ECE385 Experiment #2

Eric Meyers, Ryan Helsdingen

Section ABG; TAs: Ben Delay, Shuo Liu

February 3rd, 2016

emeyer7, helsdin2

I. Introduction

THE purpose of this lab is to create a synchronous storage unit consisting of two four-bit shift registers, plus some control logic, which will have four basic operations: *load, read, write, and do nothing*. The idea behind creating a storage unit out of shift registers is to explore different implementations of RAM and to explore basic principles of storing and loading digital data.

II. Description of Circuit

As stated above, the primary purpose of the circuit that was designed in this lab was to create a 4-by-2 bit storage unit out of shift registers and TTL chips. As with any storage unit, the operator must be able to specify whether they would like to ”write” or ”read” to/from the storage. Therefore, there must be a convention used to choose a specific address from the unit and read those contents into a buffer.

For this lab, the Storage Address Register (SAR) will contain the address being read, and the Storage Buffer Register (SBR) will contain the contents from the address specified by SAR.

Three switches were used on the I/O switchboard to achieve the read/write commands: STORE, FETCH, and LDSBR. Two switches were used for the data-in given by the user (DIN0, DIN1). Finally, two more switches were used so that the user can specify the SAR for reading (SAR0, SAR1), for a total of 7 switches used on the I/O switchboard.

The actual storage of data is implemented by using a universal shift register that is loading in one bit at a time through the serial-right input. This shift register is continuously shifting its contents, which are specified by the user. ”Read” and ”write” commands are accomplished by setting the FETCH or STORE signal high.

A STORE signal takes the contents specified by the SBR and moves it into the shift register to the appropriate location specified by the SAR.

LDSBR is used to store the data specified by the user from the DIN1 and DIN0 switches into the SBR.

A FETCH signal takes the contents in the address specified by the SAR and stores it into the SBR.

Because the bits are continuously shifting inside the shift register, some control logic must be added to select the proper address specified by the user in the SAR. In order to do that, a two-bit counter must be used along with a comparator to check to see if the proper address is being accessed inside the shift register.

The MUXs were used for selection logic on loading into the SBR register and determining the proper timing to load into the correct address as specified by the counter.

The following TTL chips were used:

- 4-bit Binary Counter - SNLS169
- 4-bit Magnitude Comparator - DM74LS85
- (2) 4-bit Universal Shift Register - SN74LS193
- 4-to-1 MUX - DM54LS153
- 2-to-1 MUX - DM54LS157
- Quad 2-Input NAND - DM54LS00
- Quad D-Latch - DM74LS75

These components were assembled on a protoboard and (as stated earlier) connected to a I/O switchboard which allows the operator to control the bits going in/out of the system. The protoboard was then connected to a function generator used to generate a 1 kHz 5 V peak-to-peak square-wave with +2.5V offset used for the clock in the components.

III. Block Diagram

Please refer to Figure 1 in ”Section IX: Figures” of this document for the Block Diagram of the Storage Unit.

IV. Design

A truth table was created to determine the control logic on both the MUXs and the comparator. Please refer to Figure 2 in ”Section IX: Figures” of this document for the Control Logic Truth Table created during the prelab.

The team then created a Karnaugh-Map from this truth table to determine the proper minterms and the resulting logic function for each of the multiplexers. The K-Map for the 4-to-1 MUX selection logic is shown below. FETCH is represented by ”F”, LDSBR is represented by ”L”, and ”CompareOUT” is represented by C.

C \ F/L	00	01	11	10
0	1	1	X	1
1	1	1	X	0

Selction Line 1 (S1)

C \ F/L	00	01	11	10
0	1	1	X	1
1	1	0	X	1

Selection Line 0 (S0)

This gives us SOP function as follows:

$$S1 = \neg C \vee \neg L$$
$$S0 = \neg C \vee \neg F$$

These logic functions were then constructed into a AND-OR circuit, then converted into a NAND-NAND circuit using deMorgan’s Laws.

The selection logic for the 2-to-1 MUX was much less complex than the logic for the 4-to-1 MUX and simply was a direct line to a NAND of the comparator and the STORE signal. This is so that it can store the proper signal when the correct address is reached in the counter.

V. Component Layout Sheet

Please refer to Figure 3 in ”Section IX: Figures” of this document for the Component Layout Diagram created during the prelab.

VI. Circuit/Logic Diagrams

Please refer to Figure 4 in ”Section IX: Figures” of this document for the Logic Diagram of the Storage Unit.

VII. Documentation from Experiment

The documentation required from the lab is expalined throughout this document as needed. Please refer to those sections as necessary.

VIII. Conclusion

Overall, the team managed to design and construct a synchronous 4-by-2 bit storage unit from shift registers, however some minor setbacks occured with the implementation of our circuit.

The circuit was constructed prior to lab and during the testing phase, the team encountered a problem in the logic. The storage unit would allow the user to store into the SBR using LDSBR and the DIN switches perfectly and consistently. The storage unit would also allow the user to write these values (in the SBR) into the shift-register using the LOAD switch. However, if the user loaded words into the shift-register in a particular order, and attempted to fetch one of those values, using the FETCH switch, the values read into SBR would be incorrect. Specifically, instead of a solid color indicating high/low on the LED, the SBR would blink the ”correct” value every fourth clock-cycle.

The team narrowed this down to a problem in the logic with both the LDSBR and FETCH switch. This is because when both the FETCH switch and the LDSBR switch were turned on, the system would function properly and output the correct values to the LED. The team tried to further examine the error and attempted to deconstruct the logic to correct the circuit.

When FETCH is on and LDSBR is off, the improper values were being fetched from the shift register during three clock cycles, and when FETCH is on and LDSBR is on, the SBR was being loaded with the proper value during one clock cycle and was doing nothing on the other three clock cycles, which is the intended functionality.

The team then constructed truth tables of the improper logic (shown in the below table), and it was concluded that a single bit was being flipped, which led to the intended functionality during when LDSBR and FETCH were both on.

C	F	L	S1	S0
0	0	0	1	1
0	0	1	1	1
0	1	0	0	1
...	...	...	...	...

The team then realized that the NAND gates were improperly wired in the circuit that was constructed. This is what led to the single bit being flipped in the table above.

The team unfortunately ran out of time attempting to fix the error however, received almost-full credit on the demo.

The prelab circuit was the same design as the final circuit, so there are no changes to document. However, due to wiring inconsistencies.

Some other possible causes of error might be synchronization issues. The function generator was daisy-chained to each chip which might have caused an issue in the timing of the rising edge of the clock cycle.

IX. Figures

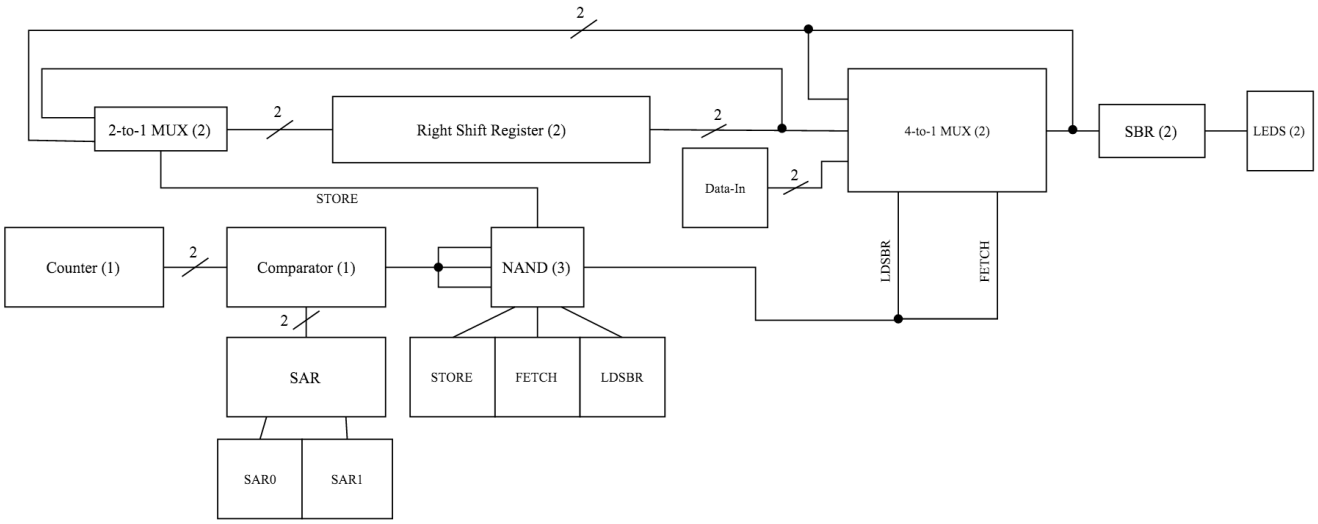

Fig. 1: Block Diagram from Pre-Lab

4-1 MUX					2-1 MUX		
Comparator OUT	FETCH	LDSBR	4-1 MUX inputs		Comparator OUT	STORE	MUX input
			S1	S0			S
0	0	0	1	1	0	0	1
0	0	1	1	1	0	1	1
0	1	0	1	1	1	0	1
0	1	1			1	1	0
1	0	0	1	1			
1	0	1	1	0			
1	1	0	0	1			
1	1	1					

Fig. 2: Control Logic Truth Table from Pre-Lab

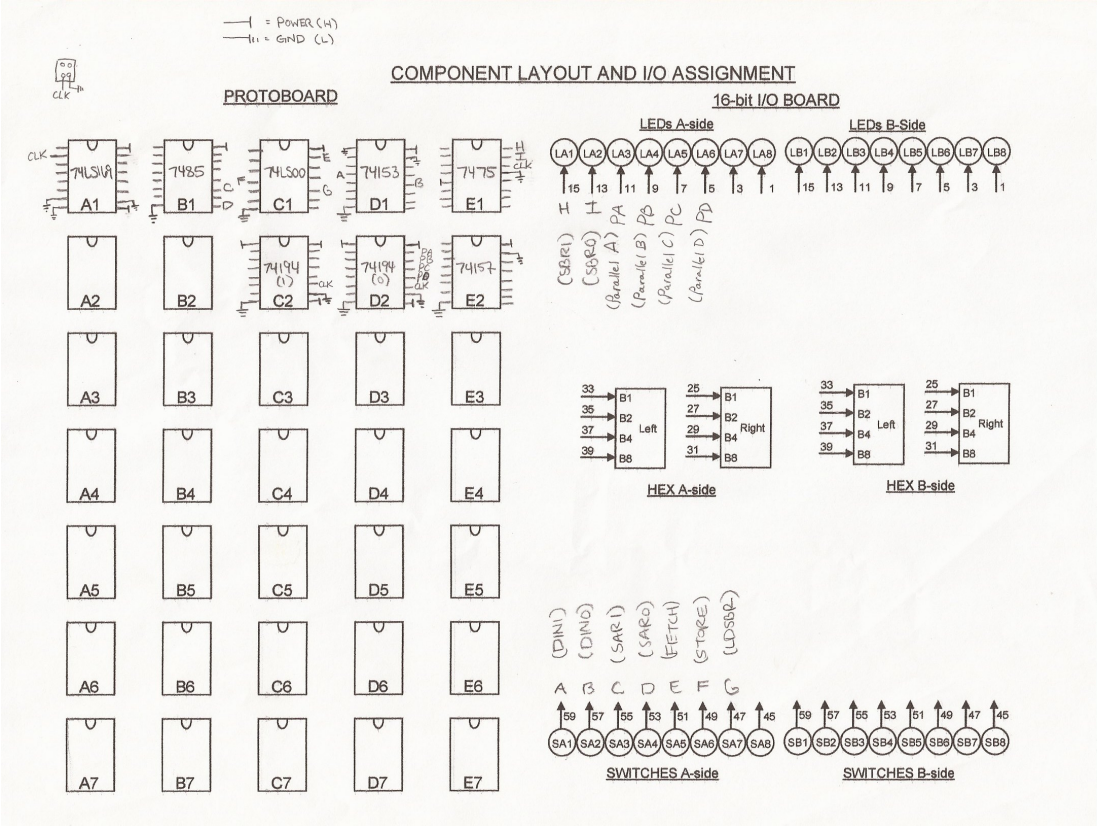

Fig. 3: Component Layout for Storage Unit

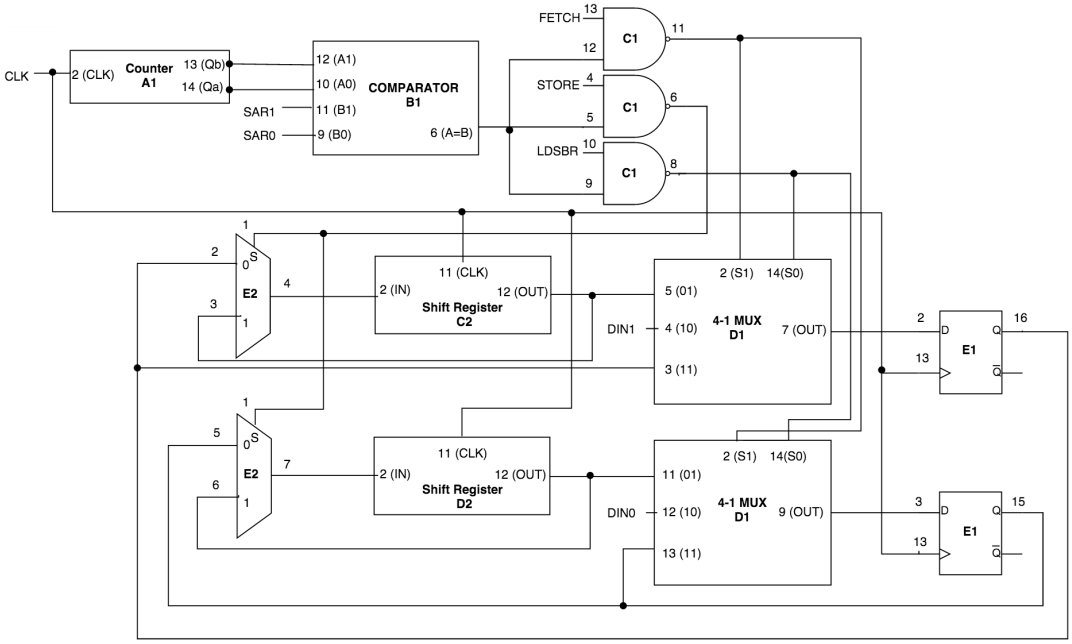

Fig. 4: Logic Diagram from Pre-Lab

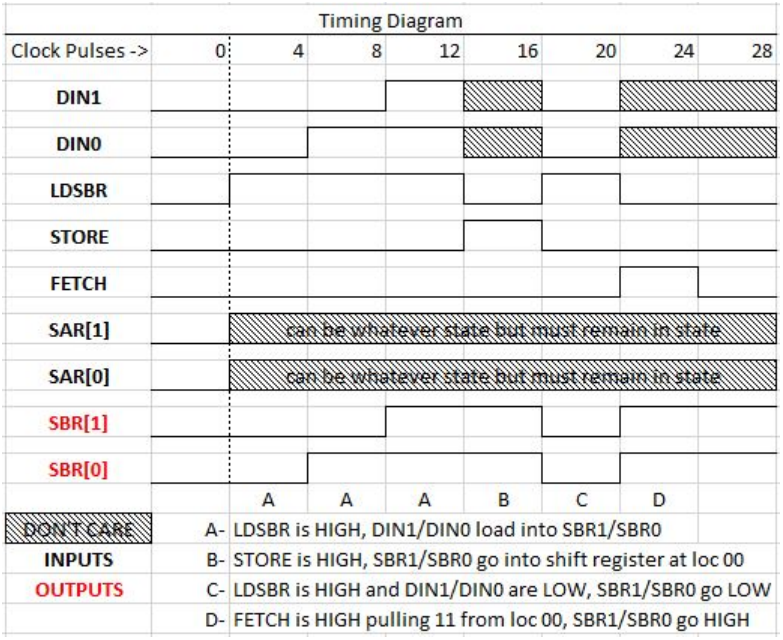

Fig. 5: Timing Diagram for Storage Unit