ECE 385

Spring 2019

Experiment #2

Data Storage

Rob Audino and Soham Karanjikar
Section ABJ, Friday 2:00-4:50
TA: David Zhang

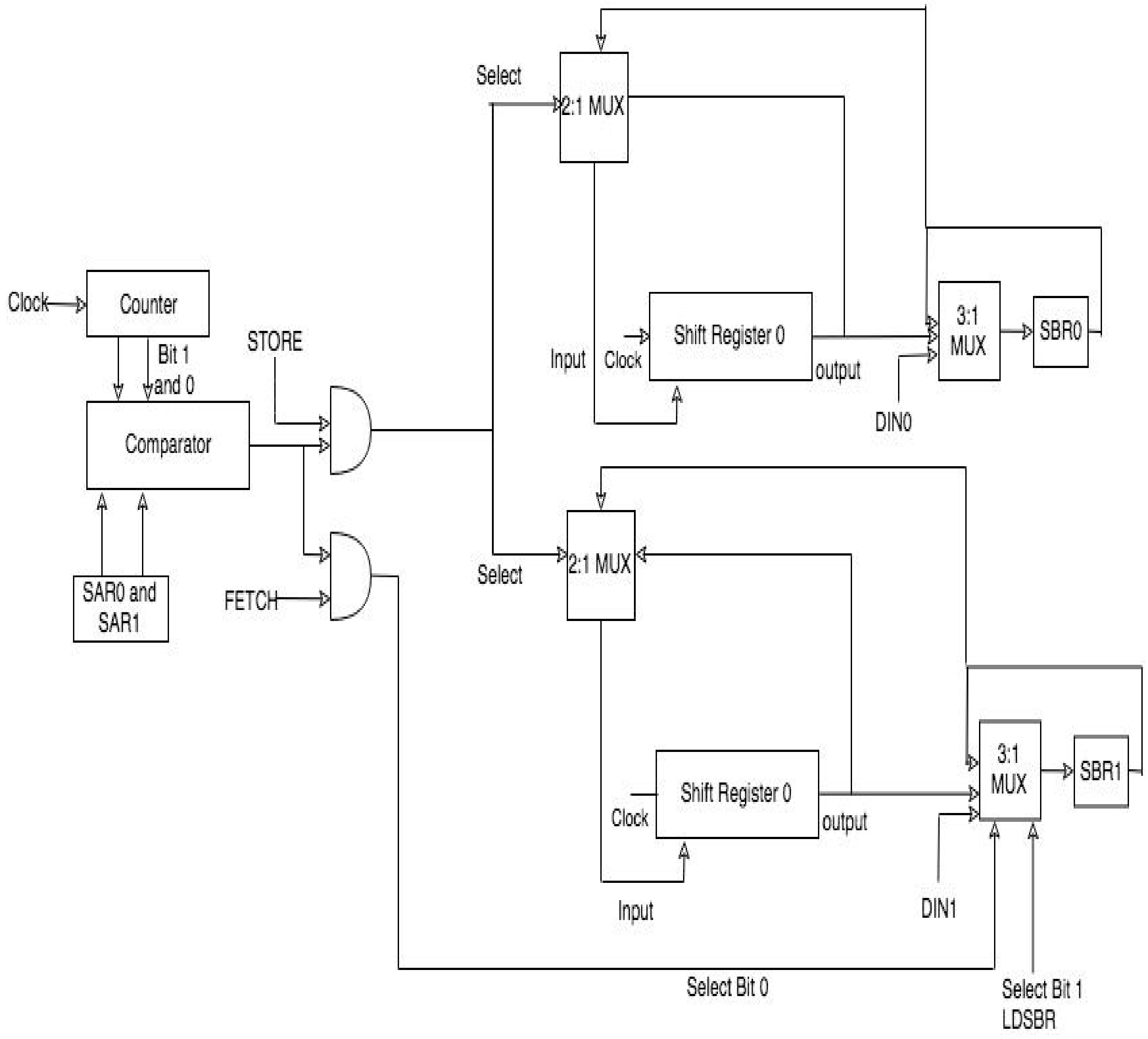
Introduction:

This week's lab focused on very basic data storage. We were given the task of creating a 4 word, 2 bit shift register memory with the caveat that parallel loading and outputs were not allowed. Our shift registers had to be constantly shifting, and have a serial input and output. The circuit had to have a Read (fetch) function, allowing the user to fetch previously data stored in the shift register memory from a chosen index. It also had to have a Write (store) function, which writes a two bit word from the SBR into a chosen index of the shift register memory. In addition, it must have a LDSBR function, which lets the user write a two bit word using the DIN switches into the SBR. Lastly, it had to have a NOP operation, which constantly cycled the bits through the shift register and preserved the memory of the shift register while not writing, reading, or loading any new memory into the SBR. Our solution to this problem utilized: D flip flops for the SBR; a few basic gates, counters, and comparators for the control logic; and multiple copies of the MUX's and shift registers to accommodate for the processing of the two bit words.

Operation of Circuit:

The circuit worked through very simple logic: loop the data in the registers until you want to store or fetch data from it. SAR is the location from where you want to STORE/FETCH. When it matches the counter, then an operation can be successfully completed. Whatever you want to read gets loaded into SBR, which is just two D Flip-Flops. If you want to write, however, two different operations need to be done. You need to load your data into SBR using switches for DIN and turn LDSBR on, which signals that new data is coming in. Then, you set the STORE signal high, which takes whatever is in SBR, and once the counter matches SAR, writes it to the location you chose.

Block Diagram



Pre Lab Write Up:

Given a rough diagram of what our circuit layout should look like in lecture, we realized that since we were designing a Shift Register Memory for 4 2-bit words, we would need to use another copy of several of the components in the circuit. Anywhere that the 2-bit words would travel, we needed to use a second chip for the other bit. Therefore, we needed two 2-to-1 MUX's, two 4-bit Shift Registers, and two 3-to-1 MUX's. The control logic could be routed in parallel to the 2-to-1 MUX's and 3-to-1 MUX's, so we only required one copy of the control logic. However, since each word was two bits, we required the use of two DIN switches as the input for these bits, and since the shift registers were 4 bits long, we needed two SAR switches to indicate which position in the registers to read from or write to. In addition, a separate switch for the Read (Fetch), Write (Store), and Load SBR functions.

To create our SBR, we realized that we required some sort of storage unit, and also needed the capacity for two bits. Therefore, we used two D flip-flops, each one attached to the end of one of the 3-to-1 MUX's. We were able to use chips for the exact purposes for the 2-to-1 MUX's, 4 bit shift registers, but ended up using 4-to-1 MUX's instead of 3-to-1 MUX's, and just ignored the last input.

The key to creating the control logic was realizing what did and didn't matter for the select bits that went to the 2-to-1 MUX's and 3-to-1 MUX's. Since it only required a single select bit, we started with the 2-to-1 MUX's when looking at the control logic.

We very quickly realized that by default, the 2-to-1 MUX's were set to the 'shifting' input, which meant that the bits could continuously and circularly shift through the shift registers. The only time that the other input to the 2-to-1's were selected was when data needed to be moved from the SBR's into the 2-to-1's. Therefore, the Write switch was the true select bit that controlled the 2-to-1's. However, only using the Write switch as the select bit for the 2-to-1's would cause a problem: the data would often be written to the incorrect place. This problem also presented itself later when dealing with the Read function.

In order to fix this placement issue, we created a 'counter' that would indicate when each of bit of the shift registers were at the front of the register. For example, if we labeled the bits in the shift registers as 'a,b,c,d' from left to write, the counter would read '10' when the bit that was originally in the 1st position (index 0) arrived at index 2. Conversely, this also meant that the bit that originally was at index 2 arrived at index 0. We realized that this counter would need to count bitwise from 0 to 3, so we utilized a 4 bit counter that counted from 0 to 15, but only used the two least significant bits, which would continuously count 0 to 3. Because the Read function drew data from the back (right) end of the shift registers and not the front (left) end like the Write function did, we needed the signal for the Write function to be one clock tick ahead of the Read function. This was already built into the circuit, as the signal that was being read had to go through the SBR, which was a D flip flop and caused delay. The SAR would end up being the two bits (switches) that selected which index of the shift register to write into or read from. We utilized a comparator with the SAR and the counter as inputs to make sure that we knew exactly when we could read or write. We then used the output of this comparator as well as the Write switch as the inputs of an AND gate whose output would become the select bit for the 2-to-1's. In doing so, we ensured that writing would only occur when the Write switch was selected to 1, and would only write in the index of the shift register selected by the SAR.

The next challenge was creating the control logic for the 2-bit select for the 3-to-1 MUX's. The inputs to the 3-to-1's were as follows: the output of the shift registers, the input from the DIN switches, and circular input from the SBR. We decided that by default, the 3-to-1's should be selected to simply circulate data from the SBR. The Load SBR function would be activated by a switch, and should simply take data from the DIN switches and move it into the SBR. As a result, the DIN input to the 3-to-1's should only be selected when the Load SBR switch is on. Lastly, the output of the shift registers should only be selected as an input when we want to read data from the shift registers into the SBR. However, we need to make sure we are reading data from the index that we want, so the SAR switches should be able to select which index the data is read from. We utilized the Read switch ANDed with the comparator described in the last paragraph to make sure that the output of the shift registers only goes into the SBR when the Read switch is 1 and only pulls from the index selected by the SAR switches. So in conclusion, the most significant bit of the 2-bit select was simply Load SBR and the least significant bit was the Read signal ANDed with the comparator described earlier. This exactly corresponds with our assignment of the 2-bit select output: 10 (2) selects the DIN switches, 01 (1) selects the data output from the shift registers, and 00 (0) selects the output of the SBR.

Truth Tables:

<u>Select Bit for 2-to-1 MUX's:</u>

Write (Store)	Comparator Output	Select Bit
0	0	0
0	1	0
1	0	0
1	1	1

Boolean Statement: Select Bit = Write AND Comparator Output

Least Significant Select Bit for 3-to-1 MUX’s:

Read (Fetch)	Comparator Output	LS Select Bit
0	0	0
0	1	0
1	0	0
1	1	1

Boolean Statement: LS Select Bit = Read AND Comparator Output

Most Significant Select Bit for 3-to-1 MUX’s:

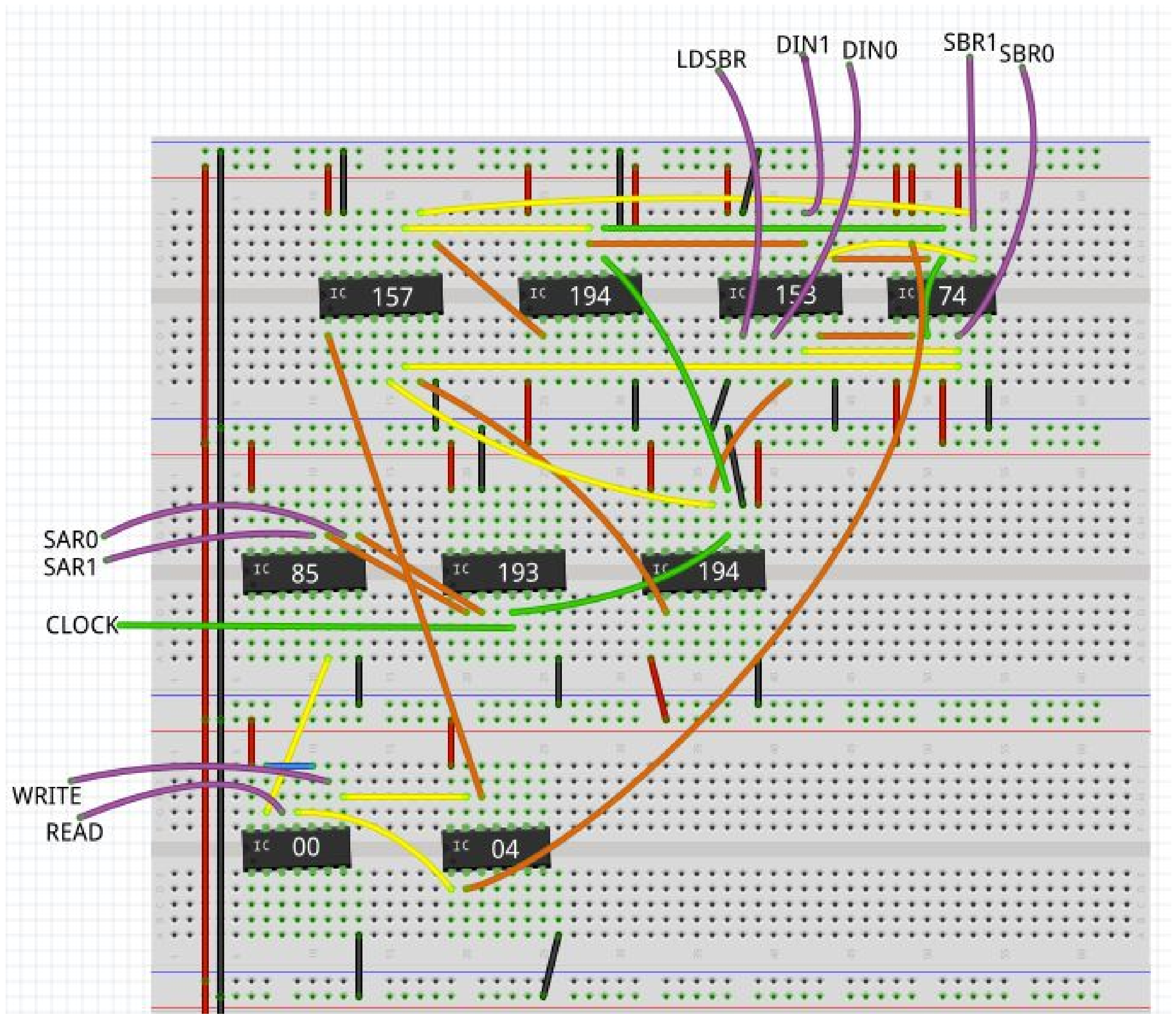
Load SBR	MS Select Bit
0	0
1	1

Boolean Statement: MS Select Bit = Load SBR

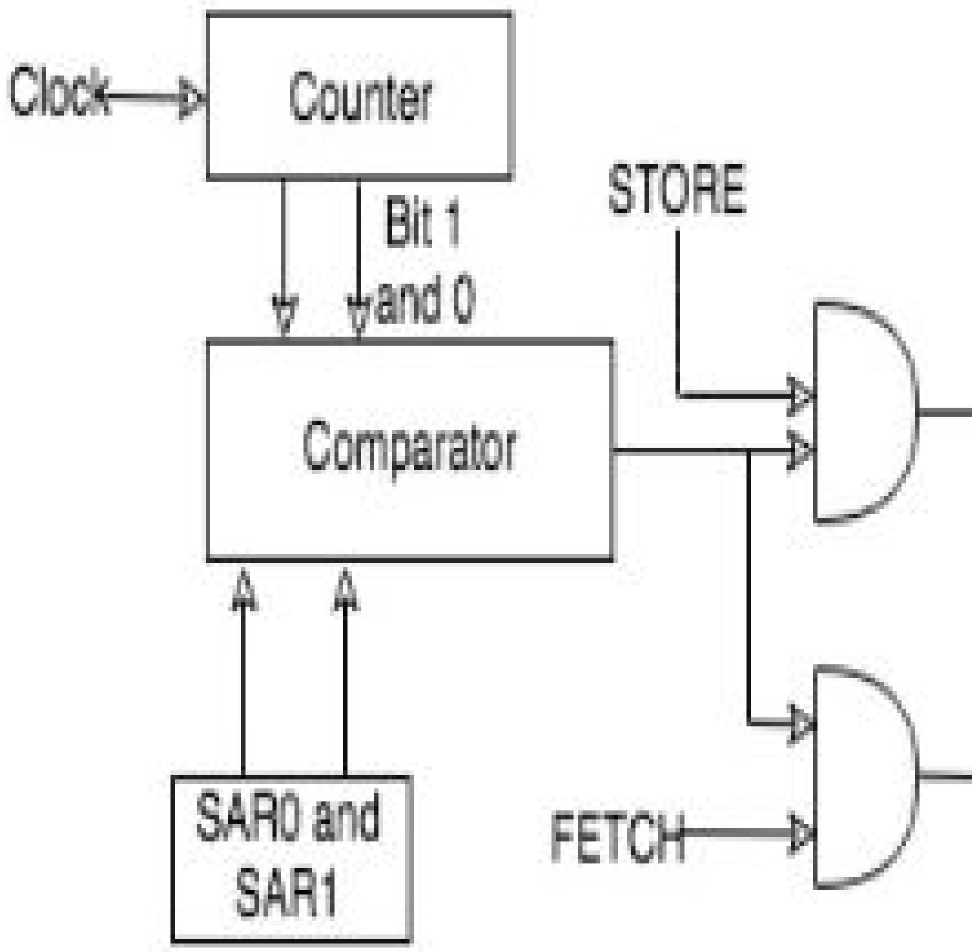
Karnaugh Maps

Since the logic was simple enough for the select bits, no Karnaugh Maps are necessary.

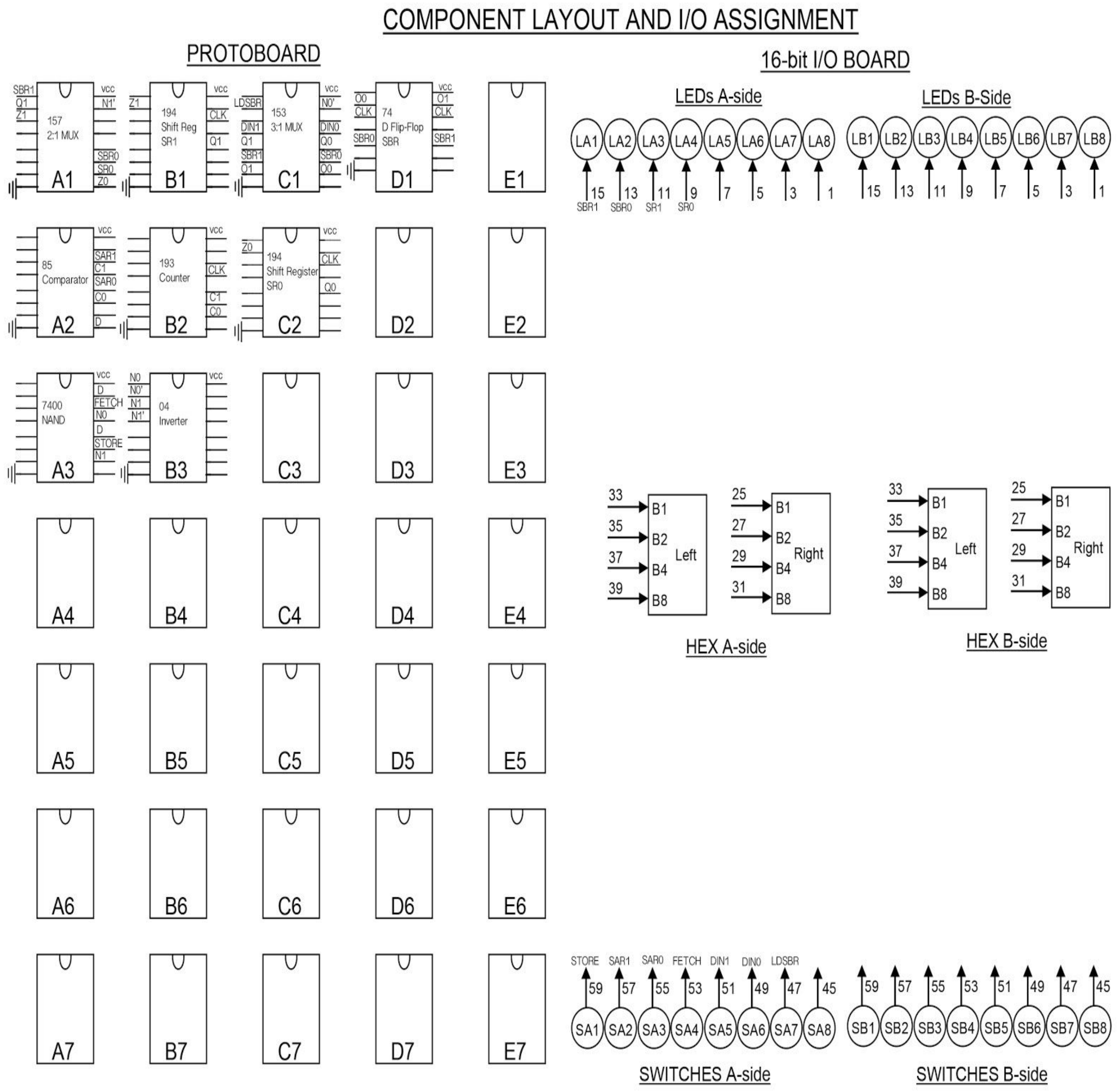
Board Layout



Circuit Diagram for Control Logic



Component Layout Sheet



COMPONENT LAYOUT AND I/O ASSIGNMENT
PROTOBOARD
16-bit I/O BOARD
A1 157 2:1 MUX: SBR1, Q1, Z1, vcc, N1', SBR0, SR0, Z0
B1 194 Shift Reg SR1: Z1, vcc, CLK, Q1
C1 153 3:1 MUX: LDSBR, DIN1, Q1, SBR1, O1, vcc, N0', DIN0, Q0, SBR0, O0
D1 74 D Flip-Flop SBR: O0, CLK, SBR0, vcc, O1, CLK, SBR1
A2 85 Comparator: vcc, SAR1, C1, SAR0, C0, D
B2 193 Counter: vcc, CLK, C1, C0
C2 194 Shift Register SR0: Z0, vcc, CLK, Q0
A3 7400 NAND: vcc, D, FETCH, N0, D, STORE, N1
B3 04 Inverter: N0, N0', N1, N1', vcc
E1 D2 E2 C3 D3 E3 A4 B4 C4 D4 E4 A5 B5 C5 D5 E5 A6 B6 C6 D6 E6 A7 B7 C7 D7 E7
LEDs A-side
LA1 LA2 LA3 LA4 LA5 LA6 LA7 LA8
15 SBR1, 13 SBR0, 11 SR1, 9 SR0, 7, 5, 3, 1
LEDs B-Side
LB1 LB2 LB3 LB4 LB5 LB6 LB7 LB8
15, 13, 11, 9, 7, 5, 3, 1
HEX A-side
Left: 33 B1, 35 B2, 37 B4, 39 B8
Right: 25 B1, 27 B2, 29 B4, 31 B8
HEX B-side
Left: 33 B1, 35 B2, 37 B4, 39 B8
Right: 25 B1, 27 B2, 29 B4, 31 B8
SWITCHES A-side
SA1 STORE 59, SA2 SAR1 57, SA3 SAR0 55, SA4 FETCH 53, SA5 DIN1 51, SA6 DIN0 49, SA7 LDSBR 47, SA8 45
SWITCHES B-side
SB1 59, SB2 57, SB3 55, SB4 53, SB5 51, SB6 49, SB7 47, SB8 45

Pre Lab Questions:

a: Q: Why is gating the clock a bad practice in digital design?
A: Firstly, clock gating can cause unwanted propagation delay in the circuit, as the clock signal usually goes directly into the components of the circuit. Also, race hazards may occur, which means that changes to inputs to a logic gate could arrive to the logic gate at different times, causing an unwanted output for a certain amount of time.

b: Q: Why does the clock need to be debounced to step through the circuit?
A: Debouncing a clock makes sure that the components aren't confused by the typical 'bounce' of the mechanical switch, as they can misinterpret this 'bounce' as a second input pulse. Debouncing ensures that only the signal from the first 'bounce' will be read.

Post Lab Questions:

1. Q: Make corrections to your pre-lab write up
A: We didn't realize that the preset input for the D flip flop chip had to be set to high for normal function. After changing this, our circuit functioned perfectly.

2. Q: What are the performance implications of your shift register memory as compared to a standard SRAM of the same size? What are the implications of the different counters and shift register chips, what was your reasoning in choosing the parts you did?
A: Our shift register memory requires more space but requires less time to access data than a standard SRAM of the same size. We chose our shift registers and counters for simplicity.

3. Q:What are the implications of the different counters and shift register chips? What was your reasoning in choosing the parts you did?
A: The lab required us to create a shift register memory for 4 2-bit words, so we simply utilized two SISO shift registers: one for each bit of each of the 4 2-bit words. The counter we used was a synchronous counter instead of using a ripple counter. The advantage of this was that the counter increased with each clock cycle rather than the previous output like the ripple counter. Although this would not affect the circuit heavily, at high clock speeds it made more sense to use something that was synchronous. As for the shift registers, using a "Serial In Serial Out" method was implemented as it was easiest to do and also what we were told in lecture to use. Though it is slower than using parallel methods, it allows for a more compact circuit that does not require multiple data IN switches our OUT leds.

Conclusion:

In summary, this lab went extremely well considering the type of errors we encountered along the way. Twice, touching resistors impeded our circuit from working correctly, but as we organized our breadboard to look cleaner, we fixed these issues. On the day of the lab, we made two errors: we attempted to test the function of the circuit with the switchbox without actually plugging into the switchbox, and we also neglected to connect the preset input of the D flip flop chip to high. In addition, we forgot to set the function generator to Hi-Z mode for the clock, giving us errors in our circuit output as well as puzzling images and numbers on the Oscilloscope. The TA (David) fixed this for us and we had no problems after this. We got all five points for the demo after fixing these small issues.

Lab 2 taught us a whole lot about the process of understanding how a circuit has to function before any part of it can be built. Although we were lucky to have every part function in the end, we did not do as much intermediate testing as we should have, and we plan to change this for future labs.