CSE241 Final Lab Documentation

Nathan Fox
Spring 2023

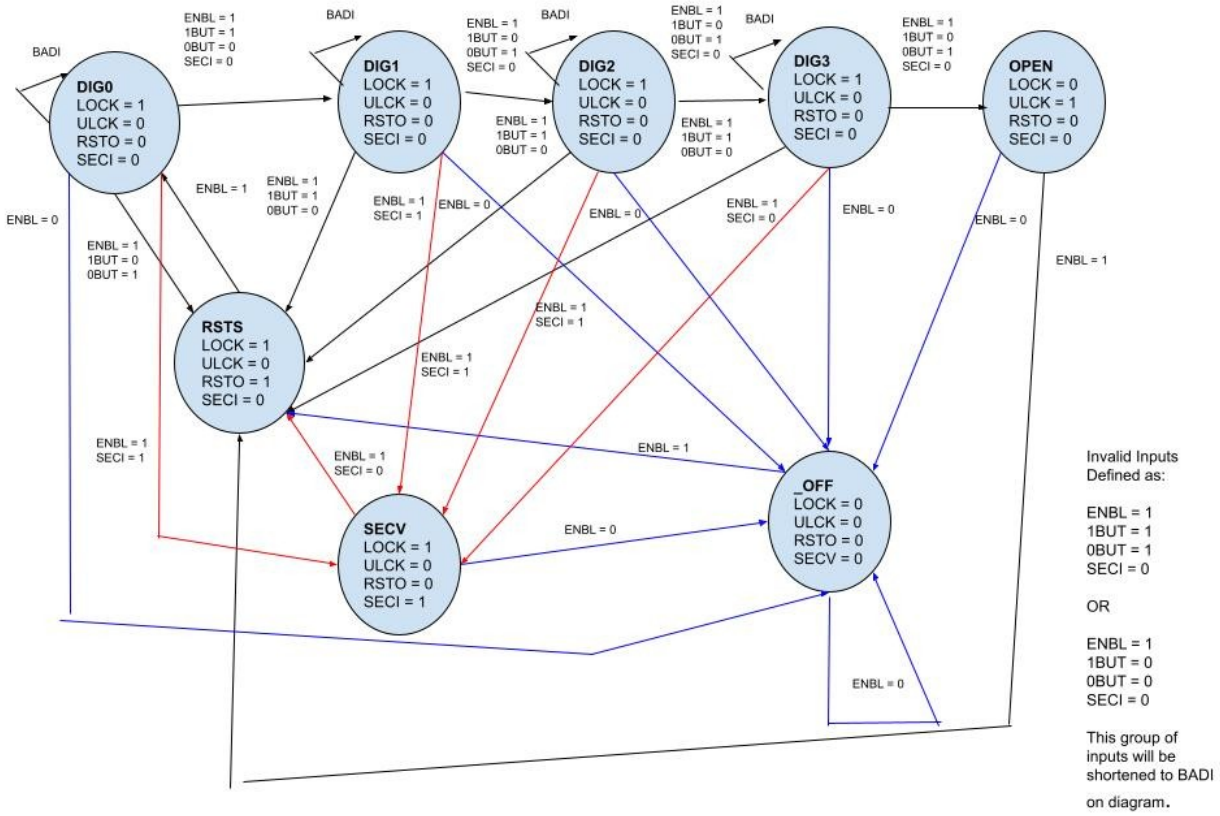
Table of Contents

1 System Description

For the final lab, the task was to design a security system using a finite state machine setup, as well as define the surrounding environment to the security system. As such, a keypad for a safe on a desk was created and is what will be discussed in this document. The combination to this safe is going to be 0b1000. The keypad has four inputs: OBUT, ZBUT, ENBL, and SECI. The keypad has two buttons: a button that enters a 0 (zbut), and a button that enters a one (obut). ENBL or enable (E) would be physically represented with plugging the system into the wall, and unplugging it would turn the system off. Security violation (seci) would be something such as moving the safe while it is active, or setting it down on a slant, since it needs to be perfectly perpendicular. For output, the system also has four LEDS: LOCK, ULCK, RSTO and SECV. SECV only triggers during a security violation. The system reads in the passcode one bit at a time, and whenever it reads in a wrong bit, it triggers the RSTO LED. ULCK and LOCK represent whether the safe is unlocked or locked, respectively.

2 System Specifications

As touched on in section 1, there are four inputs and four outputs. These inputs and outputs are controlled via a Moore-style state machine. The following is the state diagram the system was design using:



As shown, there are a total of 8 states. Four states for each bit of the passcode and four special states. These special states are the following:
Reset-State (RSTS): The state when a incorrect digit of the passcode has been passed in, triggering Reset-Out (RSTO). On clock update, assuming the system is enabled, it will transition back to Digit 0.

Security Violation: The emergency state when a security violation has been detected, using the Security Input (SECI). This could be anything from an attempted break in to setting the safe down at a slant. Regardless, the SECV state will trigger the eponymous output, SECV. Only when SECI is turned off will the system transfer into RSTS and resume normal reading. This will not occur when the safe is OPEN, because it is pointless to try to lock the system further if it is already open.

_OFF: The state that represents when everything is off. Since the system needs electricity to work, and enable (ENBL) has been defined to be plugging the system into the wall, any time ENBL is off the system immediately shuts down.

OPEN: The state which represents an open safe. The safe is open, and locking it is done mechanically after a clock update.

Transitioning through these states was found and implemented using kmaps to find transition equations, and output equations. These kmaps were derived from the state table and transition table. A total of 7 Kmaps were used. They will be shown below:

Next State – State Table										
	Inputs						Outputs			
	ENBL'	ENBL								
		SECI'				SECI				
Current State		1BUT= 0, 0BUT = 0	1BUT= 0, 0BUT = 1	1BUT= 1, 0BUT = 1	1BUT= 1, 0BUT = 0		LOCK	ULCK	RSTO	SECV
DIG0	_OFF	DIG0	RSTS	DIG0	DIG1	SECV	1	0	0	0
DIG1	_OFF	DIG1	DIG2	DIG1	RSTS	SECV	1	0	0	0
DIG2	_OFF	DIG3	DIG3	DIG3	RSTS	SECV	1	0	0	0
DIG3	_OFF	DIG3	OPEN	DIG3	RSTS	SECV	1	0	0	0
SECV	_OFF	RSTS	RSTS	RSTS	RSTS	SECV	1	0	0	1
RSTS	_OFF	DIG0	DIG0	DIG0	DIG0	SECV	1	0	1	0
_OFF	_OFF	RSTS	RSTS	RSTS	RSTS	RSTS	0	0	0	0
OPEN	_OFF	RSTS	RSTS	RSTS	RSTS	RSTS	0	1	0	0

Next State – Transition Table										
	Inputs						Outputs			
	ENBL = 0	ENBL = 1								
		SECI = 0				SECI = 1				
Current State		1BUT= 0, 0BUT = 0	1BUT= 0, 0BUT = 1	1BUT= 1, 0BUT = 1	1BUT= 1, 0BUT = 0		LOCK	ULCK	RSTO	SECV
000	111	000	101	000	001	100	1	0	0	0
001	111	001	011	001	101	100	1	0	0	0
011	111	011	010	011	101	100	1	0	0	0
010	111	010	110	010	101	100	1	0	0	0
100	111	101	101	101	101	100	1	0	0	1
101	111	000	000	000	000	100	1	0	1	0
111	111	101	101	101	101	101	0	0	0	0
110	111	101	101	101	101	101	0	1	0	0

		Second + Third				
	LOCK	00	01	11	10	BC
First bit	0	1	1	1	1	
	1	1	1	0	0	
	A					

LOCK = B' + A'

		Second + Third				
	ULCK	00	01	11	10	BC
First bit	0	0	0	0	0	
	1	0	0	0	1	
	A					

ULCK = ABC'

		Second + Third				
	RSTO	00	01	11	10	BC
First bit	0	0	0	0	0	
	1	0	1	0	0	
	A					

RSTO = AB'C

		Second + Third				
	SECV	00	01	11	10	BC
First bit	0	0	0	0	0	
	1	1	0	0	0	
	A					

SECV = AB'C'

First bit is SECI, Second is 1BUT, Third is 0BUT

		Bit 1 of Next State								
	ABC	Inputs								
	State	000	001	011	010	100	101	111	110	DEF
DIG0	000	0	1	0	0	1	1	1	1	
DIG1	001	0	0	0	1	1	1	1	1	
DIG2	011	0	0	0	1	1	1	1	1	
DIG3	010	0	1	0	1	1	1	1	1	
SECV	100	1	1	1	1	1	1	1	1	
RSTS	101	0	0	0	0	1	1	1	1	
_OFF	111	1	1	1	1	1	1	1	1	
OPEN	110	1	1	1	1	1	1	1	1	

BIT1 = D+AC'+AB+C'E'F+BEF'+A'CEF'

		Bit 2 of Next State								
	ABC	Inputs								
	State	000	001	011	010	100	101	111	110	DEF
DIG0	000	0	0	0	0	0	0	0	0	
DIG1	001	0	1	0	0	0	0	0	0	
DIG2	011	1	1	1	0	0	0	0	0	
DIG3	010	1	1	1	0	0	0	0	0	
SECV	100	0	0	0	0	0	0	0	0	
RSTS	101	0	0	0	0	0	0	0	0	
_OFF	111	0	0	0	0	0	0	0	0	
OPEN	110	0	0	0	0	0	0	0	0	

BIT2 = BE' + BF + CE'F

		Bit 3 of Next State								
	ABC	Inputs								
	State	000	001	011	010	100	101	111	110	DEF
DIG0	000	0	1	0	0	0	0	0	0	
DIG1	001	1	1	1	1	0	0	0	0	
DIG2	011	1	0	1	1	0	0	0	0	
DIG3	010	0	0	0	1	0	0	0	0	
SECV	100	1	1	1	1	0	0	0	0	
RSTS	101	0	0	0	0	0	0	0	0	
_OFF	111	1	1	1	1	0	0	0	0	
OPEN	110	1	1	1	1	0	0	0	0	

BIT3 = AC'D' + ABD' + A'CD'E' + A'CD'E + BD'EF' + A'B'D'E'F

3 System-Verilog Implementation

The system was partially implemented in system verilog using a d-flip-flop module, and assign statements. Partially because it is incomplete, due to time constraints. Originally the implementation was using a large case statement to manually set each output LED depending on the state, but upon compiler error after compiler error the decision was made to switch to assigns, as a case statement in verilog turned out to the extraordinarily unpleasant. Regardless, there are some quirks of this implementation, the biggest one being vcd was not able to be used to create a timing diagram. It is unknown why, possibly because of all of the continuous assignment statements instead of gate-level implementation or decoders, but a truth table was all the information that could be extracted from this implementation. Arrays were used to conveniently store the 3-bit state into one data structure, so all manipulation of state was done on this array.

As for testing the system, it is again incomplete. The goal was to go through most possible inputs and transitions, to validate that resetting worked, _OFF worked, and so on. Only validation of it reading in the correct sequence of bits and unlocking was gotten to. The following is a truth table of the last compiled example, and it is not correct. All inputs are shown as well as all outputs. The final two columns are displayed for debugging purposes and would not be visible to a normal user, as they represent the state of the system.

```
┌─nfox@YoRHa in repo: CSE241-FinalLab on Y master [!?] took 40ms
└─> iverilog -g2005-sv -o fsm.vvp fsm.sv fsm-tb.sv dflipflop.sv && ./fsm.vvp
obuttb zbuttb  seci   enbl      clk |  lock     ulck    secv    rsto    ostatetb  fftb
1       0       0       1       0  |  1        0       0       0       000       000
1       0       0       1       1  |  1        0       0       0       000       000
1       0       0       1       0  |  1        0       0       0       000       000
0       1       0       1       1  |  1        0       0       0       000       101
0       1       0       1       0  |  1        0       0       0       000       101
0       1       0       1       1  |  1        0       0       1       101       000
0       1       0       1       0  |  1        0       0       1       101       000
0       1       0       1       1  |  1        0       0       0       000       101
0       1       0       1       0  |  1        0       0       0       000       101
1       0       0       1       0  |  1        0       0       1       101       010
1       0       0       1       1  |  1        0       0       0       010       101
1       0       0       1       0  |  1        0       0       0       010       101
0       1       0       1       1  |  1        0       0       1       101       000
0       1       0       1       0  |  1        0       0       1       101       000
0       1       0       1       1  |  1        0       0       0       000       101
0       1       0       1       0  |  1        0       0       0       000       101
0       1       0       1       1  |  1        0       0       1       101       000
0       1       0       1       0  |  1        0       0       1       101       000
0       0       0       1       1  |  1        0       0       0       000       000
0       0       0       1       0  |  1        0       0       0       000       000
1       0       0       1       0  |  1        0       0       0       000       000
1       0       0       1       1  |  1        0       0       0       000       000
1       0       0       1       0  |  1        0       0       0       000       000
0       1       0       1       0  |  1        0       0       0       000       101
0       1       0       1       1  |  1        0       0       1       101       000
0       1       0       1       0  |  1        0       0       1       101       000
0       1       0       1       1  |  1        0       0       0       000       101
0       1       0       1       0  |  1        0       0       0       000       101
1       0       0       1       0  |  1        0       0       0       000       000
0       0       0       1       1  |  1        0       0       0       000       000
fsm-tb.sv:64: $finish called at 30 (1s)
0       0       0       1       0  |  1        0       0       0       000       000
```

4 Analysis

The results from the little testing that was implemented suggest that the transition equations are implemented incorrectly and need to be reviewed. Since the transition equations require review, the output equations cannot be trusted. Overall, a disaster of a project. This safe is not going to protect anything. Little other useful information can be extracted, since the veracity of *any* information is in doubt considering the probable incorrect calculation and implementation of the transition and output equations.

5 Recommendations

It needs to work, quite simply. Before any other recommendations can be suggested it needs to work correctly first.