Lab 2

Data Storage

In this experiment, you will design and construct a simple 2-bit, four-word shift-register storage unit.

Assignment

- Read Experiment #2 description and complete the Lab 2 Pre-Lab before the lab section. Again you will need the information from the Dat a Sheet to correctly wire up your circuit.
- If you are not familiar with registers and counters, you might wish to review our Notes on Registers and Counters.
- Take a look at the TTL wiring guide for helpful hints on building your circuit.
- Work on Lab 2 report (**JOINT** report from now on, follow the guidelines in the lab 2 report outline) and Lab 3 Pre-Lab after the lab section.

Demo

You need to create the shift register based 4-word 2-bit memory. At most only one of the three control switches (FETCH, STORE, LDSBR) will ever be high at any given time.

- Demonstrate that you can load the SBR with data from the switches by using the LDSBR switch. (1 points)
- Demonstrate that you can store data from the SBR into the shift register memory at the location specified by the SAR switches using the STORE switch (1 points)
- Demonstrate that you can fetch the word at the memory location specified by the SAR into the SBR by using the FETCH switch. (Can be demonstrated simultaneously with demonstrating STORE) (3 points)
- Note: you may get 1.0 point of partial credit from these 3.0 for demonstrating that your shift register can shift right on each pulse of the clock. If you get the entire assignment working, you do not need to demonstrate this independently.

FAQ

- **"When both FETCH and LDSBR are trying to load SBR with data from memory and DIN respectively, should one override the other?"**
    - You can safely assume that only one of the FETCH/STORE/LDSBR switches will be up at any given time.

- **"I can't seem to find my chip, and even if it's there, I cannot find a data sheet for it"**
    - Our chipset are numbered as 74xx, 74 being the 74-hundred family of TTL (see GG. 2). The "xx" is the important number you should be looking for on both the chips themselves and in the data sheets. There will be some letters in between the numbers, such as 74YYxxZZ. The "YY"s and "ZZ"s stands for the specific manufacturing variations of the chip, but all of them should be logically identical. Therefore, when you are looking for your chip from the chipset or the specific data sheet, you should be primarily looking for the "xx" and not the letters. The only exception is when you find the specific data sheet for your chip, if there exist two or more distinct pin numberings based on these "YY" and "ZZ" variations, then you should find the corresponding pin numbering for your chip based on the letters. For example, for the 7485 comparators in Appendix B Data Sheets p.9, there is a pin numbering distinction between SN7485/SN74LS85/SN74S85 and SN74L85J/N. But other than these exceptions, you do not need to care about the letters.

- **"I wired up my chip but it doesn't work ☹"**
    - Most likely it is because you have not yet wired up all the relevant pins properly. If you look at the data sheets carefully, you will find some extra input pins that are not a part of the chip's logic, e.g., for the 74157 MUX chip, you can find a "STROBE" pin that seems irrelevant. However, if you take a closer look at the provided function table, you can see that the "STROBE" needs to be set low in order for the MUX to function at all. Therefore, it is always a good idea to READ YOUR DATA SHEETS THOROUGHLY!

- **"The lecture says the the "shift register needs to be continuously shifting". I'm not entirely sure what this means. does anyone know?"**
    - As it literally means, the shift register needs to be fixed in the shifting mode, so that once your circuit is powered on, the shift register should be constantly shifting. This implies that you will need to figure out a way (see the lecture slides for hints!) to be able to store and fetch at the proper cells from the constantly-shifting registers.

      Note that since the register is always shifting, it is meaningless to indicate "absolute" storage addresses. Rather, all addresses are "relative" in that if you happen to store data X in address Y, you can pick any random cell Z to store the data, and arbitrarily associate cell Z with address Y, so that later on when you wish to fetch from address Y, you go look for cell Z (the animation on the lecture slides will be able to explain this better than words).

- **"What does that mean to have 4 words? It asks for 2 bit 4 words. I don't understand this part."**
    - It means that you have 4 distinct data, with each data being 2-bit long. The 2 bits in a single data has to go through the exact same process at any time, hence the two shift registers (and also two MUXs, two SBR FFs, etc.). The four data has to be

continuously circulating, hence each register is 4 bits long.

You are given both the counters (74169 or 74193) and the d FFs (7474). The complete list of available chips are listed on GG.4. Please be aware that the letters in between the numbers (the X's as in 74XX169X) mostly means different chip packaging and pin layout, and therefore you should consult with the datasheet for the proper pin layouts of your specific chip.

- **"If we use a 4 bit ripple counter instead of a synchronous counter, will it cause a problem? Because it has glitch. We are currently looking for chip 74169 and 74193, but it looks a bit complicated to use..."**
    - Short answer: Probably not.

      Long answer: The glitches in the ripple counter happen right after every clock edge, which temporarily outputs wrong values before settling on the correct one. In your Lab 4 design, the counter value is compared against the SAR using some type of comparator, which might then raise the wrong flag before settling on the right one. Practically, this most likely will not cause a problem, since we are designing synchronous circuit, all components will only update upon a clock edge, where the glitches should have already disappeared by then. However, it is still recommended that you still use the regular counter (it is not that complicated!).

      *Additionally, some students have seen issues with the 74193 chip when using the switch-boxes as the clock. It should work properly when using the function generator. As a result, we recommend that you use the **74169** counter.*

      On the other hand, if you are dealing with asynchronous circuits, everything will change on the fly. If a ripple counter with guaranteed glitches is used, those glitches will very likely to cause the circuit to divert from what it is supposed to do, with no way of reverting or stabilizing the circuit back to its proper function, and this could be very bad.

- **"Do we have to provide truth table and K-map for this lab report..?"**
    - Since most likely you did not need to derive any equation from any table, you do not have to include any for the Lab 2 report.

      However, you should not simply put down the logic diagram without any explanation either. Remember that you should always tell the reader how you come about your design (showing table/K-map is only among one of the many options!). For example, for Lab 2, some of the obvious key design feature worth explaining are:

      1. How come there's a MUX in front of the shift register? What are the options? How to select them?

      2. How come there's a MUX in front of the SBR? What are the options? How to select them?

      3. According to #1 and #2, you probably have come up with some kind of controller for the MUX selections. How is the controller built? How do your inputs affect the thinking of the controller?

      Of course, there are much more to talk about for circuit A. In general, you should guide the reader through your design steps (whether through charts, tables, or descriptions), and give logical reasoning to every major decision you've made on the design (as in contrast to giving every trivial wirings down to the gate-level, e.g. we AND this with that and the output is XORed with blah blah blah... Way too much detail!)

      Also, please don't forget to "describe" the circuit as well, i.e. what does it do? How does it function? What are its design features, requirements and/or constraints? For example, Lab 2 specifically requires circuit A to be continuously shifting. The readers will probably not be aware of this if you don't note it down somewhere!

Implementing Your Design

This document has a few "golden rules" you should keep in mind as you implement your design, which will make your debugging process much more smooth.