CSE 352 Laboratory Assignment 3

Introduction to Registers

Issued: August 14th, 2013
Due: August 24th @ 5:20PM

The objective of this lab is to introduce you to edge-trigged D-type flip-flops as well as linear feedback shift registers. Chapter 3 of the Harris&Harris text covers registers and clocking. Unfortunately clock signals are generally much too fast to see what the registers are doing. Therefore, we will implement the clock in this lab using one of the push-buttons on your board to generate the clock. This will allow you to generate clock edges slowly and one at a time so you can see what happens to the registers.

This lab consists of both lab check offs and a series of short answer questions. Please write your short answer questions on a separate sheet of paper and turn it in at the end of the lab. We will not check off these short answer questions.

Part 1: D Flip-Flops

1. Find the '74 package, which has two positive-edge-triggered D flip-flops. You'll note each flip-flop has a data input (D), a clock input (CP), two outputs (Q and Q') and two additional inputs, SD and CD, which are active-low set and clear control inputs (they have effect when set to 0 and none when set to 1).

2. Insert the '74 chip into your breadboard and connect the D input to one of the switches, a button to the clock input, and Q to one of the LEDs. Make sure to also connect SD and CD to a logical 1. You do this by connecting these terminals to Vdd.

3. Spend some time experimenting with the flip-flop. Toggle your switch so that a value of 1 is on the D input to your flip-flop and press the button. What happens to the LED you connected to Q? Try changing the value of D and push the button again. Try changing D back and forth while not pushing the button wired up as your clock. Note how Q only changes when you press the push button. This is a positive edge-triggered flip-flop, so changes in the output only occur when a rising clock edge (recall that when you press the button, its output changes from 0 to 1.

4. We will experiment with clock skew by using even numbers of inverters to delay the propagation of the clock signal. Connect the second D flip-flop on the ’74 package as follows: use the output from the first flip-flop as the input for the second flip-flop and wire your clock button to both clock inputs so that they share the same clock. Your circuit is now a 2-bit shift register. Try shifting in some bits and observe how the bits shift through the shift register each time the clock ticks.

5. Now, add a ’04 inverter package to your breadboard and wire together two of the inverters in a chain back to back to implement a simple buffer. Connect the clock button to the clock of the first flip flop and to the input of the first inverter in the chain. Connect the inverter output at the end of the inverter chain to the clock of the second flip-flop. The delay from the two inverters will now skew the clocks received by the two flip-flops. That is, the clock arrives at the second flip-flop slightly later than the clock at the first flip-flop. Does your shift register still work?

6. Now add four more inverters to the inverter chain, for a total of six inverters, adding even more delay to the second clock signal and skewing the second clock signal even further. Does your shift register still work?

7. Now reverse the clocks to the two flip-flops so that the clock arrives at the second flip-flop before the first flip-flop. Does your shift register still work?

   Short Answer #1: Explain the behavior of the shift register for all four different ways of connecting the clock (4, 5, 6, 7).

Check Off Requirements

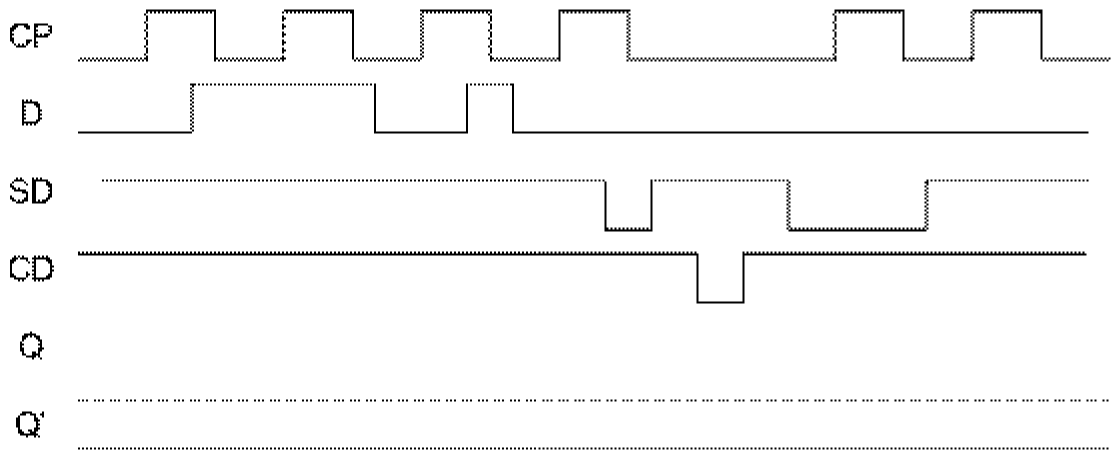
1. Show your TA a case where the two-bit shift register functions as expected. Briefly explain and justify how you know the circuit is functioning correctly.
2. Show your TA a case where the two-bit shift register is not working because of clock skew. Explain and justify how you know the circuit is functioning correctly.

Part 2: Set and Clear

1. Now it is time to experiment with the set and clear control inputs. Connect these two inputs to two switches instead of Vdd. Make sure the switches are set to 1 so that your registers behave as they did.

2. Now, experiment with the asynchronous set and clear (SD and CD) inputs using the two switches. Make sure that the switches are initially set to output a 1. Now, set the value of Q to 0 using the D input and the push-button (clock). Flip the SD switch. What happens? Did you have to press the push button? Asynchronous inputs take effect immediately, without waiting for the next clock edge. Repeat the experiment with CS instead of SD. Try setting both SD and CS to 0 (set and clear at the same time): which dominates? Does the flip-flop set or clear?

   Short Answer #2: Complete the following timing diagram by drawing the signals Q and Q’ and turn this in.



Part 2: Linear Feedback Shift Registers (LFSRs)

LFSRs are shift registers with feedback that causes the shift register value to cycle through a maximal length sequence of output values. In the case of a 4-bit shift register, a maximal length sequence would have 15 (16 - 1, since the all-zero pattern is not counted) different outputs. Since the function can be implemented efficiently, this means that an LFSR is much faster and cheaper than a binary counter, although it counts in a “random order”. This makes LFSRs very attractive when we need to count to large values but don't care about what the sequence is (that is, they don't have to be consecutive binary numbers). Variations of LFSRs are often used as random number generators as well - consecutive output patterns can be made to look quite different and are uniformly distributed over the space of all possible patterns. You can read a lot more about LFSRs at Wikipedia: you can find a list of functions that will generate maximal sequences for any number of bits from 4 to 32 and beyond.

For example, a 4-bit LFSR with maximal length sequence can be implemented using the function: D1 = Q4 xor Q3, and a larger 8-bit LFSR with D1 = Q8 xor Q7 xor Q6 xor Q1. Even a 32-bit LFSR can also have a maximal sequence (2^{32}-1 patterns long) with a function of only one 4-input XOR: D1 = Q32 xor Q31 xor Q30 xor Q10.

1. Wire up your '377 octal D-FF to form a 4-bit shift register. Connect the four FF outputs to four of the LEDs. Connect the output of a 2:1 multiplexer to the first input with a switch connected to the MUX's control input. The two MUX inputs should be the value of another switch and the last output of the shift register (the fourth bit). Verify the operation of your shift register by setting the input to come from the switch. Go through a few clock cycles shifting in different values. Make sure to tie the enable input of the '377 to a value rather than leaving it floating as it may not function properly without a valid logic level on that input.

Check Off Requirements

1. Show your functional 4-bit shift register to a TA and prove that it works by cycling through a few cycles and inputs.

2. Verify the operation of your shift register loading it serially from the switch. Now shift the pattern 1, 1, 0, 0 into your shift register. Flip the input MUX switch so that the last output is now fed back into the input. Go through a few clock cycles by pressing on the push-button you have wired up as your clock. You should see your pattern shifting in a circular pattern through the register.
   Short Answer #3: How many different patterns are there in all before the output pattern on the LEDs repeats itself?

3. Invert the output of the shift register being fed back into the MUX. (To save yourself some time, use an XOR gate to do this inversion.) Repeat the previous task with this new configuration.
   Short Answer #4: How many different patterns are there?

4. Remove the inverter and replace it with an XOR gate with the 4th and 3rd FF outputs as its inputs. Connect the output of this XOR gate to the input MUX of the shift register. This is a 4-bit LFSR. Begin by shifting in zeros into your shift register (use the switch input to the MUX). Now flip the MUX to select the output of the XOR gate to be the input. Go through a few clock cycles. Does the pattern change? Now, shift all ones (instead of zeroes) to set up the shift register and then run through a few clock cycles.

Check Off Requirement

1. Demonstrate your LFSR from the last step to the TA. Explain and justify why your LFSR works.

Short Answer #5: How many patterns do you go through before they begin to repeat? Is this a maximal sequence? Try different taps instead of 4th and 3rd, for example, 4th and 2nd. How many different patterns does this configuration generate?

Submit your completed short answer questions to the TA. Make sure your name and SID is on the top right corner of each page.

Consolidated Check Off Requirements

1. From Part 1. Show your TA a case where the two-bit shift register functions as expected. Briefly explain and justify how you know the circuit is functioning correctly.
2. From Part 1. Show your TA a case where the two-bit shift register is not working because of clock skew. Explain and justify how you know the circuit is functioning correctly.
3. From Part 2. Show your functional 4-bit shift register to a TA and prove that it works by cycling through a few cycles and inputs.

4. From Part 2. Demonstrate your LFSR from the last step to the TA. Explain and justify why your LFSR works.
3. Submit your answers to the short answer questions to the TA by the lab due date.