ECE 385

Spring 2020

Experiment 2

Data Storage

Michael Faitz and Zohair Ahmed

Section ABJ: Friday 2:00-4:50

Yuming Wu and Lian Yu

Introduction

The function of this circuit is to create an elementary form of data storage. It performs operations based on inputs given by the user via switches. It allows for data import and export from specific addresses in the memory. The user can load specific data into select addresses as well. Our memory contains 4 words with a bit width of 2.

Operation of the Memory Circuit

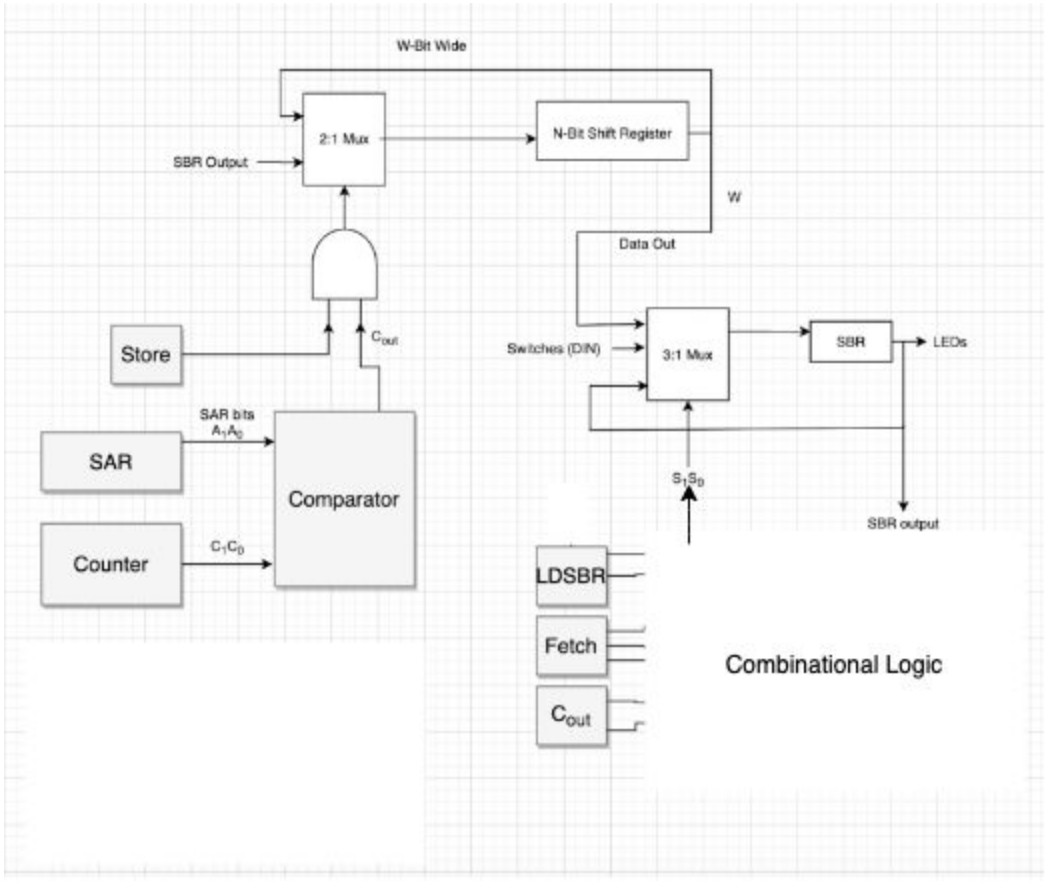
a. The addressing is implemented not by concrete address values, but by comparing the value contained in the SAR to the counter value to provide an effective address system. The counter is incremented every time the register shifts, and when these bits match the data in the SAR, the write and read operations are triggered and are allowed to interact with the register.
b. A write operation takes the data inside the SBR and pushes it to the shift register. We flip the store switch to affect the calculation of the select bit in the 2:1 and 3:1 multiplexers. The select bit in the 2:1 is an AND gate of the store bit and the output of the comparator when the counter bits match the SAR bits. The select bits in the 3:1 multiplexer also depend on the store bit, which affects what input goes to the SBR. When the write operation is called, the comparator will check that the counter bits match the SAR bits, and when they do will allow the 2:1 to push the data in the SBR to the leftmost bit input to the shift register. If the bits don't match, the register shifts until they do with no data being processed.
c. A read operation takes the data in the address given by the SAR and puts it into the SBR so it can be read. The fetch switch is set high, which affects calculation of the 3:1 select bits and allows the output of the correct address to be loaded into the SBR. When the read operation is called, the comparator will check that the counter bits match the SAR bits, and when they do they will allow the output (right most data) of the shift register to be input to the SBR. If the bits don't match, the register shifts until they do with no data being processed.

Written Description and Block Diagram

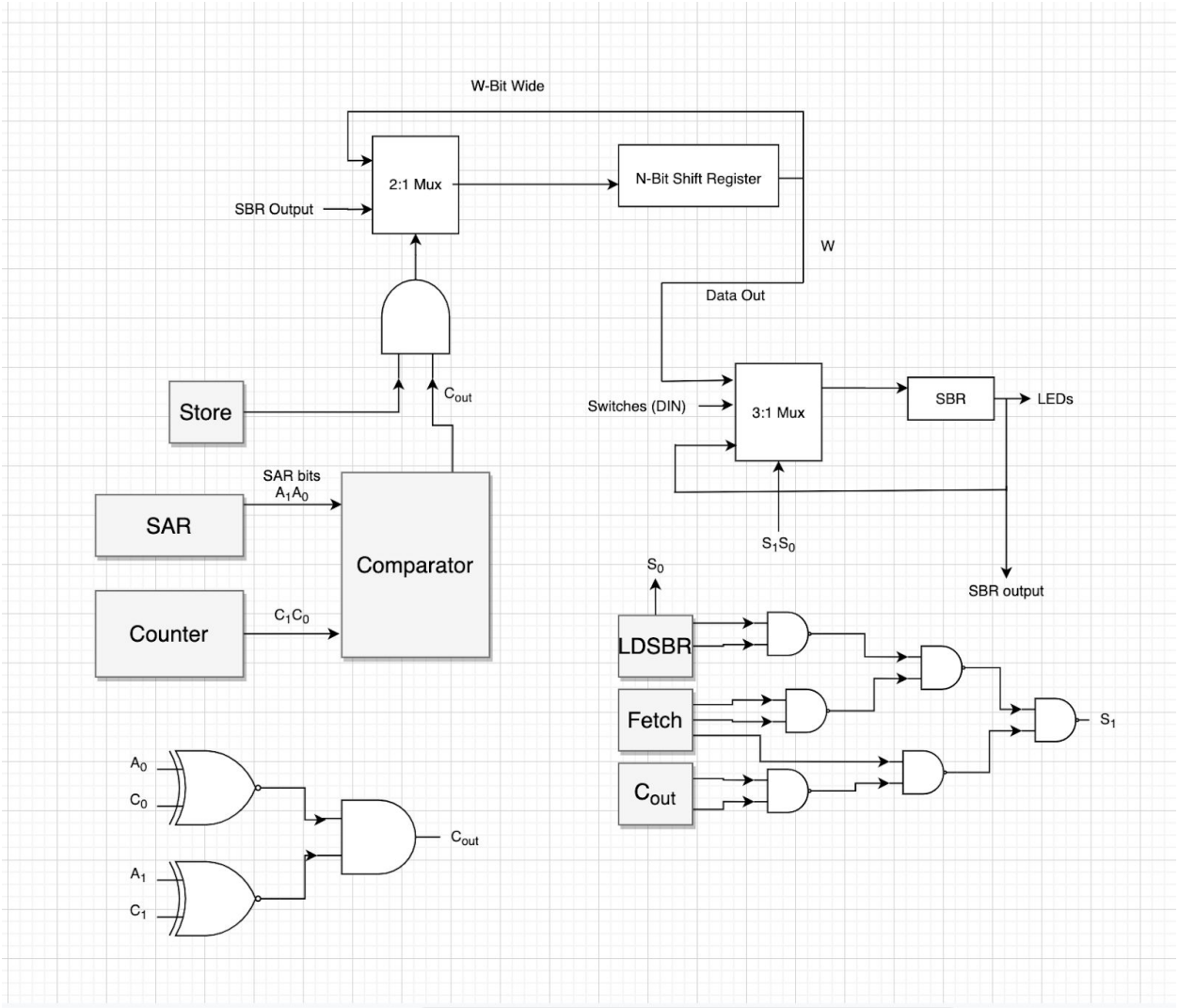
In order to keep track of memory addresses and avoid unwanted reading and writing, we needed a counter and a comparator. These compared the current state of the shift register to a given count of shifts and allowed us the ability to compare this count to the data in the SAR. From here, we merely confirmed they were equal before going ahead with the fetch or store operations.

The 3:1 Mux was used to make sure our SBR contained the data we needed for a particular operation. We used combinational logic to create select bits from the switches and the comparator output, which allowed us to go ahead with read, write, and load operations.

The 2:1 Mux was used to either shift the data lost by the shift register from shifting back in, or to store new data in. This was done by making sure the store instruction was active, as well as confirming that the address we wanted to write was present.



Control Unit



The circuit ties the switch inputs and counter bits through combinational logic before using them as inputs to MUXes, the SBR (D flip-flops) and the shift registers. The registers, counters, and flip flops are all connected to the same clock layout, making the circuit synchronous. This way, we can avoid designing the circuit to halt certain elements and instead focus on allowing each element to continue running until we reach states that we wish to act upon.

Design Steps and Detailed Circuit Schematic

a.

Truth Table					
compared	Load (L)	Fetch (F)	Stored (S)	Left 3:1 Select Bit	Right 3:1 Select Bit
0	0	0	0	1	0
0	0	0	1	1	0
0	0	1	0	1	0
0	0	1	1	X	X
0	1	0	0	0	1
0	1	0	1	X	X
0	1	1	0	X	X
0	1	1	1	X	X
1	0	0	0	1	1
1	0	0	1	1	X
1	0	1	0	0	X
1	0	1	1	X	X
1	1	0	0	0	0
1	1	0	1	X	0
1	1	1	0	X	0
1	1	1	1	X	X

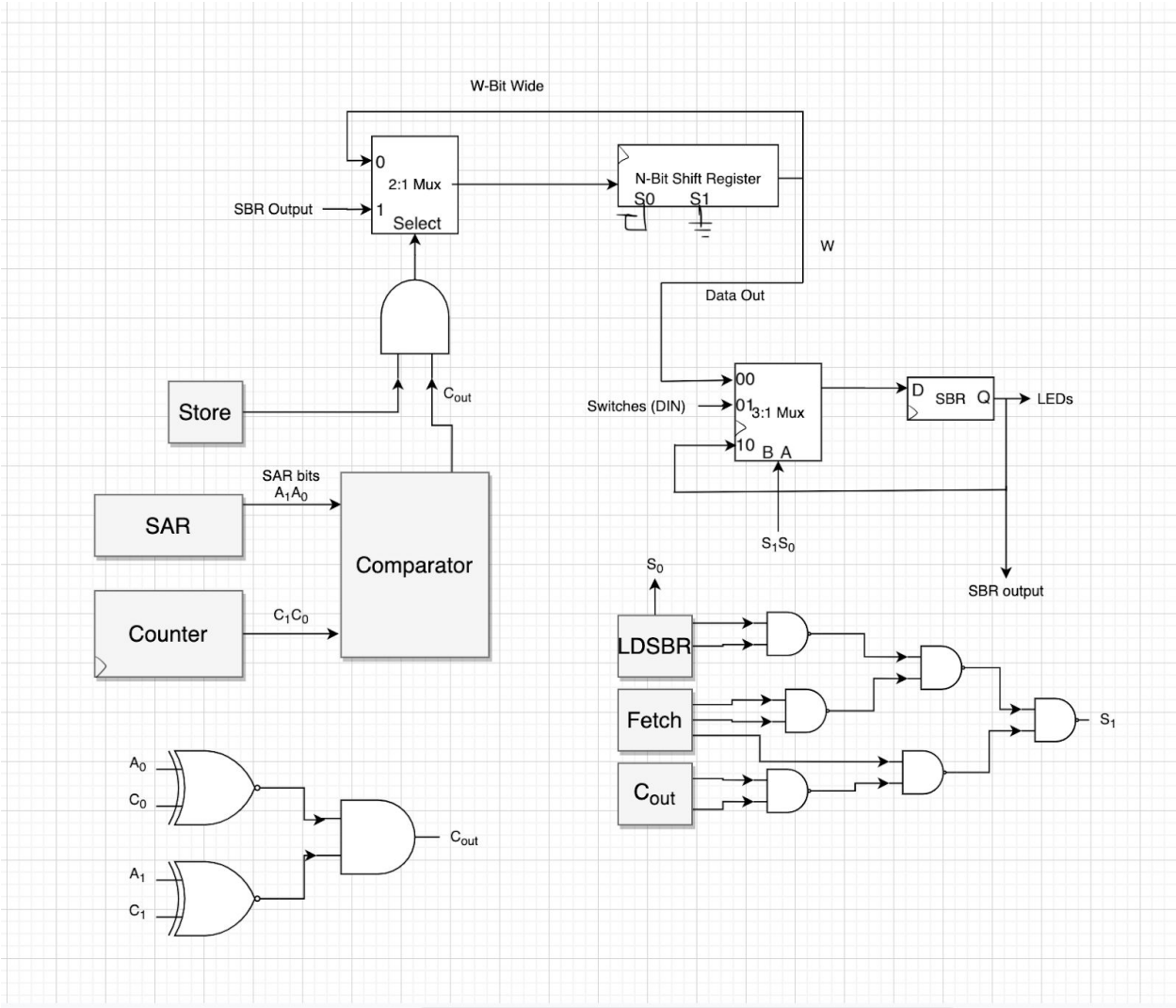
3:1 Mux Left Bit K-map				
	Fetch (F)		Store (S)	
compared	1	1	X	1
	0	X	X	X
Load (L)	0	X	X	X
	1	1	X	0

3:1 Mux Right Bit K-map				
	Fetch (F)		Store (S)	
compared	0	0	X	0
	1	X	X	X
Load (L)	1	X	X	X
	0	0	X	0

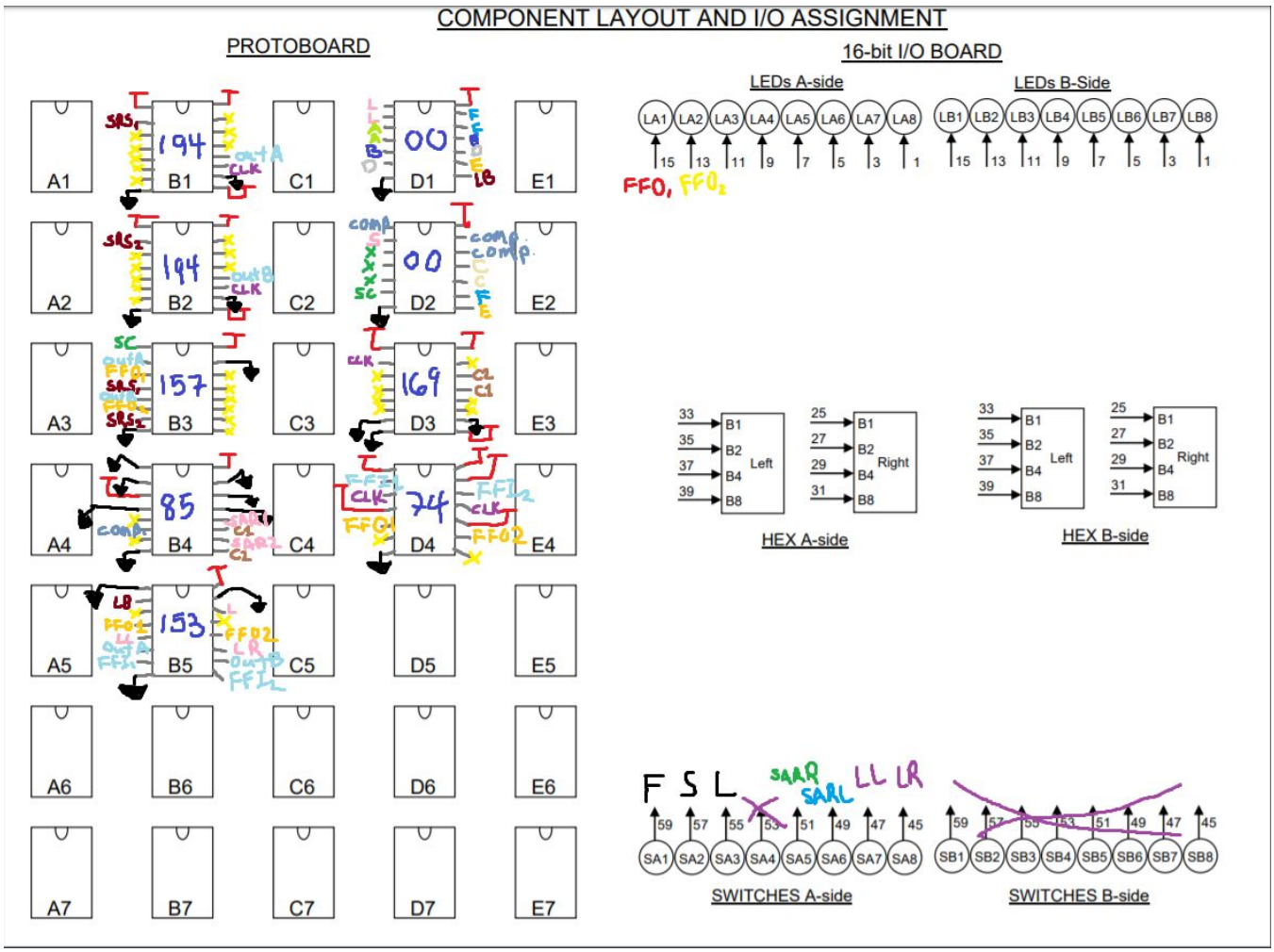
c.

We used synchronous versions of every component so as to reduce confusion and simplify the design and testing of the circuit. We ignored the ripple carry output on the counter, and generally avoided using anything involving overflow.

Schematic Diagram



Component Layout Sheet



Bugs Encountered and Corrections

One major issue we ran into was having different grounds throughout various components which destroyed our 2:1 multiplexer and gave us several issues debugging. We fixed this by connecting everything to one ground, specifically the one on the lab bench. We ran into no other issues while designing or testing.

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: Since our implementation of RAM requires data to be written in specific addresses within the memory it will generally be slower than SRAM which quickly moves data in and out of

operation without storing and reading from select input addresses. The benefit of the RAM we implemented over the SRAM, however, is the ability to more easily track the data present within the memory due to each address being clearly loaded into.

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 benefit of our decision to implement synchronous counters and shift registers was that we could design the entire layout using a single clock which can be uniformly applied to each component and avoid delay bugs like those observed within Experiment 1. A possible tradeoff with the synchronous counter is that it is not easily extendable if the bit size for each word was increased past 16 bits, however, for the given counter we only used 2 of the 4 available bits so there was effectively no negative effect from the synchronous device. Since we wanted the shift register to shift each time the clock was cycled, as instructed within the lecture slides, there is effectively little tradeoff by choosing a synchronous shift register like 194. Choosing a different non-synchronous shift register would make the clocking significantly more complicated and extend the difficulty of the design without much benefit.

Conclusion

This lab taught us to be more careful where we ground our circuit, as well as mapping connections in our circuit. We enjoyed using a synchronous design throughout our circuit and watching the entire system run one cycle at a time. This was a very rewarding lab, as we got to see our implementation of something useful run in real time.