ECE 385

EXPERIMENT #2

Data Storage

I. OBJECTIVE

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

II. INTRODUCTION

Conceptually, random access memory (RAM) is a storage device arranged as a set of binary words that can be individually identified and accessed through the use of unique addresses (see Figure 1).

		STORAGE address	STORAGE contents
SAR			
	$\boxed{101}$	word 0	$\boxed{0110}$
		word 1	$\boxed{1100}$
SBR		word 2	$\boxed{0000}$
	$\boxed{1110}$	word 3	$\boxed{0000}$
		word 4	$\boxed{0000}$
FETCH		word 5	$\boxed{1110}$
		word 6	$\boxed{1101}$
STORE		word 7	$\boxed{0000}$

Figure 1: An Eight-word Storage Unit Using 4-Bit Words

To fetch a word from storage, the unique word address is placed in the Storage Address Register (SAR) and a FETCH signal is sent. The binary string or a content of the specified word appears in the Storage Buffer Register (SBR) a short time later (exactly how much later depends upon the particular technology used for the storage).

To store a word into storage the unique word address is placed in the SAR, the binary data string to be stored is placed in the SBR, and a STORE signal is sent. The binary string in the SBR is stored in the word whose address is specified in the SAR. The previous contents of the word are destroyed by the STORE operation.

Detailed knowledge of the exact technology used to implement the storage is usually of little concern to the computer engineer. Cathode ray tubes and magnetic cores were once used for storage. Semiconductor RAMs are common now. You should be able to construct a storage unit from parallel-in/parallel-out shift registers, multiplexers, counters, and combinational logic.

One storage technique uses serial-in/serial-out (SISO) shift-registers shifting synchronously. A single 1024-bit SISO shift register could be used to provide 1024 words of storage, where each word is a single bit long. Note that with a 1024-bit SISO shift register, only the output of the rightmost flip-flop and only the input to the leftmost flip-flop are available. In theory, a shift register can be built at much lower cost compared to a RAM. This is because there are far fewer pins and interconnections in shift register than in RAM. In addition, the storage cell of a shift register could be very simple, a capacitor for example. The shift operation is simply moving charge from one capacitor to the neighboring capacitor. Such shift registers are called Charge Coupled Devices (CCDs). Today, CCDs are primarily used in imaging applications, such as in digital cameras. The charge in a cell slowly decays and therefore must be refreshed before it is lost. For this reason a SISO memory based on CCDs must be continuously shifted to keep the information from being lost.

Words larger than a single bit can be constructed by using more 1024-bit shift-registers clocked synchronously. Typically, 16 such SISO shift registers would be used to construct 1024 words of storage, where each word is 16 bits long. More generally, an n-bit, m-word shift-register storage consists of n m-bit shift-registers shifting together (see Figure 2).

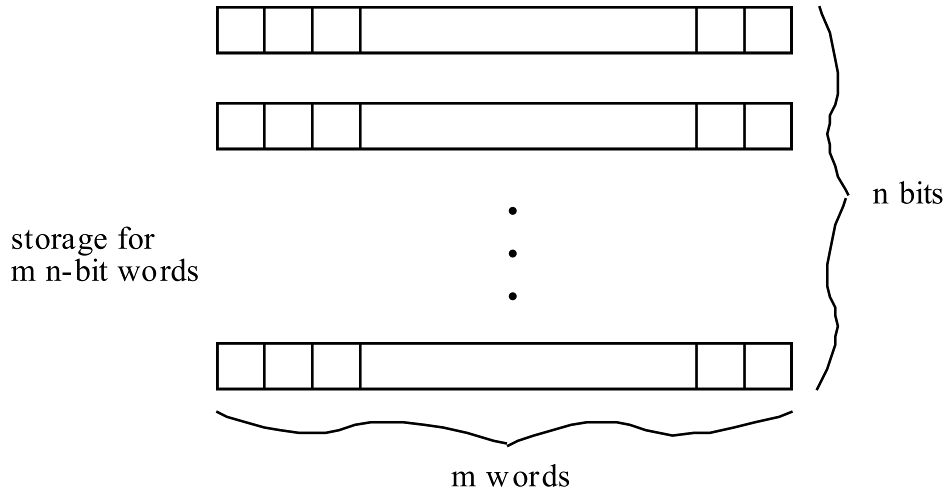

Figure 2: Configuration of a Shift-register Storage

As mentioned earlier, an alternative to the above storage devices are those devices that are built with "static" logic elements (SRAM). This is a setup where the storage device can retain data as long as a specified supply voltage is maintained. These SRAM chips are readily available from a number of manufacturers with varying features and parameters.

III. PRE-LAB

A. Design, document, and build a 2-bit four-word shift-register storage unit using two 74LS194 shift-registers without using the parallel load or parallel output capability. Of the 74LS194 data (non-control) inputs and outputs, you may use only the serial input and the rightmost (Qd) output. For the purposes of this experiment, imagine that the maximum and minimum clock period is specified as 1 millisecond for the 74LS194. The registers must be shifted on each clock pulse. The clock must run continuously – do **not** gate the clock.

Signal Definitions:

LDSBR	When LDSBR is high, the SBR is loaded with the data word DIN1, DIN0.
FETCH	When FETCH is high, the value in the data word specified by the SAR is read into the SBR.

STORE	When STORE is high, the value in the SBR is stored into the word specified by the SAR.
SBR1, SBR0	The data word in the SBR; either the most recently fetched data word or a data word loaded from switches (note that when none of the LDSBR/FETCH/STORE switches is set, SBR should maintain the data in it)
SAR1, SAR0	The address, in the SAR, of a word in the storage
DIN1, DIN0	Data word to be loaded into SBR for storing into storage

Use flip-flops for the SBR. DIN1, DIN0, SAR1, and SAR0 should be obtained from switches. FETCH, STORE and LDSBR should also be obtained from switches. Display SBR1 and SBR0 on LEDs. You may also wish to include the display of other signals in your design for convenience when debugging your circuit. You can assume that only one of the FETCH/STORE/LDSBR switches will be set at any given time. Do not combine the FETCH/STORE switches!

To design the shift-register storage unit, we first need to look at the required specs. The most crucial requirement is for the shift registers to shift continuously, while using the serial input and output to store and fetch the data. We can break down our circuit operation into four operations: *load*, *read*, *write*, and *do nothing*. Let's first imagine the scenario where the circuit is turned on, but we are neither loading, reading nor writing. This is the most common state of the circuit, where no action is taken from the user – *do nothing*. Our requirements dictate that the shift registers have to continuously shift, where any potentially stored data will be shifted out of the registers and into the void. To prevent losing any data, we will need to redirect the data shifting out of the registers back by connecting the serial output of the registers to their serial input, where the stored data will now be looping over and over again in the shift registers. However, during a *write* operation, we do want to replace the old data with new data. To serve both purposes, a 2-to-1 multiplexer (MUX) can be placed at the serial input of the shift registers, taking either the new data or the old data depending on the current operation.

The rest of the circuit operation hinges upon the SBR, which serves two purposes: loading new data from DIN during a *load* operation, and reading from the shift register during a *read* operation. Notice that <u>all</u> of the registers must be continuously shifting, including the SBR. Thus, it is easy to see that the input of the SBR takes in these three different choices by using a 3-to-1 MUX (or a 4-to-1 MUX with one input ignored), again depending on the current operation.

But what is the 'current operation' at any given moment? Surely the desired operation is dictated by the user using the switches, but the inputs alone is not sufficient to tell each part of the circuit what to do. For example, if you would like to read from a specific address in the shift registers, you would first set the SAR to the specific address then you would hit the FETCH switch. But since the shift registers are constantly shifting data in and out of their serial ports, when exactly do you load the data into the SBR? How do you exactly tell what input the MUX should choose from? To solve the various problems associated with controlling and timing, it is generally not a good idea to use the inputs to directly control the various circuit components. Rather, it is almost always desired to have a centralized control logic that takes in all the inputs, process the request, and sends out various signals to control the circuit components. Figure 3 shows a general block diagram for the proposed circuit design. The most common form of a control logic is a state machine, which we will discuss in the next experiment. In this experiment, we will improvise a simpler control logic base on the properties of our specific circuit.

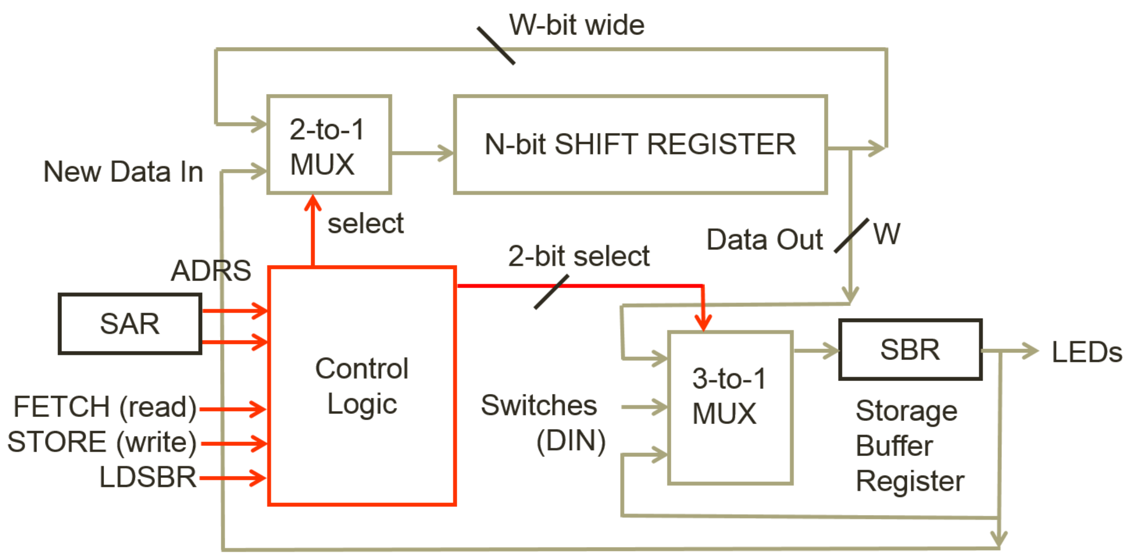

Figure 3: Block diagram of the shift-register storage unit.

First, notice that our shift registers are four word long, that is, each data will take exactly four shifts/clock cycles to loop back to its original location. We can exploit this property by employing a 2-bit counter (four distinct values) to keep track of the internal data address, then use a comparator to match the internal address with the SAR. Note that since the register is always shifting, it is meaningless to indicate "absolute" storage addresses. Rather, all addresses are "relative." If you wish to store data X in address Y, you can write the data into a random cell Z at the moment when the internal data address from the 2-bit counter matches the SAR. This random cell Z will now be associated with the address Y. Later on, when you wish to fetch from address Y, you wait for the internal counter to match the SAR again, and that is when cell Z once again becomes available for reading or writing. Another interpretation that might be useful is that the counter always keeps track of the address associated with data to be shifted out from serial output/into serial input of the shift register array at the up-coming clock edge. Note that in order to control the MUXs, the 'select' signals generated by the control logic has to take into account of the input switches and the comparator output (to indicate if we are currently looking at the correct address for reading/writing).

Your pre-lab writeup should contain a written description of your circuit operation, a block diagram, operation of the controller, a logic diagram and layout documentation.

HINT: Use the Pulse Generator to provide a basic clock. Continuously clock the shift-register <u>and</u> a counter that keeps track of which word is currently available. Use combinational circuitry (or 74LS85) to check for a match between the available word and the SAR.

B. Meet with your lab partner and wire up and test your design <u>before</u> coming to the lab. Use either the mini-switchbox circuit that you built at the end of Lab 1 (detailed in the General Guide) or attend an open lab session to test your circuit with the real switchbox. Only the clock input needs to be de-bounced in order to strep through your circuit (why?).

Demo Points Breakdown:

1.0 point: When LDSBR is high, the data in DIN is loaded from the switches into SBR

1.0 point: When STORE is high, the contents of the SBR are stored into the location specified in SAR

3.0 points: When FETCH is high, the data word specified by the SAR is read into the SBR

Note: you may get 1 point of partial credit from these 3 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 (since you may need to rewire a shift register to demo this).

IV. LAB

Finish testing and demonstrating your circuit to your TA. Correct the design if necessary and document your changes for the lab report.

Follow the Lab 2 demo information when debugging is completed.

V. POST-LAB

Your post-lab writeup (notes) should contain a corrected version of your pre-lab writeup and an explanation of any remaining problems in the operation of the circuits. This will aid the writing of your lab report.

VI. REPORT

In your lab report, should hand in the following:

- An introduction;
- Written description of the operation of circuits from the pre-lab;
- Block diagrams for part A;

- Design steps taken for all circuits. This includes but not limited to design considerations on the SBR MUX, the Shift Register MUX, and the control logic. Truth tables/K-maps leading to the final circuit design should be included (if any);
- One (1) component layout sheet, with the package layout of all circuits (DO NOT draw the interconnections! Refer to GG.20 for the proper documentation);
- Circuit diagrams for all circuits;
- Requested documentation from the lab;
- A conclusion regarding what worked and what didn't, with explanations of any possible causes and the potential remedies.