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	Draw state transition diagram	2	
Step 1	Create linear feedback shift register	4	
Step 2	Create self-correcting ring counter	4	
Step 3	Create sequence recognizer (digital lock) state machine	6	
Step 4	Create scrolling status display	5	
Step 5	Thought questions	4	
	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."

Printed Name:_____ Class No. __ __ __ __ - __

Signature:_____ Date: _____

Digital Combinational Lock with Pseudo-Random Combination

Instructional Objectives:

- To practice creating a sequence recognizer

Pre-lab Preparation:

- Read this document in its entirety
- Watch the Demo Video on the course website
- Review the referenced Module 3 lecture material
- Read pp. 736-740 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 DIP1=1 (and "hidden" when DIP1=0). The randomly-generated combination will keep changing as long as DIP0=1; it will "freeze" when DIP1=0. DIP7 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; DIP7 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 pushbutton switches S1 and S2: S2 (the switch on the left) will be used select a "0" (released) or "1" (pressed) as the combination bit to be entered, while S1 (on the right) will be used to "clock in" the selected data bit. A ring counter (displayed on the second row of red LEDs) will advance 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.

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 pushbutton switches until the "relock" input (DIP2) is asserted, at which time clocking in either a "0" or "1" data bit (via the pushbutton switches) will return the state machine to the 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 1 Hz rate and continuously scrolling the string "**Error**" across the four 7-segment LEDs.

Pre-lab Step (1):

On a separate sheet, draw a state transition diagram that specifies the behavior of the sequence recognizer portion of the circuit using a **Moore model**. Note that the 8-bit binary combination to be recognized is "unknown" (i.e., will be based on "matching" the randomly-generated combo).

Step (1):

Create an 8-bit linear feedback shift-register that generates a pseudo-random sequence, as described on page 738 of the course text. Use DIP0 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, and use DIP1 to control whether the LFSR outputs are "hidden" or displayed. Use DIP7 to *asynchronously set* the flip-flops that comprise the LFSR portion of the circuit. Next, create a counter that divides the on-chip oscillator (`tmr_out`) by four, producing a clocking signal of approximately 1 Hz. Use this "divided clock" as the clocking source for the shift register. Demonstrate LFSR functionality to your Lab Instructor.

Step (2):

Create an 8-bit self-correcting ring counter and route its outputs to the second row of red LEDs. Next, create a bounceless switch using pushbutton switch S1. Use the bounceless switch output to clock the ring counter. Use DIP7 to *asynchronously set* the right-most (low-order) bit of the ring counter and to *asynchronously reset* all of the higher order bits, i.e. asserting DIP7 should initialize the counter to **`00000001`**. Demonstrate ring counter functionality to your Lab Instructor.

Step (3):

Realize the sequence recognizer you designed for pre-lab. Use DIP7 to provide an *asynchronous reset* to all of the sequence recognizer's flip-flops. Use the S1 bounceless switch to clock the sequence recognizer, and use pushbutton switch S2 to successively enter the data bits of the binary combination (S2 does not need to be debounced). 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 ring counter "points" 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 ring counter 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 (DIP2) is asserted, at which time clocking in either a "0" or "1" data bit (via the pushbutton switches) 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 1 Hz rate to signify an "alarm" condition. Note that an asynchronous reset (DIP7) is the only means by which the sequence recognizer can be reset once the alarm state is reached. Demonstrate sequence recognizer functionality to your Lab Instructor.

Step (4):

Create a 7-bit wide, 4-word "left shift" register to facilitate status message scrolling. Route each "word" of the shift register's outputs to the corresponding 7-segment display (DIS1–DIS4). Use the 1 Hz signal created in Step 1 as the clocking signal for the display shift register. Use DIP7 to asynchronous reset the flip-flops that comprise the display shift register. Next, create a state machine that generates the character sequences to scroll across the 7-segment displays: "SECUrE" (when the sequence recognizer is in the initial "locked" state), "OPEn" (when the sequence recognizer reaches the "open" state), and "Error" (when the sequence recognizer reaches the "alarm" state). Demonstrate scrolling display functionality to your Lab Instructor.

Step (5): 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`.

4. Describe what is meant by the term *maximum-length sequence*.

