ECE 270 Lab Verification / Evaluation Form

Experiment 11

Evaluation:

IMPORTANT! You must complete this experiment during your scheduled lab period. All work for this experiment must be demonstrated to and verified by your lab instructor *before the end* of your scheduled lab period.

STEP	DESCRIPTION	MAX	SCORE
Pre-lab 1	Block diagram of circuit detailing interconnections	4	
Step 1	Scrolling message display modules ported from Lab 10	3	
Step 2	Binary up-counter module with integrated state decoder	4	
Step 3	Linear feedback shift register module	3	
Step 4	Combination match detector module	2	
Step 5	Sequence recognizer state machine module	6	
Step 6	Thought questions	3	
	TOTAL	25	

Signature of Evaluator: _____

Academic Honesty Statement:

"In signing this statement, I hereby certify that the work on this experiment is my own and that I have not copied the work of any other student (past or present) while completing this experiment. I understand that if I fail to honor this agreement, I will receive a score of ZERO for this experiment and be subject to possible disciplinary action."

Last Name (Printed): _____ **Lab Div:** _____ **Date:** _____

E-mail: __ __ __ __ __ __ __ __ **@**purdue.edu **Signature:** _____

Digital Combinational Lock with Pseudo-Random Combination

Instructional Objectives:

- To practice creating a sequence recognizer

Pre-lab Preparation:

- Read this document in its entirety
- Review the referenced Module 3 lecture material
- Read pp. 736-740 (4th Ed.) or pp. 574-577 (5th Ed.) in the course text

Experiment Description:

In this experiment you will create a digital combinational lock (DCL) that generates a pseudo-random 8-bit binary combination and "unlocks" when the selected combination is manually entered. A linear feedback shift register (LFSR) will be used to generate the pseudo-random combination, which will be displayed on the top row of red LEDs when DIP[5]=1 (and "hidden" when DIP[5]=0). The randomly-generated combination will keep changing as long as DIP[4]=1; it will "freeze" when DIP[4]=0. The left pushbutton (S2BC) will be used to *asynchronously **set*** the flip-flops that comprise the LFSR portion of the circuit.

When the DCL is in the initial ("locked") state, the scrolling string "**SECUrE**" will be displayed on the four 7-segment displays. The left pushbutton (S2BC) will be used to *asynchronously reset* the flip flops that comprise the sequence recognizer portion of the circuit to enter this initial state. The randomly-generated binary combination will then be entered into the sequence recognizer portion of the circuit using DIP[7]; the right pushbutton (S1BC) will be used to "clock in" the selected data bit entered on DIP[7]. The second row of LEDs (MIDRED) will be used to display a ring-like counter that advances as each digit of the combination is entered, serving as a "pointer" to the selected bit of the pseudo-random combination that is to be "matched." As the combination is being entered, the 7-segment displays will go blank and the (jumbo) yellow LED will be illuminated (to signify that "entering combination" is in process).

Once the 8-bit combination has been successfully entered, the scrolling string "**`OPEn`**" will be displayed on the four 7-segment LEDs and the (jumbo) green LED will be illuminated. Once unlocked, the sequence recognizer will ignore any further data entry from the state machine clocking pushbutton (S1BC) until the synchronous "relock" input DIP[6] is asserted, at which time clocking in either a "0" or "1" data bit (entered via DIP[7]) will return the state machine to its initial locked condition.

If a "mistake" is made at any point while the combination is being entered, however, the lock should enter the "alarm" state. The alarm condition is indicated by **flashing** the jumbo red LED at a 2.5 Hz rate and continuously scrolling the string "**`Error`**" across the four 7-segment LEDs.

Pre-lab Step (1):

Referring to the `lab11_top_template.v` file provided for this experiment on the course website, sketch a **block diagram** of the entire circuit on the page that follows, showing the interconnections among the various modules as well as all the inputs (DIP switches and pushbuttons) and outputs (LEDs).

 © 2019 by D. G. Meyer

Block Diagram:

 © 2019 by D. G. Meyer

Step (1):

Carefully examine the `lab11_top_template.v` file provided for this experiment on the course website. Rename this file *myuserid*`_lab11.v` (where *myuserid* is your 8-character login name) and insert your identifying information where indicated. Next, open ispLever and create a new Verilog project named `Lab11`, select `LC4256ZE-5TN144C` as the device to use, and import your *myuserid*`_lab11.v` source file. Next, import the "scrolling message display" Verilog modules (`msggen.v` and `dispshift.v`) you wrote for Lab 10 into your *myuserid*`_lab11.v` source file. Modify the `msggen.v` state machine so that a new message will start *immediately* as soon as `mSEL` is changed. Test your modified scrolling message display as outlined in the top template file. The four message strings should be as follows:

`mSEL`	*Scrolling Message Displayed*
0	*blank* → …
1	*blank* → S → E → C → U → r → E → *blank* → *blank* → …
2	*blank* → o → P → E → n → *blank* → *blank* → …
3	*blank* → E → r → r → o → r → *blank* → *blank* → …

Step (2):

Write a Verilog module `up_count_decode.v` that realizes a 3-bit synchronously resettable binary up counter with an integrated 3:8 decoder. This counter should also include an enable as well as the usual asynchronous reset. Test all the modes of your counter/decoder module as outlined in the top template file.

Checkpoint: Demonstrate Steps 1 and 2 to your Lab Instructor.

Step (3):

Write a Verilog module `lfsr.v` that realizes an 8-bit linear feedback shift-register that generates a pseudo-random sequence, as described in the referenced pages of the course text. Use DIP[4] to enable/disable the sequence generation (when disabled, the current state should be retained). Route the outputs of the LFSR to the top row of red LEDs (TOPRED), and use DIP[5] to control whether the LFSR outputs are "hidden" or displayed. Use the left pushbutton (S2BC) to *asynchronously* ***set*** the flip-flops that comprise the LFSR portion of the circuit.

Step (4):

Write a Verilog module `match_detect.v` that determines if the current combination data bit that has been entered on DIP[7] and "clocked in" (using the right pushbutton S1BC) matches the *corresponding digit* of the randomly-generated combination (output of the LFSR, displayed on TOPRED). Whether or not a match has occurred should be signaled on output `match_out`. Test your match detector module as outlined in the top template file.

Checkpoint: Demonstrate Steps 3 and 4 to your Lab Instructor.

 © 2019 by D. G. Meyer

Step (5):

Write a Verilog module `dcl.v` that realizes the sequence recognizer portion of the circuit. Use the left pushbutton (S2BC) to provide an *asynchronous reset* to all of the sequence recognizer's flip-flops. Use the right pushbutton (S1BC) to clock the sequence recognizer, use DIP[7] to enter the combination digits, and use DIP[6] to provide the (synchronous) "relock" signal. Note that the data bit entered has to "match" the corresponding bit of the pseudo-random combination generated by the LFSR in order to advance toward the "open" state, and that the MIDRED LEDs (the decoded counter outputs) "point" to the bit of the combination that is currently in play. While the combination is being entered, the jumbo yellow LED should be illuminated. If all eight digits of the combination are successfully entered, the jumbo green LED should be illuminated signifying the "open" state. Once the open state is reached, the "combination pointer" should return to its initial state (and *stay there*) and the sequence recognizer should ignore any further data entry from the pushbutton switches until the "relock" input (DIP[6]) is asserted, at which time clocking in either a "0" or "1" data bit will return the state machine to the initial locked condition. If, however, a "mistake" is made at any point during entry of the combination, the jumbo red LED should begin to flash at a 2.5 Hz rate (approximately) to signify an "alarm" condition. Note that an asynchronous reset (left pushbutton S2BC) is the *only means* by which the sequence recognizer can be reset once the alarm state is reached.

Checkpoint: Demonstrate Step 5 to your Lab Instructor.

Step (6): Write your answers to the following Thought Questions in the space provided**.**

1. Examine the fitter report generated by ispLever and determine the total number of flip- flops utilized by your design as well as the total number of P-terms and macrocells.

   Number of flip-flops: _____

   Number of P-terms: _____

   Number of macrocells: _____

2. Discuss why use of a Mealy model for the sequence recognizer portion of this design should be avoided.

   _____

   _____

3. Describe how the LFSR could be modified to start in state `00000000` instead of state `11111111`.

   _____

   _____

 © 2019 by D. G. Meyer