# ECE 385

Fall 2020

Experiment 2

# Data Storage

**Zhicong Fan / Xin Jin**
**AB2/Online**
**Yucheng Liang**

# Introduction:

In this lab, we designed a 2 bits 4 words shift register. Within the circuit, we implemented three operations: FETCH, STORE and LDSBR, which achieved the functionalities of read and write to register. Specifically, we used switches to control the input signals and used LEDs to display the outputs.

# Memory Circuit Operation:

- Addressing:
  We used a control unit to specify the kind of operations, and a counter to locate the relative address in the shift register. Whenever a read or write operation was initiated (switch flipped), it would have to wait the counter matched the address specified by SAR. Since the lock in counter and shift register updated simultaneously, the counter was used to locate the relative address in the shift register. In addition, to avoid timing conflicts, we would use LDSBR operation to first load the inputs DIN0 and DIN1 into SBR0 and SBR1 as buffer.
- Write:
  When we wanted to do a "write" operation, we had to first flip Din1 and Din0 for the data we want to write into the shift register. After that, we need to specify the address of the data by flipping SAR1 and SAR0. Then a LDSBR function is required to load our input data into SBR. After that, we just need to flip STORE and then wait for the comparator to give us a signal if the counter matches with the address we want. It would take at most 4 cycles to load the data into shift register. At each cycle, the counter would increment by one bit and the data in the given address would remain unchanged until until it matched the SAR.
- Read:
  SAR1 and SAR0 would need to be firstly flipped if we wanted to read the data at a specific address. And then we should flip FETCH to initiate the read operation. Then it would wait the counter to match with SAR1 and SAR0. Finally, the data would be loaded into SBR, which is connected to the LEDs, after at most 4 cycles. The data in SBR will remain unchanged until the counter matched the SAR

# Description and Block Diagram:

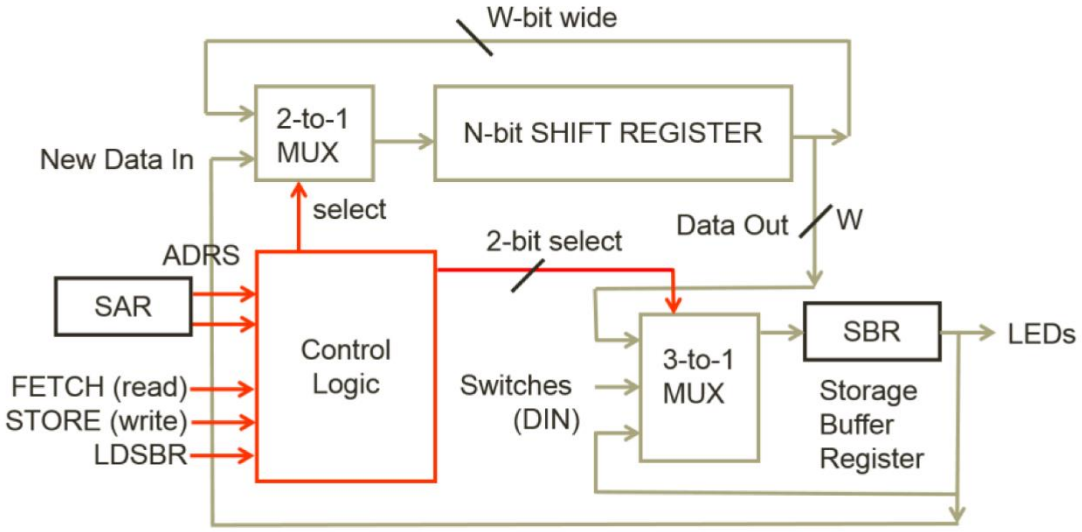
**High-level Description:**

In General, we used switches as the input to the circuit, a control unit to process the input signals and give the correct outputs to the rest of the circuit, two shift registers to store the 2bits 4 words data, two 2-to-1 MUXs to control the contents in the shift registers, two 3-to-1 MUXs (we used 4-to-1 instead) to control the content in SBRs, and two D flip-flops to serve as the buffer registers SBR. In addition, few LEDs and 330 Ohm resistors are used as the outputs (SBRs).

We used FPGA slide switches to control the circuit. The switches we assigned are: 1. SAR0, 2. SAR1, 3. DIN0, 4. DIN1, 5. FETCH, 6. STORE, 7. LDSBR. They will be served as inputs to the circuit. When an operation takes place, the corresponding switches would be flipped. All the inputs will be sent to the control unit to make sure the circuit does the right operation. The detail of the

control unit will be explained in corresponding section below. The outputs of the control unit will be the select bits to 2-to-1 MUXs and 3-to-1 MUXs. 2-to-1 MUXs determine whether update the content in the shift registers or not. Whereas the 3-to-1 MUXs determine what data should be stored in SBR for the current operation.

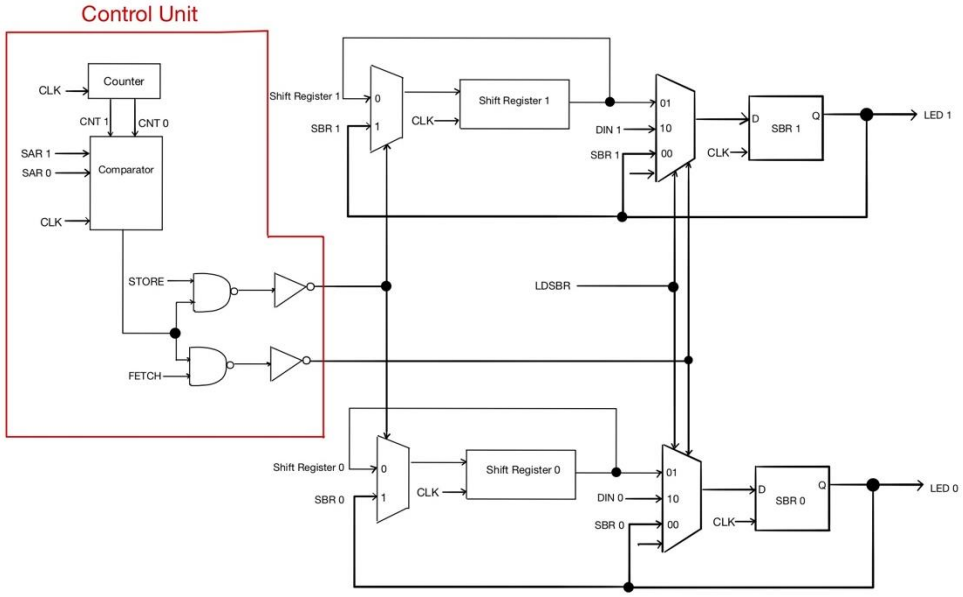
Below is a high-level block diagram of the circuit. A more specific version can be found in the latter section of this report.




# Control Unit:

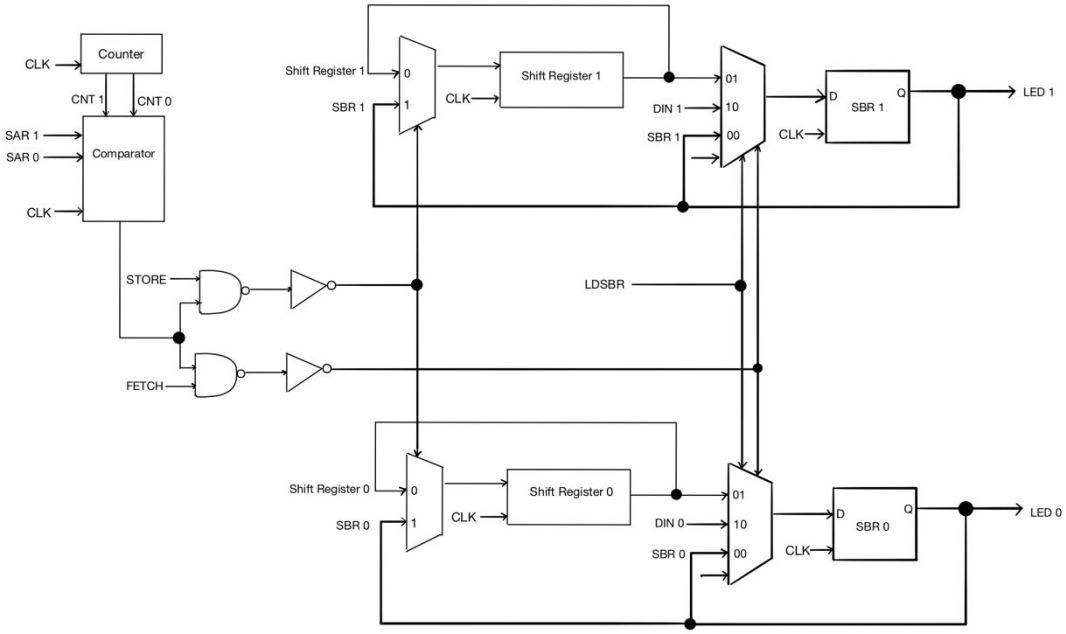
**Description:**

In the Control Logic we have a counter, a comparator, two NAND gates and two NOT gates.
The comparator takes SAR0, SAR1, CNT0, CNT1 (CNT0, CNT1 are the outputs of the counter) as inputs and output 1 if SAR0 is the same as CNT0 as well as SAR1 and CNT1. As explained in previous section, the 2 bits outputs of the counter specify the relative address in the two shift registers. If the relative addresses match with the SARs, the comparator will output a 1 (high). This output will AND (implemented by a NAND and a NOT gate) with either FETCH or STORE to determine if the current relative address is read or write ready. The outputs of the control units will proceed to the 2-to-1 MUXs and 3-to-1 MUXs to determine the contents in the shift registers and the D flip-flop (SBR). The control unit will control two branches in the circuit simultaneously.




# Design steps:

| FETCH | STORE | LDSBR | Comparator output | 2-to-1 MUX SELECT | 4-to-1 MUX SELECT |
|---|---|---|---|---|---|
| 0 | 0 | 0 | x | 0 | 00 |
| 1 | 0 | 0 | 1 | 0 | 01 |
| 0 | 1 | 0 | 1 | 1 | 00 |
| 0 | 0 | 1 | x | 0 | 10 |
| x | x | x | 0 | 0 | 00 |

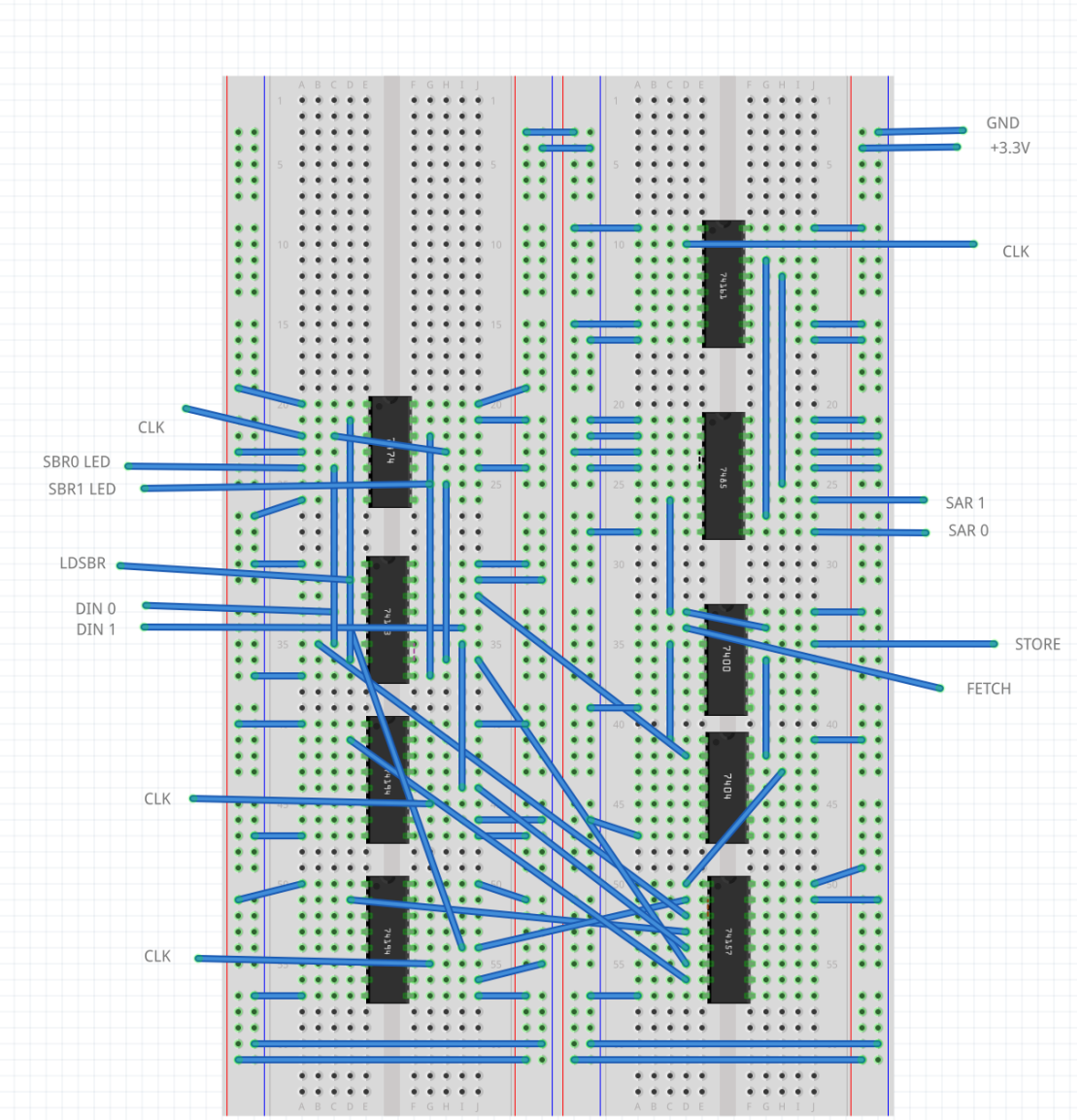


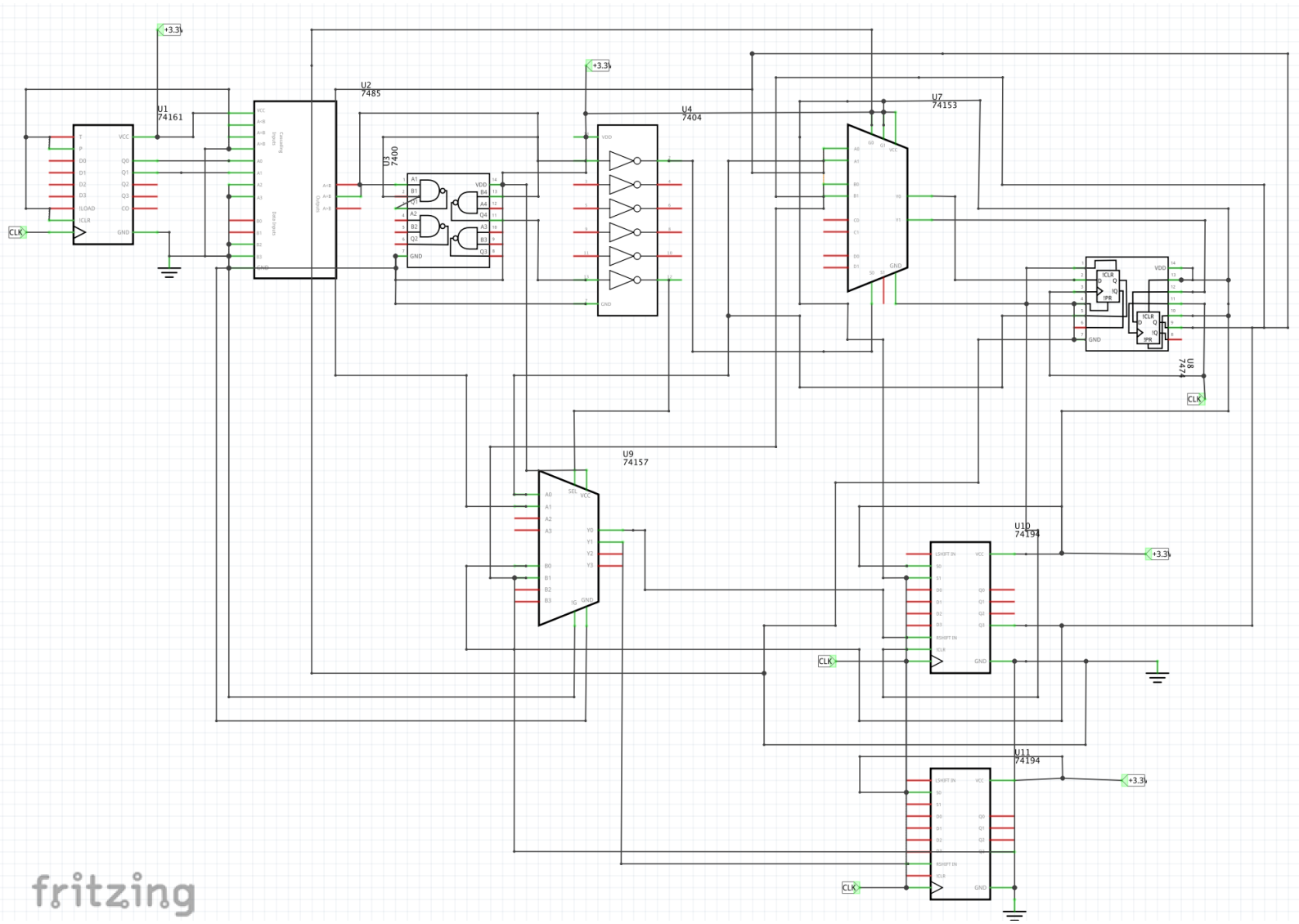
**2-to-1 MUX:**

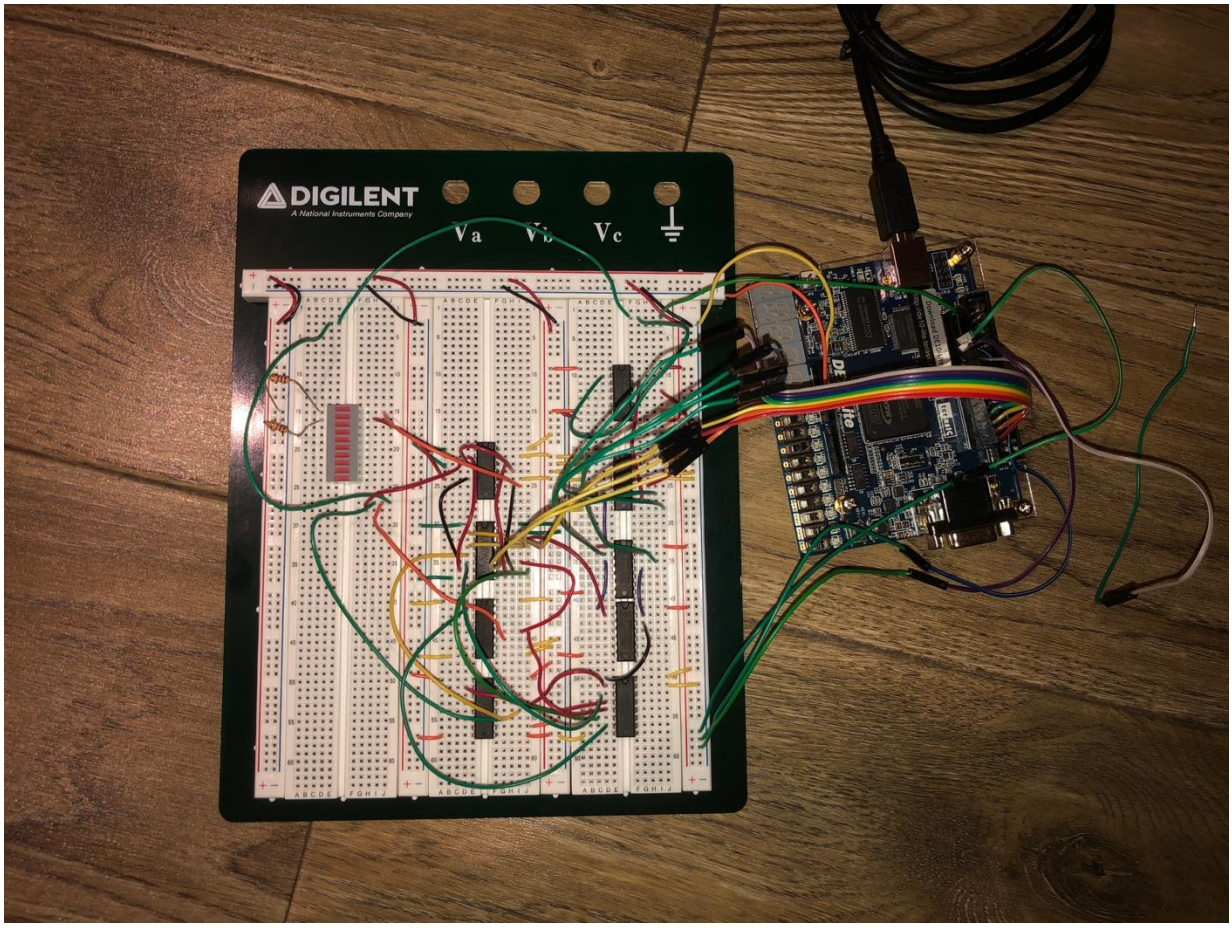
If the select pin for 2-to-1 MUX is 0, for operations such as STORE and “do nothing”, the content of the SBR will remain unchanged by feeding the content of the data back to itself. 4-to-1 MUX select pin will only be 1 if the STORE operation is initiated and the counter matches with relative address in the shift register. Then the content of the shift register at the specific relative address will be updated by the current data in SBR. The content of the shift register will remain unchanged if the current relative address does not match with SAR or it is at the “do nothing” operation.

**4-to-1 MUX:**

If the select pin for 4-to-1 MUX is 00, for operations such as FETCH, LDSBR, and “do nothing”, the content of the shift registers will remain unchanged by feeding the content of the data back to the shift registers. 4-to-1 MUX select pin will only be 01 if the FETCH operation is initiated and the counter matches with relative address in the shift register. Then the content of the shift register at the specific relative address will be loaded into SBR. 4-to-1 MUX select pin will only be 10 if the LDSBR operation is initiated. Then the content of in the DIN be loaded into SBR. At any other states, 4-to-1 MUX select pin will be 00, which will make retain the content in SBR.




U1
74161
U2
7485
U3
7400
U4
7404
U7
74153
U8
7474
U9
74157
U10
74194
U11
74194
+3.3v
CLK
fritzing




## Bugs encountered:

When choosing the feedback input for shift register in the 2-to-1 MUX, I at first chose Qa output from right shift register, which was the most recent update input. Since Qa will be updated by the input and Qb, Qc, and Qd would take the previous input from Qa, Qb, and Qc. When Qa was updated at the current cycle, Qb, Qc, and Qd would all be updated by the same data in Qa, which returned erroneous outputs. The issue was solve by choosing Qd as the feedback input for the 2-to-1 MUX, since when Qa been updated at the next cycle, Qb held the current data of Qa in the current cycle.

## Prelab Questions:

***Q: The clock must run continuously – do not gate the clock (this is bad practice in digital design, why?)***

**A:** Since all chips share the same clock, we would want to ensure all of the receive the same clock edge without any delay or static hazard. If we gate the clock, the delay is prolonged which would cause a static hazard that we explored in the last lab, which results in different chips may receive different clock edges.

***Q: Only the clock input needs to be de-bounced in order to step through your circuit (why?).***

**A:** Since all chips share the same clock and the number of clock cycles directly determine the output of the circuit, it is essential to have the clock to give a clear clock edge one at a time. Take 4 bits shift register in this lab as an example, if we update the content in Qa, we would expect Qb having the content of Qa in the next clock cycle. However, if the clock bounced several times during one clock cycle, the register would have shifted the content for several times in just one bounced clock cycle, and we wouldn't not be able to find the expected content in Qb. Other switches such as SAR, DIN, FETCH, LDSBR, and STORE do not need to be debounced because we only care about the final state of those inputs. The contact bounce erroneous middle outputs would not affect the final output of the circuit, since FETCH, STORE, and LDSBR operations would either stay the same or proceed the operation, and it does not change the final result if they bounce multiple times during one switch.

# Post-lab Questions:

***Q: What are the performance implications of your shift register memory as compared to a standard SRAM of the same size?***

**A**: An SRAM is static and thus we are able to directly read or store data from or in a SRAM and not to wait for a counter to match with the address we want to store at. The reason why a counter is needed when we are using shift register is that the voltage in the shift register would continuously drop so that the shift register has to loop around to make sure the voltage is high so that it could store and data and not to lose it. That's why it is undetermined in the states at each time so that we need a counter and a comparator to keep track of it. Since data can only be store one bit at a time, it is relatively slow for SISO shift register to store large data since it has to wait for many clock cycles.

***Q: What are the implications of the different counters and shift register chips, what was your reasoning in choosing the parts you did?***

**A:** Both ripple counters and synchronous counters are mentioned during the lecture. We chose to use synchronous counter because it has a perplexed feature that could preclude glitches in its outputs, which means it could provide impervious monotonous signal alongside the clock which a ripple counter could not. So we chose to use a synchronous counter. For shift register, the shift register has the left serial input that we needed, which works perfectly for the purpose of this lab, so that we chose the 74194 chips.

# Conclusion:

In this lab, we designed a 2 bits 4 words storage unit by using two 4 bits shift registers. The circuit needs to be carefully designed in order to achieve the desired functionalities. We have explored hardware debugging experiences and simple usage of FPGA. We have also realized that the level of complexity scaled up very quickly as the we want to implement more functionalities.