CSE 352 Laboratory Assignment 3

Introduction to Registers

Assigned: Oct. 18/20
Due: Beginning of Next Lab

Objectives

The objective of this lab is to familiarize you with 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 use one of the push-buttons on your board ***(we recommend button 0)*** 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.

Part 1: D Flip-Flops

The '74 package has 2 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. These last two are active-low (they have an effect when 0 and none when 1) asynchronous set and clear inputs. 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, such as VDD to ensure that they have no effect for this first part.

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 after you press the push button. This is a positive-edge-triggered flip- flop, so changes in the output only occur on the rising clock edge (recall that when you press the button, its output changes from 0 to 1).

Clock Skew

We will experiment with clock skew by using even numbers of inverters to delay the propagation of the clock signal. Wire up the second D flip-flop on the ’74 package. Use your output from the first flip-flop as the input for the second flip-flop. Also, 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; because both flip-flops' clocks "tick" at exactly the same time, the two registers perform a "parallel assignment".

Now, add an '04 inverter package to your breadboard and wire together two of the inverters in a chain, back to back. 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 two inverters' delay 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. See if your shift register still works.

Does it still work if you add two more inverters to the inverter chain, for a total of four inverters; skewing the second clock signal even further?

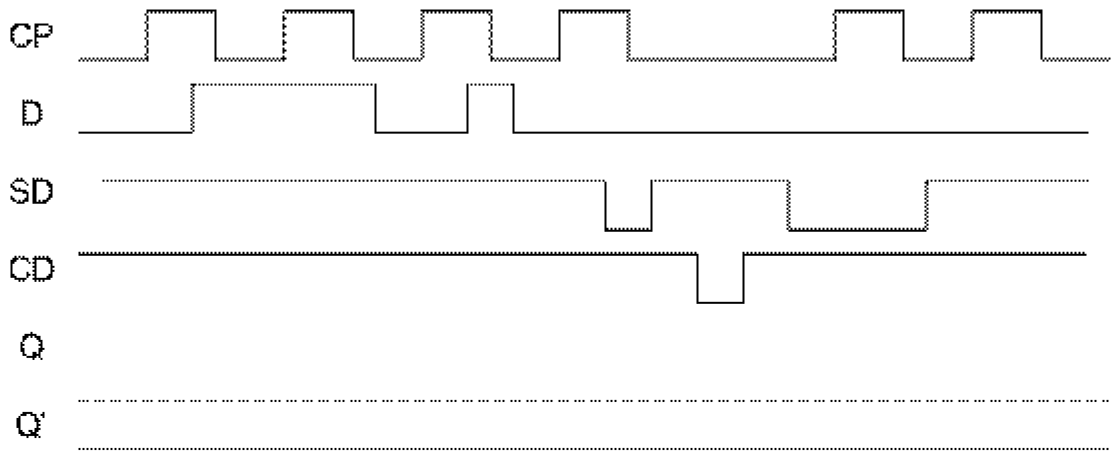
Now reverse the clocks so that the delayed clock goes to the first flip-flop and the button clock goes to the second flip-flop. What happens now when you run your shift register?

Writeup #1: Explain the behavior of the shift register for all four different ways of connecting the clock.

Demo #1: Demo your two-bit shift register to the TA for the case where it works incorrectly.

Now it is time to experiment with the asynchronous set and clear inputs. Connect these to switches instead of VDD (logic 1) to which they were previously connected. Make sure 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. Flip the SD switch. What happens? Did you have to press the push-button? Asynchronous input take effect immediately, without waiting for the next clock edge. Repeat the experiment with CD instead of SD. Try setting both SD and CD to 0 (set and clear at the same time), which dominates? Does the flip-flop set or clear?

Writeup #2: Copy the timing diagram below onto your own piece of paper and fill in the timing signals for Q and Q'.



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 : each of these sites includes a complete 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 will have the following function: D1 = Q4 xor Q3. A larger 8-bit LFSR with D1 = Q8 xor Q7 xor Q6 xor Q1 will have a 255 pattern long maximal sequence. Interestingly, a 32-bit LFSR can also have a maximal sequence (2^{32}-1 patterns long) with a function of only 4 output variables, namely, D1 = Q32 xor Q31 xor Q30 xor Q10.

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.

Demo #2: Demo the operation of your shift register to a TA.

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 the button you have wired up as your "clock". You should see your pattern shifting in a circular pattern through the register.

Writeup #3: How many different patterns are there in all before the output pattern on the LEDs repeats itself?

Writeup #4: Invert the value of the last bit being fed back around before it goes into the MUX. Repeat the previous task with this new configuration. How many different patterns do you see?

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 in all ones (instead of zeros) to set up the shift register and then run through a few clock cycles.

Writeup #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. What configuration did you try and how many different patterns does this configuration generate?

Demo #3: Demonstrate the operation of your LFSR to a TA.

Lab Demonstration/Turn-In Requirements

There are 3 demos that you need to get checked off with a TA.

There are 5 write-up questions that you need to turn in.