CSE 370 Introductory Laboratory Assignment

Introduction to Registers

Assigned: Friday, October 31, 2008
Due: End of Lab Section

Objectives

The objective of this lab is to introduce you to edge-trigged D-type flip- flops as well as feedback registers. Flip-flops and registers are very similar, their key difference lies in when their outputs change. You can refer to Chapter 6 of your textbook Contemporary Logic Design for more information if you haven't had a chance to cover it in class yet. Typically for flip-flops and registers you use a clock, however the clocks provided are far too fast for the LEDs to respond and for you to notice any changes so we will simulate clock pulses throughout this lab by using a button. Even though a button is not a clock, what happens when the button is depressed is roughly the same, a pulse is sent through the circuit. You can consider button pushing to be a very slow clock.

Before You Begin

If you followed the instructions properly in the previous lab, when you turn on your board for this lab you will notice that the old program loaded on the FPGA is still there, which means you can make use of the switches, LEDs and buttons directly connected through the input/output connectors on the board. If you find that this is not the case, call over a TA and they will assist you in restoring your old settings. Of course by now you are well versed in the FPGA, which means you can undertake the task of rerouting the switches, LEDs and buttons yourself if you so choose to.

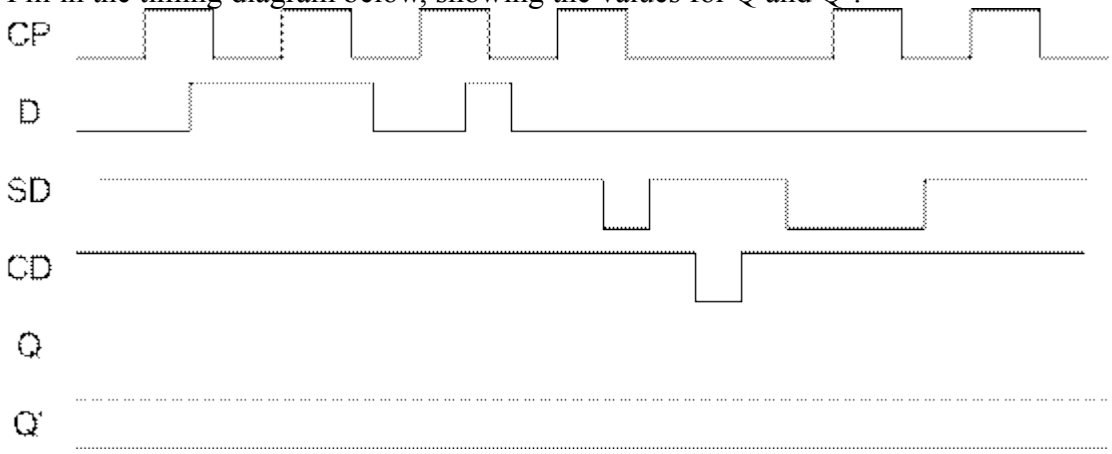
D Flip-Flops

Tasks:

1. The '74 has 2 D flip-flops in one chip package. 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 affect 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 logic 1, such as VDD to ensure that they aren't triggered for this first part.
2. Spend some time experimenting with the flip-flop. Toggle your switch so that a value of 1 can be inputted into your flip-flop and press the button. What happens to the LED you connected to Q? Try a different value of D and push the button again. Try changing D back and forth while not pushing a button wired up as your clock. Note how Q only changes after you press the push

button. This is a synchronous flip- flop, changes in the output only occur after a rising clock edge (positive edge-triggered, this occurs every time you press the button because the button generates a positive pulse). You can surmise that if you had a faster clock, such as someone pushing the button for you very quickly, whenever you flipped the switch the effect would appear almost instantaneously, depending on the speed of the clock.

3. Now it is time to experiment with the asynchronous set and clear inputs. Connect these to switches instead of the VDD (logic 1) they were previously connected to. 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 affect 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?
4. Fill in the timing diagram below, showing the values for Q and Q'.



Linear Feedback Shift Registers (LFSRs)

LFSR's have an interesting property that a particular function of a subset of the outputs will cause the shift register 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, the all-zero pattern is not counted) different outputs. If the function can be implemented efficiently, this capability can be much easier to implement than building a binary counter (recall that a binary counter is a specialized adder but still has a long carry-chain or larger and larger gates to do lookahead).

Binary counters with a large number of bits can be quite expensive in terms of the logic they require. On the other hand, LFSRs with maximal sequences can be made with input functions that are low fan-in (depend on only a few of the register's outputs) and do not have a carry-chain. This makes LFSRs very attractive when we need to count to large values but don't care about what the patterns are (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 uniformally distributed over the space of all possible patterns. You can read a lot more about LFSRs at New Wave Instruments: this site 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 (232-1 patterns long) with a function of only 4 output variables, namely, D1 = Q32 xor Q31 xor Q30 xor Q10.

Tasks:

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 - basically, this is a four-bit version of the shift register you made at the end of the last lab assignment. Make sure to tie the enable input of the '377 to a value rather than leaving it floating.
2. 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. How many different patterns are there in all before the output pattern on the LEDs repeats itself?
3. 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?
4. Remove the inverter and replace it with an XOR gate with the 4th and 3rd 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 go through a few clock cycles. How many patterns do you go through before they begin to repeat? Is this a maximal sequence? Show your LFSR in operation to one of the TAs. Try different taps instead of 4th and 3rd, for example, 4th and 2nd. How many different patterns does this configuration generate? Demonstrate your LFSR to the TA to get checked off for Part 2.

Lab Demonstration/Turn-In Requirements

A TA needs to "Check You Off" for each of the tasks listed below.

1. Turn in the completed timing diagram in Part 1. Demonstrate your 4-bit shift register from Part 1.
2. Demonstrate your LFSR from Part 2.

Comments to: cse370- webmaster@cs.washington.edu