Cpr E 281 FINAL PROJECT
ELECTRICAL AND COMPUTER ENGINEERING
IOWA STATE UNIVERSITY

FINAL Project

Objectives

The main objective of the final project is to gain experience in putting it all together and programming the Altera DE2 board to carry out an independent activity. We will design a circuit to realize one digital machine that will accept input from switches and output to LEDs and 7-segment displays.

Project Selection

You are free to choose any one of the following projects or a project of your choice to implement. *Your project must include a state machine.*

1. Random Number Generator

Design a system that will fill a register file with pseudo-randomly generated numbers. This system will include a register file to hold the values, an initial seed value for the random number generator, and a state machine to generate the random digits to be written to the register file.

You will first need to create a 4-bit wide 4-register file. It will have the following inputs and outputs:

R0 - **R3**: Contents of registers 0 through 3 (displayed via HEX LEDs).
WA: Write register address
WR: Write enable
LD_DATA: Write data input

The next component is an 8-bit parallel-in serial-out right-shift register. The seed value to the random number generator will be loaded into this register. Then, one-by-one, the bits of the seed will be shifted right out of the shift register to be fed to the state machine. Let the value that is about to be right-shifted out of the register be called S.

The last major component is the pseudo-random number state machine. In each clock cycle, it uses the previously computed digit and S to compute a digit in the range [0, 9] to be written into the register file. Its next state logic is defined as follows. Let f(t) represent the previously generated digit and S be the current input from the shift register:

$$f(0) = 0$$

$$f(t+1) = \begin{cases} 3f(t), & \text{if } S = 0 \\ f(t) - 3, & \text{if } S = 1 \end{cases}$$

Note that this arithmetic is performed with respect to mod 10 so that the values remain in the range [0, 9]. (For example, if f(t) = 2 and S = 1, the next number will be 9).

As these digits are computed, the results are stored in the register file sequentially. This means that at the first clock edge (i.e. t = 1), the output of the state machine should be written into R0 of the register file. Then at the second clock edge, the output should be written into R1. Consider how you might create this behavior with a 2-bit counter.

Use Case:

1. User loads an 8-bit value seed into the shift register.
2. Cycle the clock 8 times. At each active edge:
    a. Another random number is written to the register file.
    b. The shift register shifts its contents to the right
    c. The state machine begins calculating the next number using.
3. Observe the resulting values in the register file.

Some Notes:

- It may be helpful to display the contents of the shift register on the LEDs.
- Since f(0) = 0, the first number that will be written into the register file will be either 0 (in the case that S = 0) or 7 (if S = 1).
- The described use case will cycle through the register file twice, since the seed is 8 bits while the register file contains only 4 registers.
- Sample Case: The seed 11000011 will yield the sequence 7, 4, 2, 6, 8, 4, 1, 8 with final register values R0 = 8, R1 = 4, R2 = 1, R3 = 8.

2. Stack Arithmetic

For this circuit, you will need to implement a simple finite state machine to control a stack of 4-bit unsigned integers. (If you are not familiar with the stack abstract data type, check Wikipedia.) We will design this stack so that its contents can be used to perform some simple arithmetic. The user must be able to perform four operations:

- **Push** - A value is added to the top of the stack.
- **Pop** - The top-most value is removed from the stack.
- **Pop with Add** - The top two values on the stack are popped from the stack, added together, and their result is pushed back onto the top of the stack.
- **Pop with Subtract** - The top two values on the stack are popped from the stack, used to perform subtraction, and their result is pushed back onto the top of the stack. (The first value popped is subtracted from the second value popped.)

To keep things simple, the maximum stack depth will be four. Your circuit should display the complete contents of the stack on HEX3 through HEX0. To be clear, HEX3 should display the first value pushed onto the stack, HEX2 should display the second value, and so on.

Some notes about error handling:

- **Stack Overflow** - If a push is performed, but there is no more room on the top of the stack, the contents of the stack should not be changed and the *stack overflow register* should be set. Hook this register up to an LED to notify the user. This register and LED should remain set and lit until the board is reset.
- **Stack Underflow** - If any of the three pop operations are performed without there being enough values on the stack to perform that operation, the contents of the stack should not be changed and the *stack underflow register* should be set. Hook this register up to an LED to notify the user. This register and LED should remain set and lit until the board is reset.

Some hints:

- You may find it convenient to implement the stack using a 4-bit register file with 4 registers, two read ports, and one write port.

- If there is only one value on the stack, only HEX3 should display a value. The other HEX displays should not have any lit LEDs.

3. Vending Machine

Design a vending machine that accepts nickels, dimes and quarters and outputs two products A and B.

The machine accepts only one coin at a time, controlled by switches SW1 and SW0. A nickel, dime, or quarter is added if SW1, SW0 = 01, 10, or 11 respectively.

The vending machine cannot hold more than 35 cents. Thus, if inputting a coin would cause the total to go over this value, the total should not be updated and the coin should be rejected.

Products A and B cost 10 and 25 cents respectively. A user indicates they wish to purchase these products by setting SW2 to 1 (for product A) or SW3 to 1 (for product B). If both SW2 and SW3 are set to 1, the machine dispenses the change.

A product should not be dispensed if there is not sufficient money inside the vending machine.

You are free to choose whether or not the remaining change is dispensed along with the product (e.g., purchasing product A with 25 cents in the machine brings the total down to either 15 or 0 cents).

If both a coin is added and a product is selected, the product will have priority and the coin will be rejected.

A rejected coin should be indicated by a single LED.

Display the amount of money in the vending machine using two seven segment displays. Also display the value of the coin the user wishes to enter into the machine.

Another seven segment display will indicate the output of the vending machine:

- 0: nothing
- A: product A
- B: product B
- C: change

Prioritize the display of a dispensed product over any returned change (if that case would arise in your design).

4. Sorting Machine

Design a sorting machine. The idea is to design a two-port read two-port write register file with k registers. The data are stored in registers using some input switches (address and data are specified by switches). Then there are two counters, C1 and C2. A four state machine sorts the numbers as follows.

Load the registers with initial values. Start the machine in state S0.

State S0:

C1 is initialized to 0. Go to state S1.

State S1:

C2 is initialized to C1 +1. (Need add 1 circuit). Go to State S2.

State S2:

Read two registers from two addresses specified by C1 and C2.
Call them D1 and D2.
Feed D1 and D2 into a maximizer/minimizer circuit.
It yields MAX and MIN on two ports.
At clock edge
MAX is written into register C1 and MIN is written into register C2.
(This swaps max and min).

If C1 = k-2 then Go to state S3
Else if C2 = k-1 then increment C1 and Go to State S1
Else increment C2 and Go to State S2

State S3:

Registers are displayed on 7-segment displays.
Done.

5. Other Project

Propose a project of your own design. It should include a state machine and must be of sufficient difficulty. If you choose this option, you must discuss your intended design with a TA to see if they agree that it is of sufficient difficulty. This discussion can be over e-mail. Please contact your lab TAs.