ECE 270 Lab Verification / Evaluation Form

Experiment 12

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	4-bit addition/subtraction problems	1	
Step 1	4-bit adder/subtractor with condition codes	3	
Step 2	4-bit magnitude comparator	1	
Step 3	4-bit modified LFSR	3	
Step 4	1-digit BCD round counter	3	
Step 5	2-digit BCD adder and point total register	5	
Step 6	Scrolling display with score and message string	5	
Step 7	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: _____

The "Academic Exercise" Game
-or-
The Radix Price is Right!

Instructional Objectives:
- To practice creating a variety of arithmetic circuits

Prelab Preparation:
- Read this document in its entirety
- Review the referenced Module 4 lecture material

Experiment Description:
In this experiment you will create what is indisputably an "academic exercise" with little social significance to the "cultural elites" who sit in the back row and text on their smart phones during class. It *may*, however, provide a unique opportunity to impress your friends from other majors.

Imagine you are creating a new TV game show called *The Radix Price is Right!*, in which a digitally-astute contestant attempts to guess the price of *really cheap*, perhaps even *worthless* "stuff" (like first generation iPhones or used political campaign yard signs). The added twist is that some of the stuff is "so cheap" that someone would have to *pay* you to take it (i.e. the price is a *negative number*). Like the "real" TV game show upon which this academic exercise is loosely fashioned, each time the contestant guesses the (randomly generated) "price" *exactly* he/she will earn 9 points. If the guess is "low" the contestant will earn 4 points, but if the guess is "high" the contestant will earn zero points. The contestant's point total will be displayed as a two-digit BCD number (on `DIS2..DIS1`), while the round number (starting at 1) will be displayed as a single BCD digit (on `DIS4`). A 4-bit linear feedback shift register (LFSR) will be used to generate the "radix (*two's complement*) price" for each round: here, the **radix price** will range from **-8_{10} (1000_2)** to **$+7_{10}$ (0111_2)** dollars. The contestant will enter his/her guess using `DIP3..DIP0`. Each guess will be "clocked in" using pushbutton switch S1, which will update the contestant's score, advance the round number, and advance the LFSR to its next state. After the guess for the 9th round is clocked in, the contestant's score followed by the string "**`yEAH`**" or "**`LoSEr`**" will continuously scroll across the 7-segment displays, based on whether or not a point total of **at least 50** was earned. For debugging purposes, `DIP7` will be used to either "hide" the LFSR state and comparator results or display them (enabling one to play in "cheat" mode). Pushbutton switch S2 will be used to asynchronously reset the game.

Pre-lab Step (1):
Solve the following 4-bit radix addition and subtraction problems by hand and determine how the condition codes will be affected in each case.

```
  0 1 1 1
+ 0 1 1 0
-----------
```

```
  1 0 0 1
- 1 1 1 1
-----------
```


Step (1):

Create a 4-bit adder/subtractor (based on the CLA described on page 8 of the Module 4 Lecture Summary notes) that generates CF, NF, ZF, VF condition codes (as described on page 6 of the *Lecture Summary* notes). Signed 4-bit operand `X3..X0` will be entered using `DIP7..DIP4`, while `Y3..Y0` will be entered using `DIP3..DIP0`. The add/subtract mode will be controlled by pushbutton switch S2 ("up" for subtract, "down" for add). Route the 4-bit sum/difference to `LED3..LED0`, and route the CF, NF, ZF, and VF condition codes to `LED7..LED4`, respectively. Verify adder/subtractor and condition code functionality based on the examples worked for Pre-lab, and demonstrate these results to your Lab Instructor.

Step (2):

Change the adder/subtractor created in Step 1 to a "subtract only" circuit and create a magnitude comparator that determines if X>Y, X=Y, or X<Y (where X is the price and Y is the guess) based on the condition codes generated (as described on page 6 of the Lecture Summary notes). If X>Y, then the jumbo yellow LED should be illuminated (guess was too low); if X=Y, the jumbo green LED should be illuminated (guess was correct); and if X<Y, the jumbo red LED should be illuminated (guess was too high). Verify that the magnitude comparator works as expected and demonstrate its functionality to your Lab Instructor.

Step (3):

Create a 4-bit LFSR, based on the design illustrated in Figure 8-52 on page 738 of the course text, to generate the value `X3..X0` (formerly connected to `DIP7..DIP4`). Use the modification described on page 739 that causes the LFSR to cycle through all 16 possible states (including `0000`). Use pushbutton switch S1 (configured as a bounceless switch) to clock the LFSR, and use pushbutton switch S2 to provide it with an *asynchronous reset*. Route the LFSR outputs to `LED11..LED8`, and use `DIP7` to control whether the LFSR state is "hidden" or displayed. Verify that the LFSR visits all 16 possible states in pseudo-random order, and demonstrate its functionality to your Lab Instructor.

Step (4):

Create a 4-bit counter that, upon "game reset" (assertion of pushbutton switch S2), initializes to state `0001` and increments by one each time it is clocked (using pushbutton switch S1) until it reaches state `1010`, at which point it "freezes" (i.e. advances no further). Route the counter output to `DIS4`, displaying the state as a single BCD digit. Verify that the "round counter" works as expected, and demonstrate its functionality to your Lab Instructor.

Step (5):

Create a two-digit BCD adder circuit based on the design presented in Lecture Module 4-F (documented on pp. 11-12 of the Module 4 *Lecture Summary* notes). This circuit will tally the contestant's points earned each round to his/her "running total" (stored in an 8-bit register, the outputs of which are routed to 7-segment displays `DIS2..DIS1`): he/she will earn 9 points for an exact guess, 4 points for a "low" guess, and zero points for a "high" guess. Asserting pushbutton switch S2 will reset the point total to `00`. Note that the maximum point total possible is **81**, based on making exact guesses on each of the 9 rounds. Verify that the point total display updates on each round as expected (and "freezes" once round 9 is completed), and demonstrate its functionality to your Lab Instructor.

Step (6):

Create a 7-bit wide, 4-word "left shift" register to facilitate message scrolling. Enable the routing of each "word" of the shift register's outputs to the corresponding 7-segment display (`DIS1`–`DIS4`) after the guess from round 9 is clocked in. Use a 1 Hz signal derived from the on-chip oscillator as the clocking signal for the display shift register. Use pushbutton switch S2 to asynchronously 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: the two-digit point total earned followed by the string "**`yEAH`**" (if score was ≥ 50) or "**`LoSEr`**" (if score was < 50). Verify that the final point total followed by the appropriate string is continuously scrolled on the 7-segment displays once the game is completed. Note: If the guess is set to `0000` for all 9 rounds, the point total should be **29**. With "cheat" mode enabled, demonstrate your ability to both "win" and "lose" *The Radix Price is Right!* game to your Lab Instructor.

Step (7): 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. Write down the sequence of states (as signed decimal numbers) generated by the LFSR (these are the "prices" the contestant is attempting to guess on each round):

   _____

   _____

3. Try playing the game in "non-cheat" mode (i.e. with the LFSR state and comparator display disabled) and determine your maximum score. Write down your series of guesses.

   _____

   _____

4. Empirically determine the maximum score one can obtain with a single "fixed" guess (i.e. same guess used for each round).

   _____

   _____

