Lab 4: Counters and Pseudorandom Number Generation

CSC 258, University of Toronto

October 8, 2014

About this Lab

Learning Objectives

1. To introduce you to counters.
2. To introduce you to linear shift feedback registers, a type of pseudorandom number generator.
3. To give you more practice with sequential circuits, and exposure to two of their applications.

Marking Scheme

Successful completion of Parts I and II will result in one in-lab mark, and completing Parts III and IV will result in a second in-lab mark. To obtain full marks you will also need to complete Parts V and VI of the lab, marked as Advanced.

1 Pre-lab Assignment

As usual, all items in this document labeled **PRELAB TODO:** are to be completed before arriving to the lab, and are worth half the marks for the labs. Items marked **INLAB TODO:** are to be completed in lab and will all be marked. Perform the prelab tasks specified in parts I to VI, and bring your prepared Verilog code to the lab to be marked by the TAs. *Note for Parts II and III there are written questions to complete, along with a screenshot to provide in Part III.*

1.1 Part I

Consider the circuit in Figure 1. It is a 4-bit synchronous counter which uses four T-type flip-flops. The counter increments its value on each positive edge of the clock if the *Enable* signal is asserted. The counter is reset to 0 by setting the *Clear* signal low. Note that the *Clear* signal is a *negative, asynchronous* reset. Consult the lecture slides for more on synchronous vs asynchronous behaviour.

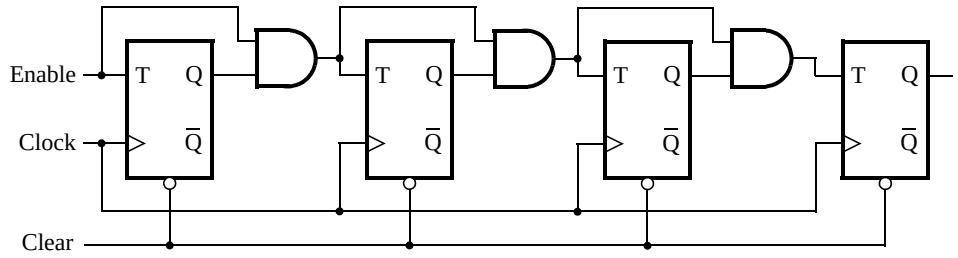

Figure 1: A 4-bit counter.

PRELAB TODO: Determine the chips and pin assignments needed to implement a 4-bit counter of this type using TTL logic.

1. Determine the number of chips that you'll need to implement this circuit on the breadboard. The labs do not stock T flip-flop chips, but they do have JK flip-flops (74LS107). Search the web for 74LS107 to find the pin layout for these chips, and consult the notes or the textbook to determine how these JK flip-flops can be used to create T flip-flop behaviour.

2. For each of the gates in this circuit, label the pin numbers that your design will use to implement this 4-bit counter.

3. The rest of this part will be performed in the lab.

1.2 Part II

Again, consider the circuit in Figure 1. **PRELAB TODO:** You are to implement a 4-bit counter of this type.

1. Write a Verilog file that defines a 4-bit counter by using the structure depicted in Figure 1. You will want to use CLOCK_50 as your input for the clock, which is the built-in 50MHz clock built into your DE2 pin assignments.

   Your code should include a T flip-flop module that is instantiated 4 times to create the counter. Compile the circuit. How many logic elements (LEs) are used to implement your circuit? **PRELAB TODO:** What is the maximum frequency, F_{max}, at which your circuit can be operated?

   (To find this maximum frequency, compile your code in Quartus, and then run the TimeQuest Timing Analyser. The F_{max} summary can be found in the Reports option, under Datasheets.)

2. Simulate your circuit to verify its correctness.

3. Augment your Verilog file to use the pushbutton KEY_0 as the *Clock* input, switches SW_1 and SW_0 as *Enable* and *Clear* inputs, and 7-segment displays *HEX1-0* to display the decimal count as your circuit operates. Make the necessary pin assignments needed to implement the circuit on the DE2 board, and compile the circuit.

4. Implement a 4-bit version of your circuit and use the Quartus II RTL Viewer to see how Quartus II software synthesized your circuit. What are the differences in comparison with Figure 1?

1.3 Part III

Another way to specify a counter similar to the one in Part II is by creating a module that contains a register unit, and having the module add 1 to the register value at each clock pulse. This can be accomplished by creating a value (e.g. Q) of type **reg**, and by using a Verilog statement like the following in your always block:

$$\text{Q} <= \text{Q} + 1;$$

PRELAB TODO: Compile a 16-bit version of this counter and determine the number of LEs needed and the F_{max} that is attainable. Use the RTL Viewer to see the structure of this implementation and comment on the differences with the design from Part II. Print out a screenshot of the RTL Viewer and hand it in for your prelab, along with your comments on the differences.

1.4 Part IV

Design and implement a circuit that successively flashes digits 0 through 15 on the 7-segment display *HEX0*. Each digit should be displayed for about one second. Use a counter to determine the one second intervals. The counter should be incremented by the 50-MHz clock signal provided on the DE2 board. Do not derive any other clock signals in your design – make sure that all flip-flops in your circuit are clocked directly by the 50-MHz clock signal.

1.5 Part V – Advanced

A linear feedback shift register (LFSR) produces a sequence of pseudorandom numbers – at each new clock cycle a new pseudorandom number is generated using the previous state of the circuit.

The initial value of the LFSR is called the seed; as the LFSR is deterministic, the sequence of values that the circuit generates will be based on its seed. There are a number of different types of LFSRs; one is the 4-bit Fibonacci LFSR[1]:

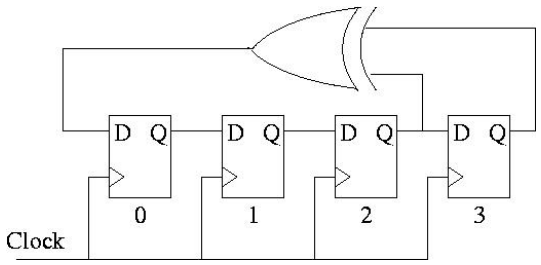

Figure 2: A four-bit Fibonacci LFSR. Note that if you use a seed of 0000, the LFSR will never change state! An animated example of this circuit in action is available at `http://en.wikipedia.org/wiki/File:LFSR-F4.GIF`

PRELAB TODO: Write Verilog code for a 4-bit Fibonacci LFSR. Use KEY0 for the clock, a seed value of 1111, and display the current number using a 7-segment display. (Hint: get the LFSR working with LED outputs before switching it to the 7-segment display!)

1.6 Part VI – Advanced

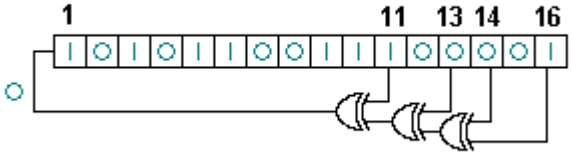

Figure 3: A sixteen-bit Fibonacci LFSR.

Figure 3 shows a 16-bit Fibonacci LFSR. **PRELAB TODO:** Write code to implement this circuit, generating a new number on the number display every second. Arrange for $KEY0$ to reset the bits of this 16-bit number to a standard seed value of 1000000000000000 (note that an all-zero seed value would have the same problems as Part V).

(Hint: since you'll be using CLOCK_50, what will you need to slow the input down to 1Hz?)

Note that in Part III, we generated a 4-bit number which we could display on one single 7-segment display. Here we generate 16-bit numbers. Before you begin, determine how many 7-segment displays you will need for this. Also, just like in Part III, you might want to consider using the **reg** keyword here.

[1]Image credit: `http://www2.hawaii.edu/~lucam/EE260/S10/Labs/Lab9/Lab9_2.jpg`

2 In lab work

For each part of the pre-lab, compile and download your code to the FPGA. Test the functionality of your design by toggling the switches and observing the displays.

1. **INLAB TODO:** Implement your counter design on the protoboard, using the chips specifed in your design. Use switches on the switch board as your EN, Clear and Clock inputs and the lights as output. Once you've got it working, demonstrate it to one of the teaching assistants.
2. **INLAB TODO:** Show your board to your TA for Part II
3. **INLAB TODO:** Show your board and the result of your work to your TA for Parts III and IV
4. **INLAB TODO:** Show your board to your TA for Part V
5. **INLAB TODO:** Show your board to your TA for Parts VI

Clean up your station once you are done the lab.