	00 01
S=0	0	X	0	X	1	X	X	X	X	1	X	1	X	X	X	0
S=1	1	X	0	X	1	X	X	X	X	1	X	1	X	X	X	1
X=0	X	1	0	1	1	X	1	0	0	1	X	1	X	X	0	0
X=1	X	0	0	0	1	X	0	1	0	1	X	1	X	X	0	0
L=0	X	X	0	X	1	X	X	X	X	1	X	1	1	X	X	X
L=1	X	X	0	X	1	X	X	X	X	1	X	1	0	X	X	X

Appendix E:

Z

Q1Q0 \ Q3Q2	00	01	11	10
00	1	1	1	1
01	1	1	X	1
11	1	1	X	1
10	1	1	X	1

2.2 SystemVerilog Code

```
//Name: Seongjae Shin
//Assignment: Final Project
//Purpose: Create a security system
//Date: May 6, 2021
//Module: proj_main
//Date of Last Update: May 6, 2021

// the code is 0100
module proj_main(S,X,U,Z);
 // S is swtich
 // X is user input (password entered)
 // U is update function for program. This always changes 0 to 1 or 1 to 0 to update program
 input S,X,U;
 // Z is 3 bits vertorization. Z[2] = Switch ON/OFF, Z[1] = Lock/Unlcok, Z[0] = Trigger/Not Trigger
 output [2:0]Z;

 // PS is present state
 reg [3:0] PS;
 // L is lock
 // if L = 0, it is unlocked, if L=1, it is locked
 reg L;

 // parameters for lock
 parameter unlock=1'b0, lock=1'b1;
 // parameters for present state
 parameter
Ab=4'b0000,Bb=4'b0001,Cb=4'b0010,Db=4'b0011,Eb=4'b0100,Fb=4'b0101,Gb=4'b0110,Hb=4'b0111,Ib=
4'b1000,Jb=4'b1001,Kb=4'b1010,Lb=4'b1011,Mb=4'b1100;
 // initially it is OFF and unlock
 initial PS=4'b0000;
 initial L=1'b0;
 // update when U is changed
 always@(U)
  // checking present state
  case(PS)
```

```
    //OFF
    Ab:if(S==1) PS=Bb;
    //ON
    Bb:if(S==0) PS=Ab; else if(S==1 && X==0) PS=Cb; else if(S==1 && X==1) PS=Ib;
    //First
    Cb:if(X==0) PS=Ib; else if(X==1) PS=Db;
    //Second
    Db:if(X==0) PS=Eb; else if(X==1) PS=Ib;
    //Third
    Eb:if(X==0) PS=Fb; else if(X==1) PS=Ib;
    //Fourth
    Fb:if(L==0) PS=Hb; else if(L==1) PS=Gb;
    //Unlock
    Gb:begin
      L=unlock;
      PS=Bb;
    end
    //Lock
    Hb:begin
      L=lock;
      PS=Bb;
    end
    //Trigger
    Ib:if(X==0) PS=Jb; else if(X==1) PS=Mb;
    //T-First
    Jb:if(X==0) PS=Mb; else if(X==1) PS=Kb;
    //T-Second
    Kb:if(X==0) PS=Lb; else if(X==1) PS=Mb;
    //T-Third
    Lb:PS=Bb;
    //Error
    Mb:PS=Ib;
   endcase

  // After change the present state based on the input, find the output corresponding state
  out_help out1(L,PS,Z);

endmodule

// helping function of output
module out_help(A,B,Z);
  // A is L, which is lock/unlock
  input A;
  // B is present state
  input [3:0]B;
  // Z is output of this system
  output [2:0] Z;
```

```
 // F is reg for output
 reg [2:0] F;

 // 6 enable outputs
 parameter F1=3'b000,F2=3'b010,F3=3'b100,F4=3'b110,F5=3'b101,F6=3'b111;
 // parameters for states
 parameter
Ab=4'b0000,Bb=4'b0001,Cb=4'b0010,Db=4'b0011,Eb=4'b0100,Fb=4'b0101,Gb=4'b0110,Hb=4'b0111,Ib=
4'b1000,Jb=4'b1001,Kb=4'b1010,Lb=4'b1011,Mb=4'b1100;
 // update output when state is changed
 always@(B)
  // check present state
  case(B)
    //OFF
    Ab:if(A==0) F=F1; else if(A==1) F=F2;
    //ON
    Bb:if(A==0) F=F3; else if(A==1) F=F4;
    //First
    Cb:if(A==0) F=F3; else if(A==1) F=F4;
    //Second
    Db:if(A==0) F=F3; else if(A==1) F=F4;
    //Third
    Eb:if(A==0) F=F3; else if(A==1) F=F4;
    //Fourth
    Fb:if(A==0) F=F3; else if(A==1) F=F4;
    //Unlock
    Gb:F=F3;
    //Lock
    Hb:F=F4;
    //Trigge
    Ib:if(A==0) F=F5; else if(A==1) F=F6;
    //T-First
    Jb:if(A==0) F=F5; else if(A==1) F=F6;
    //T-Second
    Kb:if(A==0) F=F5; else if(A==1) F=F6;
    //T-Third
    Lb:if(A==0) F=F5; else if(A==1) F=F6;
    //Error
    Mb:if(A==0) F=F5; else if(A==1) F=F6;
  endcase
 //assign reg F value into Z
 assign Z = F;

endmodule
```

2.3 TESTBENCH CODE

```
//Name: Seongjae Shin
```

```
//Assignment: Final Project
//Purpose: Correct functions
//Date: May 6, 2021
//Module: Final_test
//Date of Last Update: Mar 6, 2021


module Final_test;
 // set testbench reg and wire
 reg S_tb,X_tb,U_tb;
 wire [2:0] Z_tb;

 // reacll hw11_main to test
 proj_main test1(.S(S_tb),.X(X_tb),.U(U_tb),.Z(Z_tb));

 // set initial values of each input
 initial
  begin
   // Initially OFF
   U_tb=0;
   S_tb=0;
  end

 // Change 0 to 1 or 1 to 0 per given time
 always #1 U_tb=!U_tb;
          always
   begin
    //OFF->ON
                  #1 S_tb=1;
    //ON->First
    #1 X_tb=0;
    //First->Second
    #1 X_tb=1;
    //Second->Third
    #1 X_tb=0;
    //Third->Fourth
    #1 X_tb=0;
    //Fourth->Lock
    #1 X_tb=0;
    //Lock->ON
    #1 X_tb=0;
    //ON->Trigger
    #1 X_tb=1;
    //Tigger->Error
    #1 X_tb=1;
    //Error->Trigger
    #1 X_tb=1;
```

```
      //Trigger->T-First
      #1 X_tb=0;
      //T-First->T-Second
      #1 X_tb=1;
      //T-Second->T-Third
      #1 X_tb=0;
      //T-third->ON
      #1 X_tb=0;
           //ON->First
      #1 X_tb=0;
      //First->Second
      #1 X_tb=1;
      //Second->Third
      #1 X_tb=0;
      //Third->Foruth
      #1 X_tb=0;
      //Fourth->Unlock
      #1 X_tb=0;
      //Unlock->ON
      #1 X_tb=0;
    end

  // Set table, names and values of columns
  initial
    begin
      $display("S\t\X\t\U\t\tZ");
      $monitor("%b\t%b\t%b\t%b",S_tb,X_tb,U_tb,Z_tb);
    end

  // set waveform
  initial
    begin
      $dumpfile("Final.vcd");
      $dumpvars;
    end

  // set total timeer
  initial
    #20 $finish;
endmodule
```