CE100 Lab Report 5

Multiplication Game

Student Name: Armando Silva

Tutor: Dawn Hustig-Schultz

Lab Sec: TTH 2:00pm-4:00pm

Date: February 19th, 2016

Description:

The purpose of this lab was to get familiar with the implementation of state machines as well as using counters to display the score, increment of score or decrement of score. Using the logic learned in class, I was expected to build a state machine with Verilog, use a comparator, random number generator, a ring counter to control the 7-segment displays, a selector to choose one out of four 4-bit buses to display, and logic to display the cathodes when properly needed. Once we have all these components, I assembled state machine that helps control the state of my "multiplication" game that counts up each time Push Button 2 is pressed when the switches correspond to the binary number of the hex number displayed, and counts up number of wrongs if the user got the binary number wrong.

Part A: Random Number Generator

Essentially this symbol is to generate the two random 4-bit values that are to be multiplied. I used what was given to me from a figure on the class website, where 8 FDE flip flops were involved and an xor of input R0, R5, R6, R7.

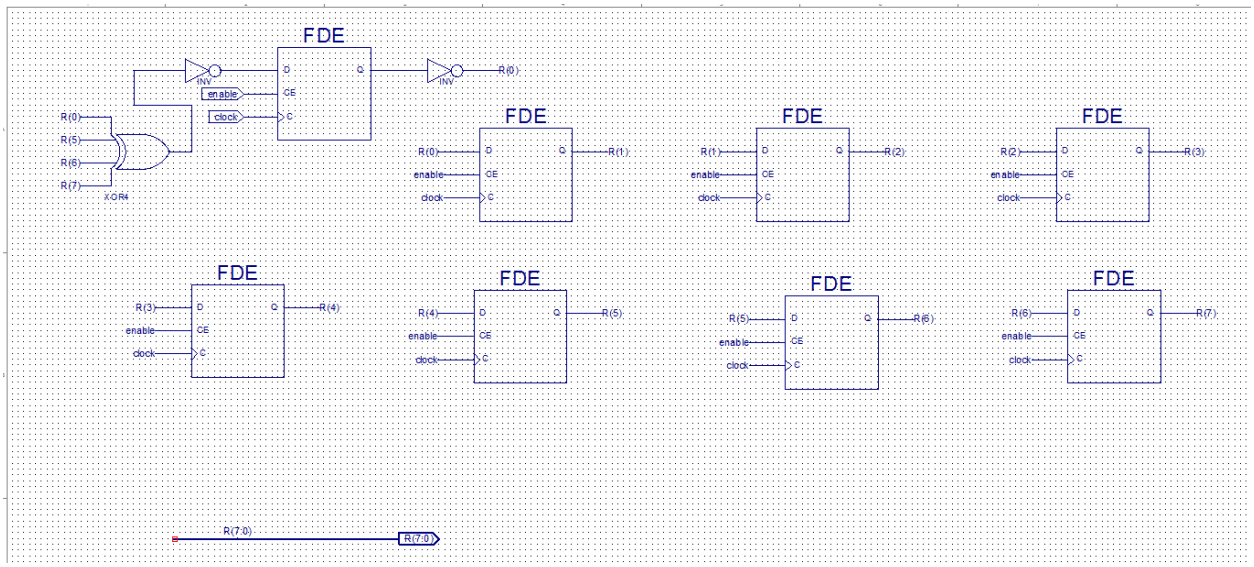


Figure 1: The LFSR (Random Number Generator)

Part B: Counters

The second part of the lab was to create counters that hold values for our user to know their stat, both correct and incorrect. The follow schematic on Figure 2 was how I went about using my counters. I used three CB4Res, where the first was to count time using quarter seconds, at the end of that 4-bit, also known as TC, was when I knew four seconds had gone by to display or hold to show the values. The second CB4RE was my right increment, the third one works exactly the same, however it is for the number of times wrong. I used RightInc and WrongInc to be when I count into these clocks. The CE are high in different states of the state machine, to be discussed further in my lab.

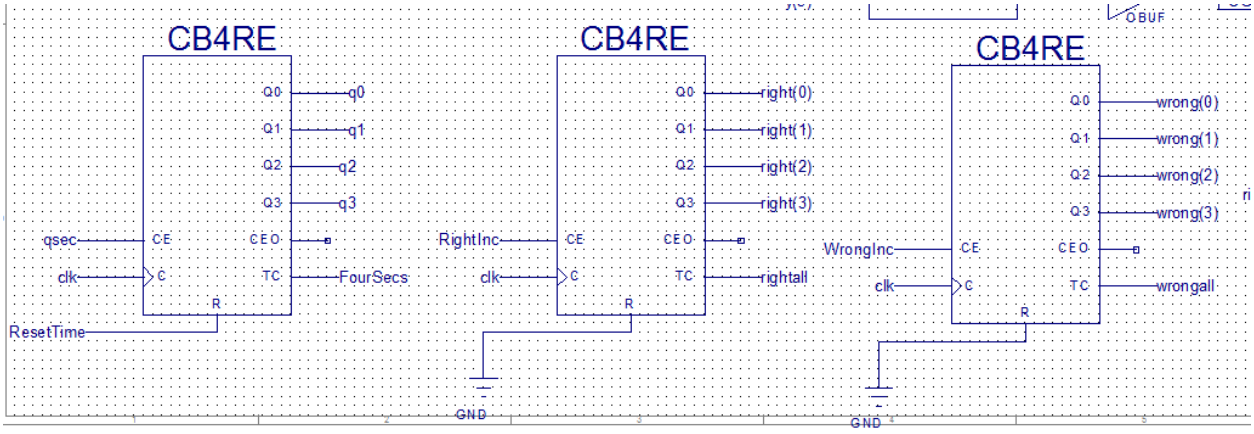


Figure 2: The counters (Qsec, Number Right, Number Wrong)

Part C: Multiplier

The third part of this lab is to use my 4-bit multiplier from my previous lab called stage. I use this to multiply the two random 4-bit numbers and then use COMP8 to compare the 8-bit with the switches to determine whether or not the user is matching the switches with the multiplied number.

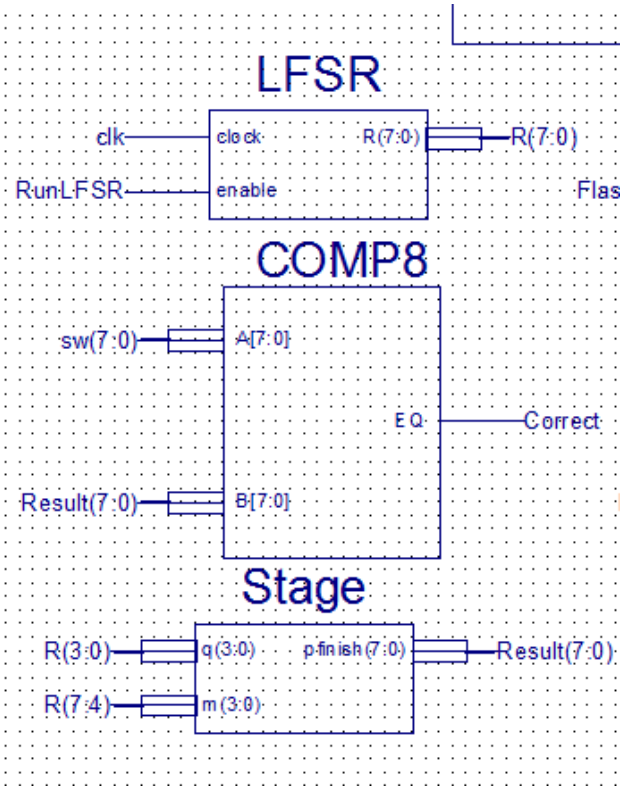


Figure 3: Stage (My multiplier from lab 4)

Part D: Display

The next part of the lab was to display all four digits of the 7 segment display, this is reusing my ring counter and selector. The only difference from the last lab was that I sent in different bits to the selector; where first 4 were from my Number Right counter, second was the first 4-bit number, third was the Number wrong counter, and lastly was the second 4-bit number.

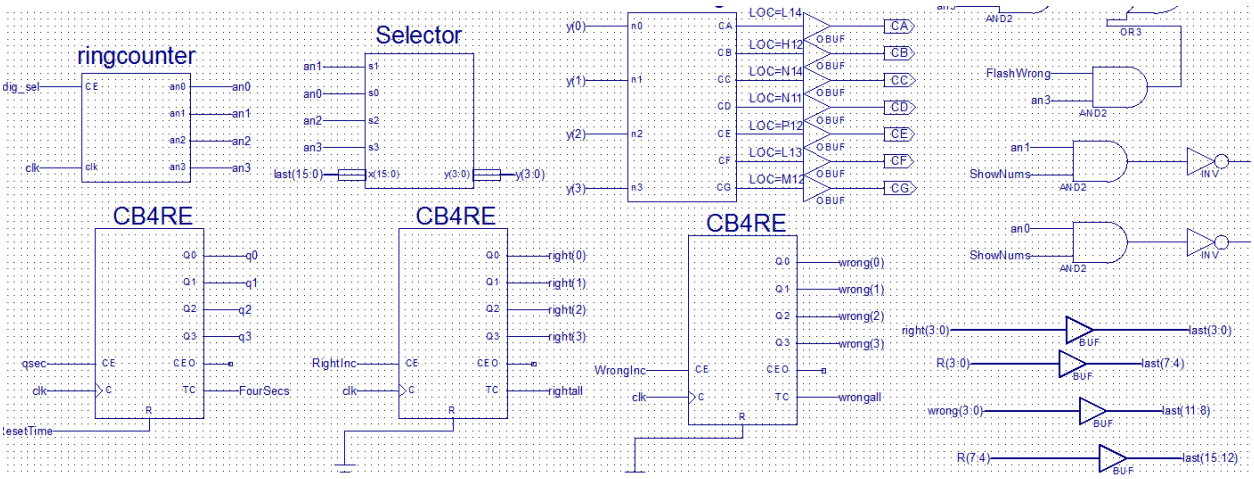


Figure 4: Ring Counter, Selector, 7-hex, and bottom left are the 15 bits to selector input

Part E: State Machine

As stated, “the heart”, of my design came from my state machine. Here I built the components to work together where I take in four inputs and output eight outputs that were used in my other symbols. I had 5 states where, in my lab report and looking at my Verilog code, one can see the state automaton that I created for the functionality to work.

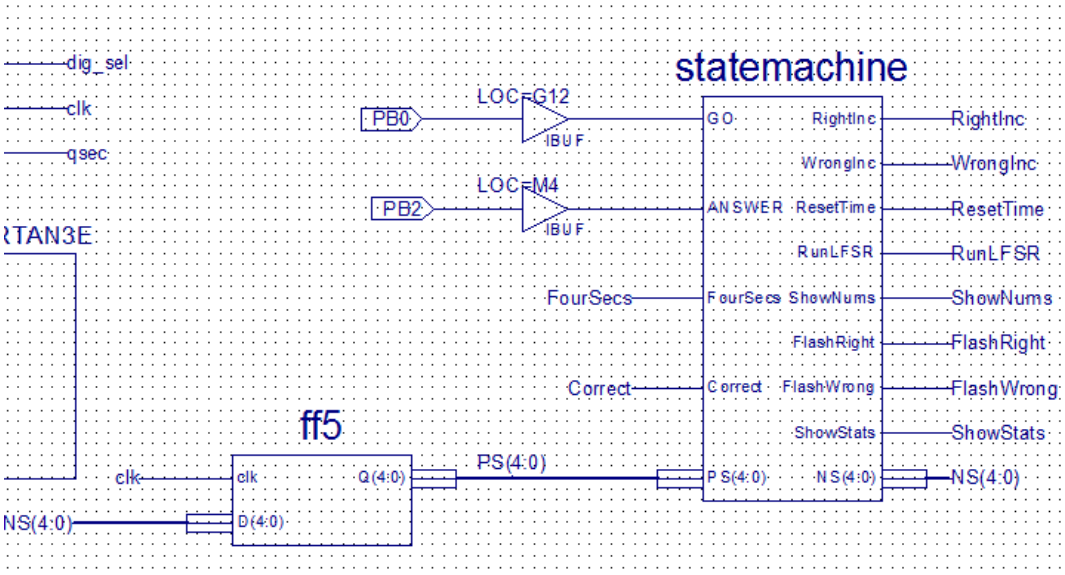


Figure 5: The state machine with the ff5

```
21  module statemachine(
22      input GO,
23      input ANSWER,
24      input FourSecs,
25      input Correct,
26      output RightInc,
27      output WrongInc,
28      output ResetTime,
29      output RunLFSR,
30      output ShowNums,
31      output FlashRight,
32      output FlashWrong,
33      output ShowStats,
34      input [4:0] PS,
35      output [4:0] NS
36      );
37     wire idle, start, yescorrect, notcorrect, final;
38     wire next_idle, next_start, next_yescorrect, next_notcorrect, next_final;
39
40     assign idle = PS[0];
41     assign start = PS[1];
42     assign yescorrect = PS[2];
43     assign notcorrect = PS[3];
44     assign final = PS[4];
45
46     assign NS[0] = next_idle;
47     assign NS[1] = next_start;
48     assign NS[2] = next_yescorrect;
49     assign NS[3] = next_notcorrect;
50     assign NS[4] = next_final;
51
52     assign next_idle = idle&~GO | final&FourSecs;
53     assign next_start = idle&GO | start&~ANSWER;
54     assign next_yescorrect = start&ANSWER&Correct | yescorrect&~FourSecs;
55     assign next_notcorrect = start&ANSWER&~Correct | notcorrect&~FourSecs;
56     assign next_final = final&~FourSecs | notcorrect&FourSecs | yescorrect&FourSecs;
57
58     assign RightInc = yescorrect&FourSecs;
59     assign WrongInc = notcorrect&FourSecs;
60     assign ResetTime = start&ANSWER&~Correct | start&ANSWER&Correct | yescorrect&FourSecs | notcorrect&FourSecs;
61     assign RunLFSR = idle&~GO;
62     assign ShowNums = idle&GO | start&~ANSWER;
63     assign FlashWrong = notcorrect&~FourSecs;
64     assign FlashRight = yescorrect&~FourSecs;
65     assign ShowStats = start&ANSWER&Correct | start&ANSWER&~Correct | final&~FourSecs;
66
```

Figure 6: State Machine created in Verilog

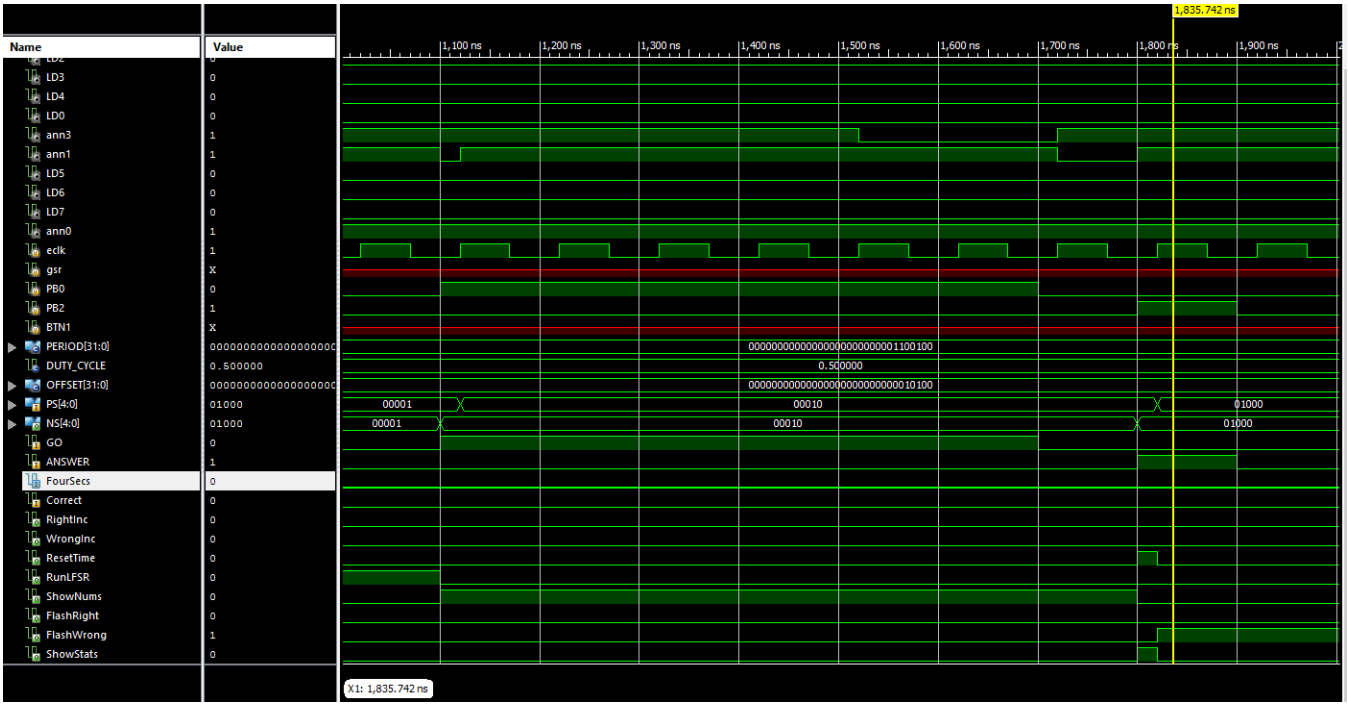


Figure 7: The simulation of my Top Level design

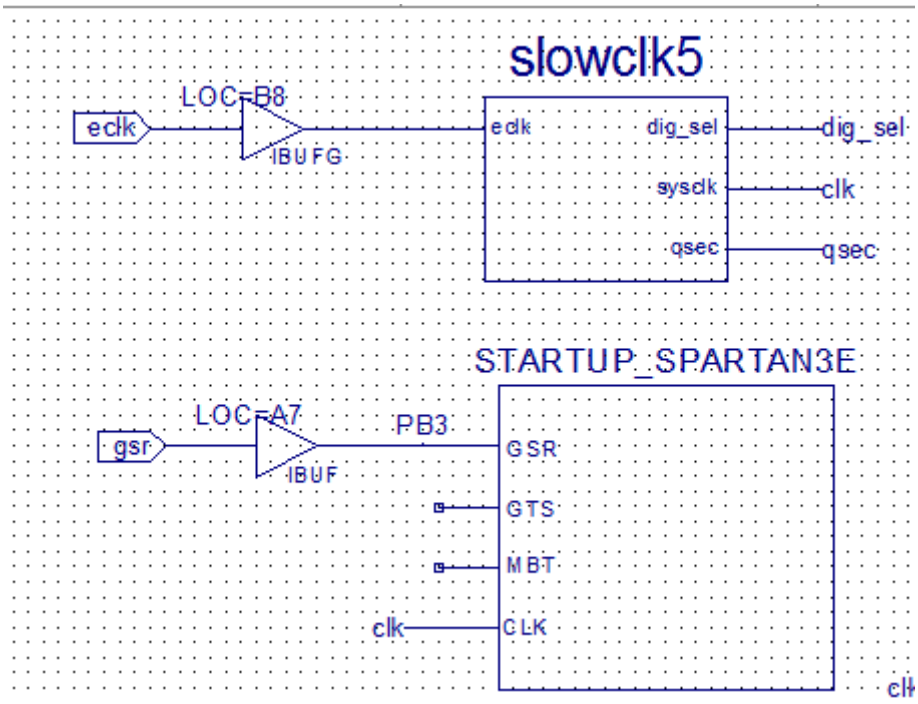


Figure 8: The clk that was provided to us by Martine

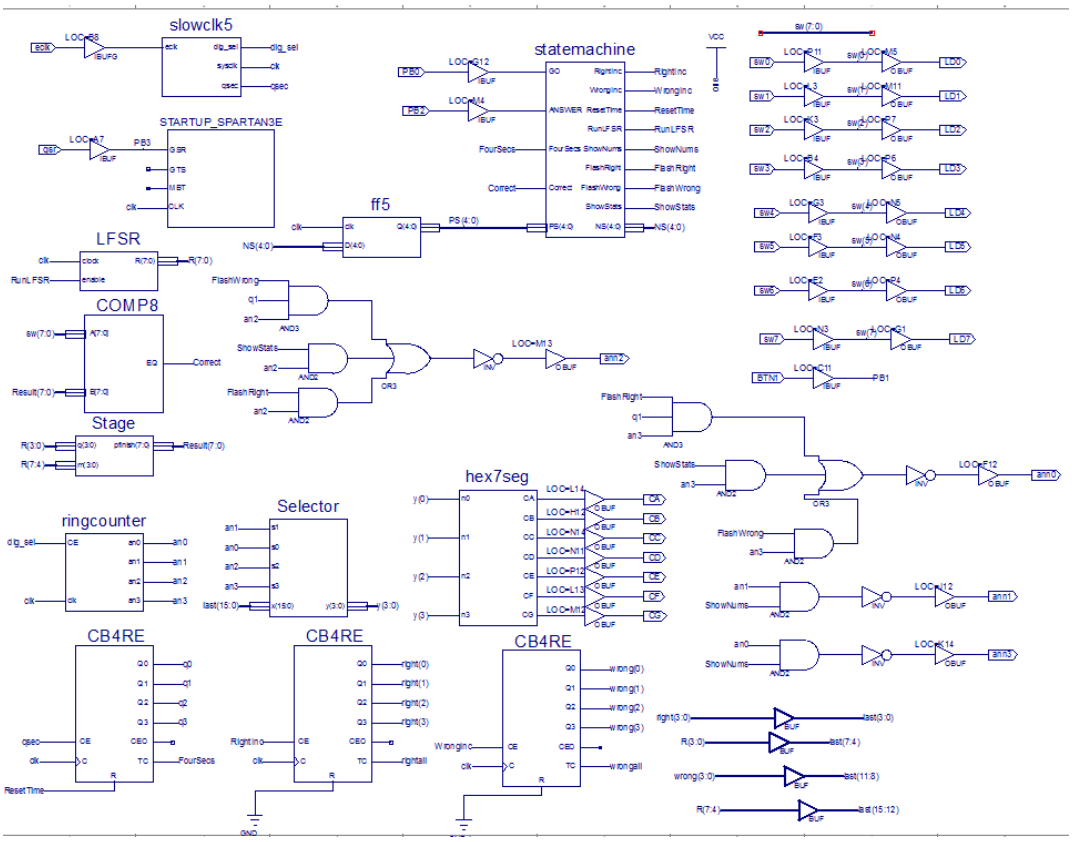


Figure 9: My Top Schematic

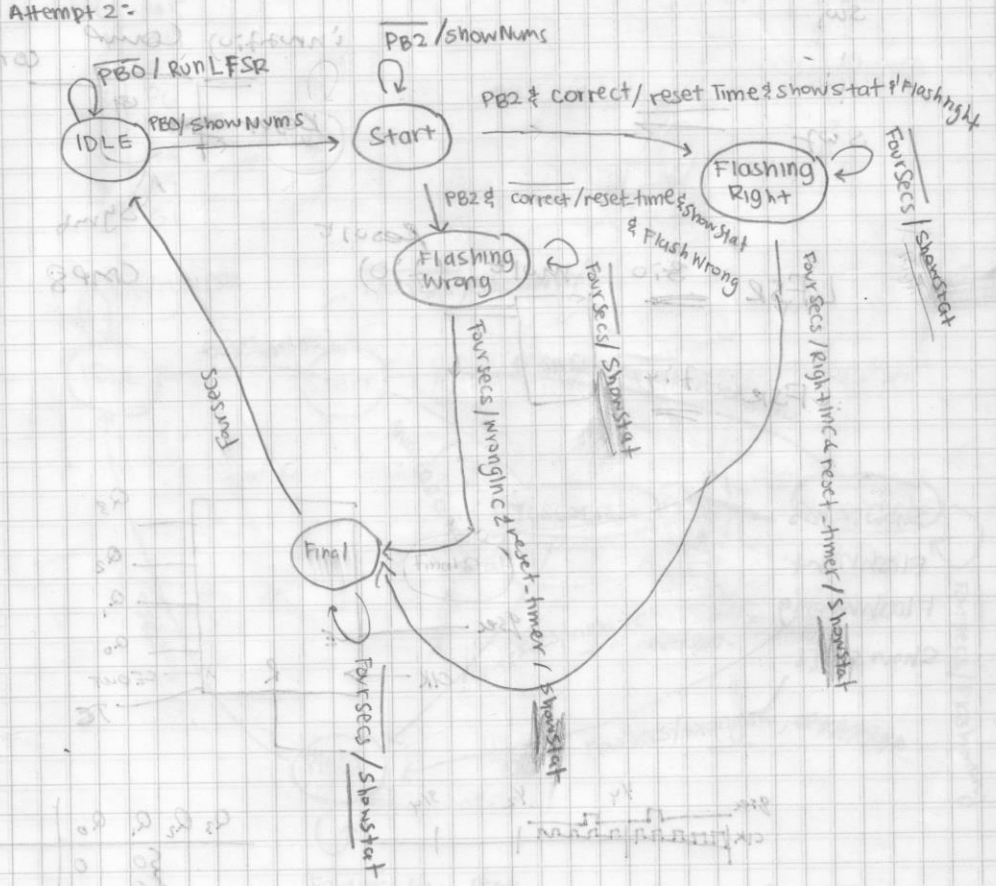
Conclusion:

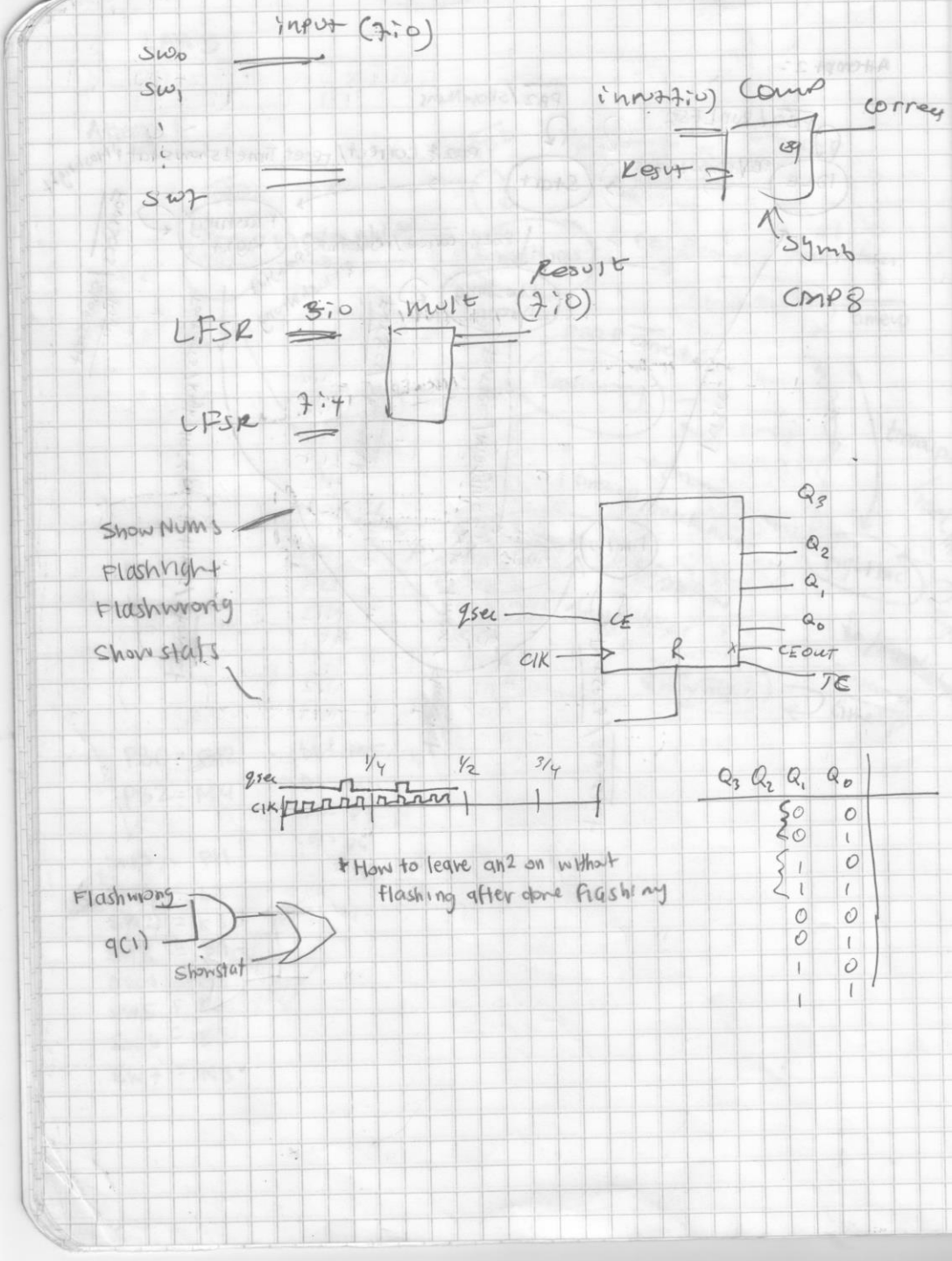
Ultimately this lab showed me how to comprehend state machines. With this I was able to process the logic behind states and transitions, along with counters and clocks.

Notebook:

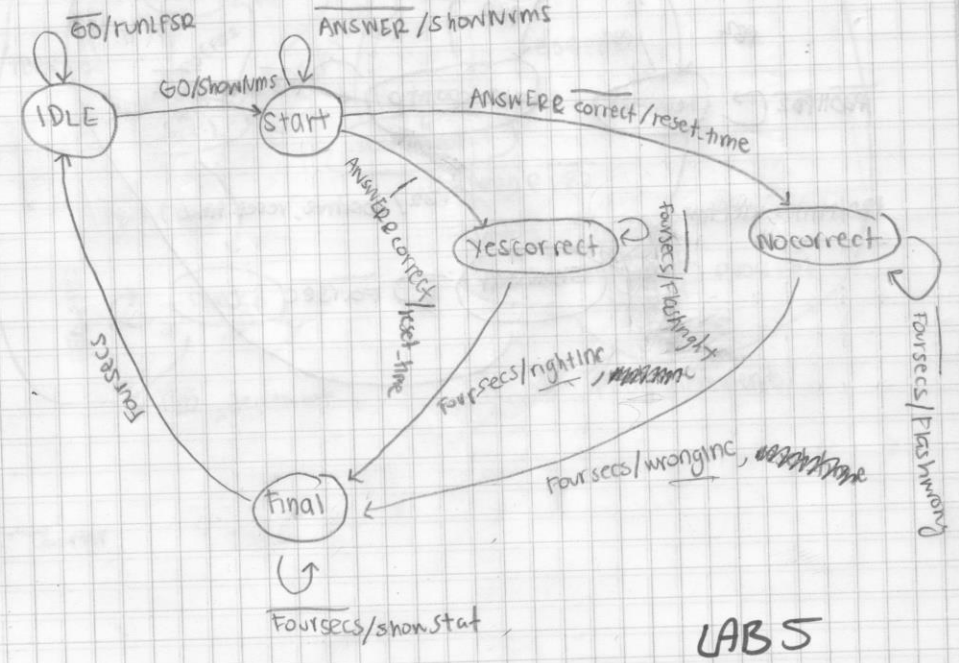

LAB 5
Attempt 1:
PB0
IDLE
PB0/RunLFSR
PB2
ShowNums
PB2 & correct /rese
FlashRight
PB2 & ~correct/reset_timer
Flashwrong
timeup
timeup/reset_timer
WrongInc
timeup/reset_time
/reset_timer
ShowStat
1 CE
qsec
reset_timer
PB0 = G12
PB2 = M4
SW0 = P11
SW1 = L3
SW2 = K3
SW3 = B4
SW4 = G3
SW5 = F3
SW6 = E2
SW7 = N3
LD0 = M5
LD1 = M11
LD2 = P7
LD3 = P6

Attempt 2:-


PB2 / showNums
PB0 / RunLFSR
PB2 & correct / reset Time & showstat & Flashright
IDLE
PB0 / showNums
Start
Flashing Right
FourSecs / showstat
PB2 & correct / reset time & showstat & Flash Wrong
Flashing Wrong
FourSecs / showstat
FourSecs
FourSecs / wrongInc & reset-timer / showstat
FourSecs / RightInc & reset-timer / showstat
Final
FourSecs / showstat


input (7:0)
sw0
sw1
sw7
input(7:0)
comp
correct
Result
eq
sym
LFSR
3:0
mult
Result
(7:0)
LFSR
7:4
CMP8
ShowNums
Flashright
Flashwrong
Showstats
Q3
Q2
Q1
Q0
qsec
CE
CLK
R
CEOUT
TC
qsec
CLK
1/4
1/2
3/4
Q3 Q2 Q1 Q0
0 0
0 1
1 0
1 1
0 0
0 1
1 0
1 1
*How to leave an2 on w/that
flashing after done flashing
Flashwrong
q(1)
Showstat

LD0	M5
1	M11
2	P7
3	P6
4	N5
5	N4
6	P4
7	G1



LAB 5
Rebecca
11:11
11 FEB 2016
1095