ECE 273: Introduction to Digital Logic

Lab <Project 2> : <Security System>

By

<Hussein El-Souri>

Lab Instructor: < Zheming Liang >

Today

Winter 2017

Honor Code:

I have neither given nor received unauthorized assistance on this graded report.

X___Hussein El-Souri

Department of Electrical and Computer Engineering
College of Engineering and Computer Science
The University of Michigan-Dearborn

Objective

I wish to design a simple security system consisting of a *Card Reader* and a *Key Pad.* A person may open a door if he or she has a card containing the corresponding code, and enters an authorized key pad code for that card. A person is given **only two chances** to key in the correct code after which an alarm will go off. The alarm also goes off if the card has the wrong code on it. If the correct key pad code is entered, the corresponding door will be unlocked when the card is inserted.

NOTE: The three bits C, D, E are set first. Then the card is inserted and the keypad code is checked. If the code is wrong, the card can be removed and reinserted after a new keypad code is set

Theory

K maps were used to design the counter needed for this lab. Otherwise other components of this project required a more conceptual approach than a theoretical approach.

Equipment Used

- Software: Xilinx, PlanAhead
- Hardware: BASYS2 FPGA Board

Procedure

I started the project by first designing a 2-bit counter using two T flip-flops. The counter should count to 2 according to the following state table:

A	B	A+	B+
0	0	0	1
0	1	1	0
1	0	x	x
1	1	x	x

Table 1: 2-bit Counter State Table

From this I can generate the excitation table for this design taking into consideration the excitation table of the T Flip Flop:

Q	Q+	T
0	0	0
0	1	1
1	0	1
1	1	0

Table 2: Excitation Table of T F/F

A	B	T1	T2
0	0	0	1
0	1	1	1
1	0	x	x
1	1	x	x

Table 3: 2-bit Counter Excitation Table

From Table 3 I can generate the k-maps for T1 And T2 such that T2 is most significant bit:

A B	0	1
0		X
1	1	X

Table 4: T1 K-map

A B	0	1
0	1	X
1	1	X

Table 5: T2 k-map

T1= B   and T2= 1. Here B is the output for T2 F/F and that’s how the two F/Fs are cascaded to create a counter the output from T2 will be an input for T1. And so I can draw the schematic for the counter.

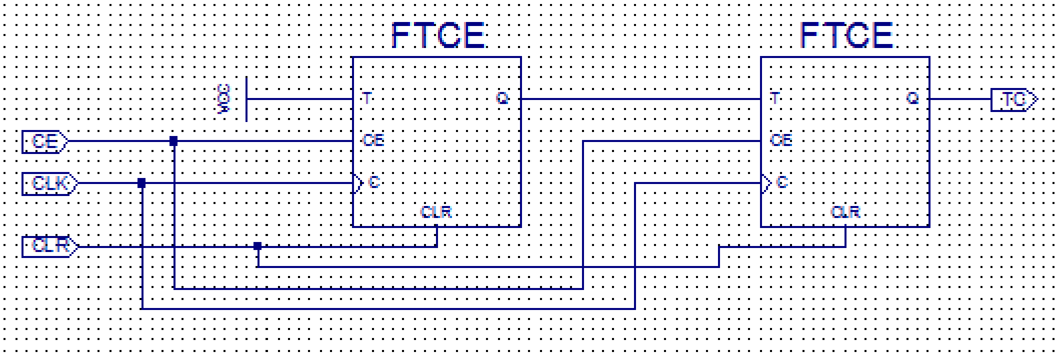

Figure 1: 2-bit counter Schematic

From this schematic, I can create a symbol to be used in our main design:

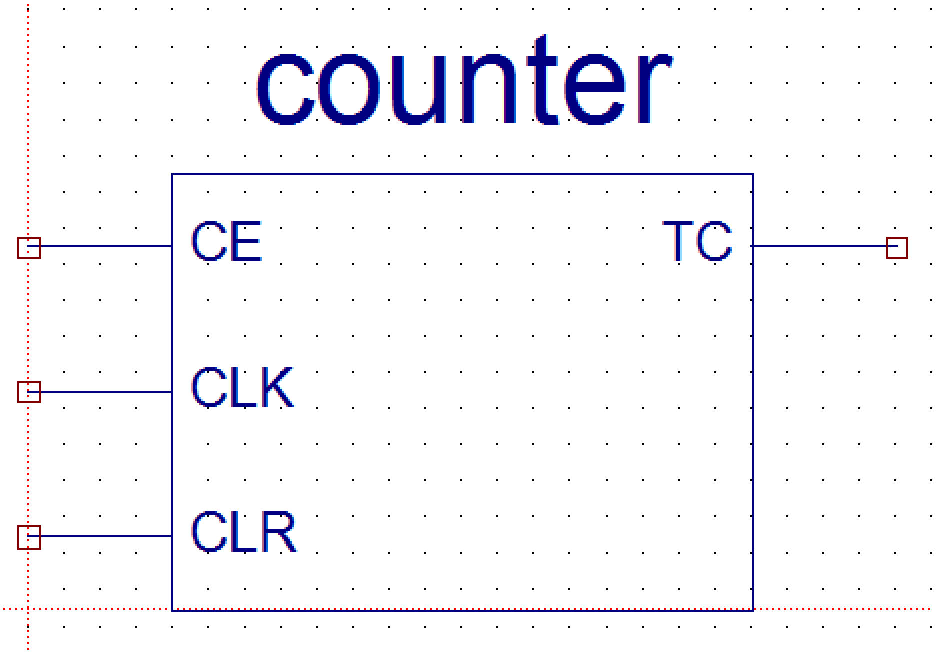

Figure 2: 2-bit counter Symbol

Now I can use a COMP4 which is a component that compares 2 4 bit digits. If they are the same it returns 1 and if they are different it returns 0. I use that to compare the input CDE to the preset password.

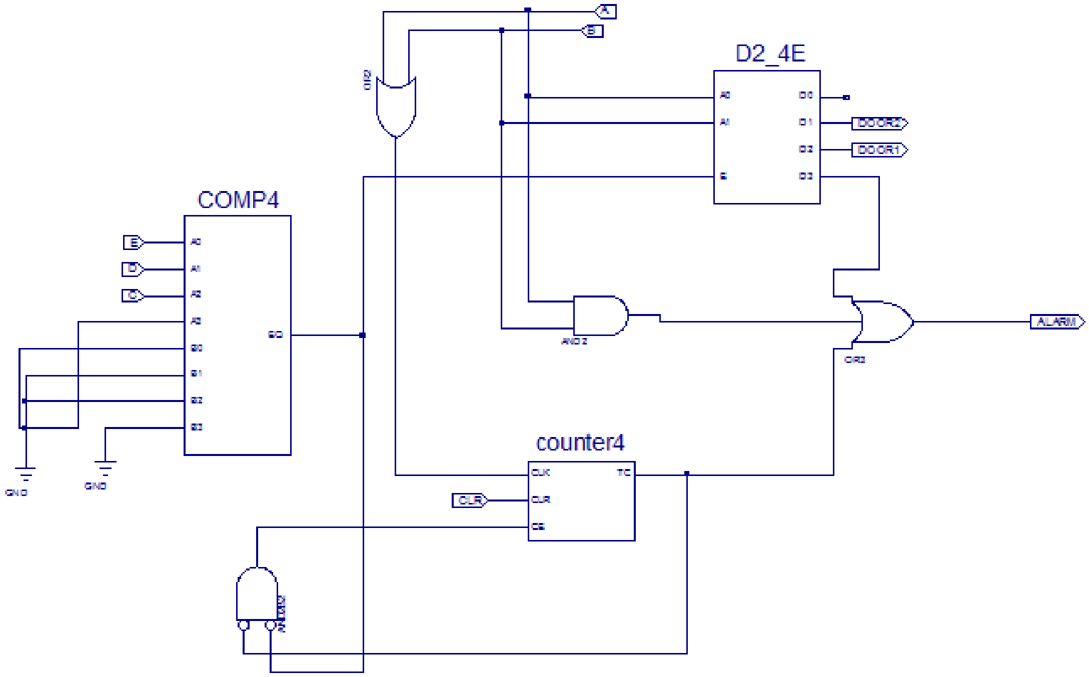

Figure 3: Main Schematic

The enable line on the counter is a bubbles 2-input and between the result of the COMP4 and previous value of counter. This means the counter is enabled only when there is a wrong code from the COMP4(output of zero) and the previous result of the counter is 0(which is initially true). The clock of the counter is A OR B this means the clock is when A or B are 1. The clear for the Counter is an input I control.

I also use a decoder to generate all minterms between A and B. Therefore I use a 2 by 4 decoder with inputs A and B. The enable line for the decoder is the result of the comparison. Meaning the decoder will be enables only when the result of the comparison is 1.

The alarm output is an OR between the fourth minterm of the decoder, the counter, and the result of the AND between A and B. This means alarm will go off when the decoder is enabled and minterm 3 is 1(A and B are both 1), or when the counter reaches 2, or when A and B are 1 regardless if decoder is active.

Door 1 is the minterm 1 and Door 2 is minterm 2. Door 1 is 1 only when the decoder is active and A is 1 and B is 0. Door 2 is 1 only when the decoder is active and A is 0 and B is 1.

Now I can write the test bench that will allows us to generate the waveforms:

```
-- *** Test Bench - User Defined Section ***
        tb : PROCESS --password:101
   BEGIN
             CLR<= '1';
        A <= '0';
             B <= '0';
             C <='0';
             D <= '0';
             E<= '0'; --NOTHING
             WAIT FOR 20ns;
             A <= '0';
             B <= '1';
             C <='1';
             D <= '0';
             E<= '1'; --DOOR 1 OPEN
             WAIT FOR 50ns;
             A <= '1';
             B <= '0';
             C <='1';
             D <= '0';
             E<= '1'; --DOOR 2 OPEN
             WAIT FOR 50ns;
             A <= '1';
             B <= '1'; --ALARM
             WAIT FOR 50ns;
             -----------------------------
             CLR <= '1';
             A <= '0';
             B <= '0';
             C <='0';
             D <= '0';
             E<= '0';           --RESET
             WAIT FOR 50ns;
             CLR <= '0';
             A <= '0';
             B <= '0';
             C <='0';
             D <= '0';
             E<= '0';
             WAIT FOR 50ns;
             A <= '0';
             B <= '1';
             C <='1';
             D <= '0';
             E<= '0';--WRONG PASS
             WAIT FOR 50ns;
             A <= '0';
             B <= '1';
             C <='0';
             D <= '1';
             E<= '0'; --WRONG PASS--> ALARM
             WAIT FOR 50ns;
             A <= '0';
             B <= '1';
             C <='1';
             D <= '0';
             E<= '0';--WRONG PASS
             WAIT FOR 50ns;
   END PROCESS;
-- *** End Test Bench - User Defined Section ***
```

This test bench allows us to generate the following waveforms:

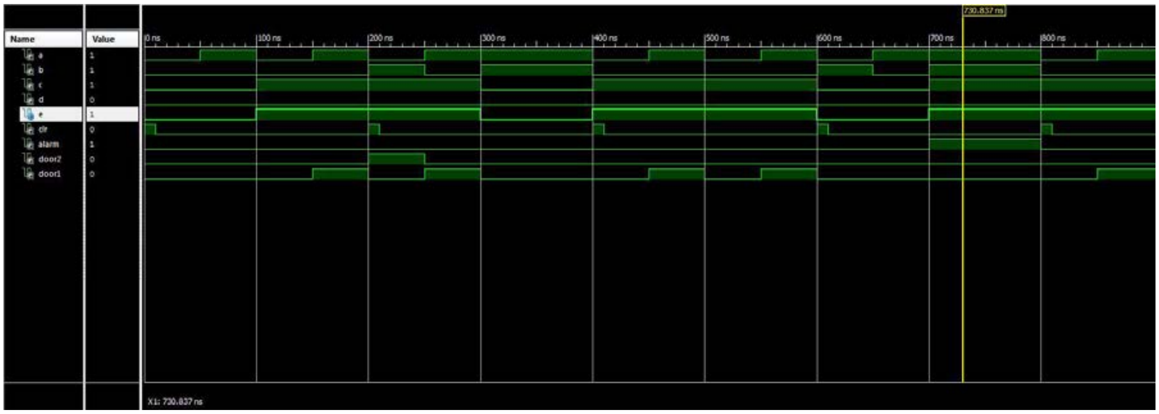

Figure 4: Waveforms

The waveforms demonstrate that let’s say at 730ns when A and B are 1 even with a wrong password (101) the alarm is on. At 180ns and A is 1 and B is 0 door 1 is open (is 1) when the password is right. And same for door 2 at 220ns.

Now to implement this on the board using the following user constraints:

```
# PlanAhead Generated physical constraints

NET "CLR" LOC = N3;
NET "E" LOC = E2;
NET "D" LOC = F3;
NET "C" LOC = G3;
NET "A" LOC = K3;
NET "B" LOC = L3;
NET "DOOR1" LOC = G1;
NET "DOOR2" LOC = P4;
NET "ALARM" LOC = N4;
```

Results

The FPGA board shows the same result as expected from the waveforms. The doors opened only when there was a correct code and when either A or B were 1. If A and B were Zero nothing happened regardless of the code because there was “no card inserted”. Finally, when A and B are both 1 alarm went off.

Conclusion

This project helped us understand better how Security systems works in general especially ATMS. It raised thoughts about how modern ATMs actually work and the complexity behind machines that is taken for granted.