ECE 385

Spring 2020

Experiment # 2

Data Storage

Megh Shah, Ishaan Patel
Lab Section: ABC, Tuesday 11:30 AM
TA: Gene Shiue, David Zhang

Introduction:

Random access memory, RAM, is essentially a storage device that consists of unique addresses which can fetch or store data. The essential point of a storage unit, such as RAM, is to safely hold data values for future use or until the unit is told to overwrite it or clear it. In this lab, we learnt how a RAM chip operates by understanding and implementing the use of clock signals, storage buffer registers(D-flip flops), "storage address registers"(switches in our case), and shift registers. We designed the RAM chip to store data up to 2-bits in 4 unique memory addresses, also known as words. By coherently utilizing these devices and their concepts, we were able to gain a coherent understanding of how RAM chips operate and their importance in today's technological world.

Operation of Memory Circuits:

A. In our circuit, the addressing is directly controlled by two input switches: SAR1, which represents the leftmost bit of the 2-bit wide address, and SAR0, which represents the rightmost bit. The signals of both of these switches, which can either be a '1' or a '0', indicate which of the 4 unique memory addresses will be accessed in order to read or write data. A read is committed in the circuit when the desired memory address that the user wants to read from is set using the SAR switches and the FETCH switch is flipped on. A write is committed when the user loads in data to the circuit, sets the address they would like to store it in using the SAR switches, and then flipping the STORE switch on.

B. To perform a successful write operation, there are several steps that one must follow in a certain order. First the user must load in the data that will be written. The 2-bit wide data can be chosen by setting the data switches, DIN1 which controls the leftmost bit of the data and DIN0 which controls the rightmost bit, to the desired value, and flipping the LDSBR switch on. By doing so, the data is stored into the circuit and is held within the flip-flop. All switches must be turned off after. Next, the SAR switches must be set to the desired memory address the user wants the data to be stored into and the STORE switch, which indicates that the process is a write process, should be flipped on. Through each clock cycle, the data flows to the 4-to-1 MUX and is stored in the D flip-flop for as long as the clock is running and no other inputs are added.

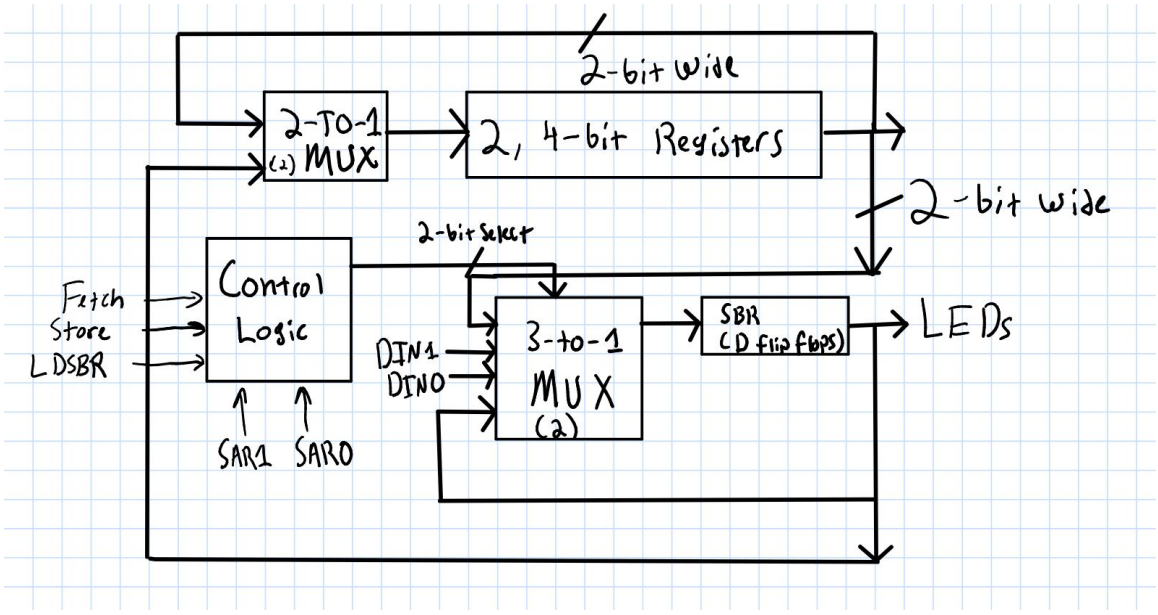
C. To perform a successful read operation, first the user must set the SAR switches to the memory address they want to read from. Afterwards, they must flip on the FETCH switch which will fetch the data from that memory address, which is being stored in the D flip-flop, and display it on the LEDs. For this operation, the flip-flop holding the data will hold whichever values are stored in the memory that we are fetching from no matter the clock signal.

Written description and block diagram of memory circuit implementation

A. High-level Description

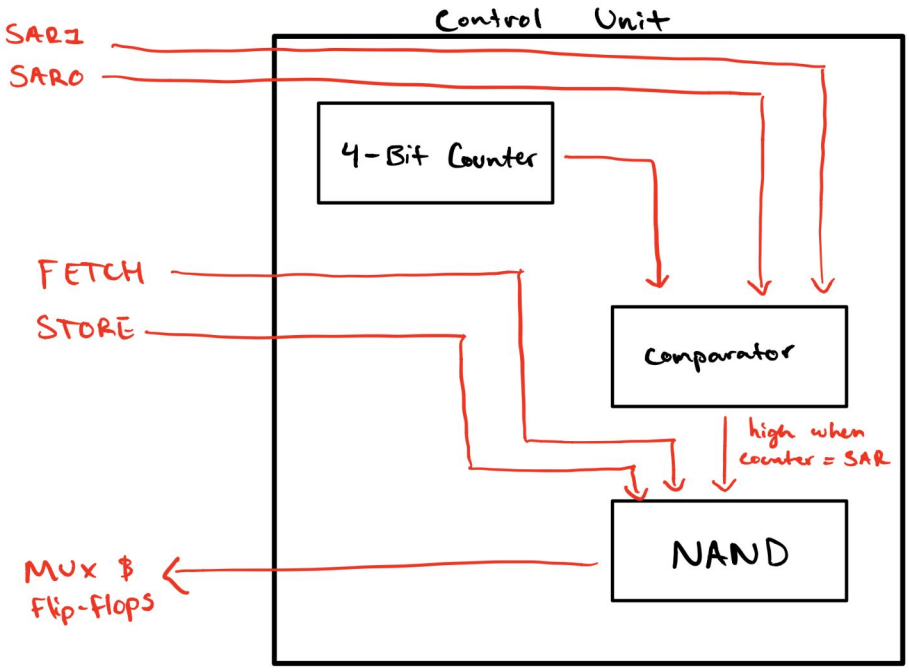
a. To perform the operations described in the pre-lab we need to implement certain components. For the Control logic we used a NAND chip to AND fetch and store with the output of a comparator. The comparator is comparing SAR and the counter. This is all that is needed for the control logic. To actually store 2 bits in 4 addresses we need two 4-bit shift registers. Each register has a 2-to-1 mux in its shift right input. One thing that goes into each mux is a D flip flop. The D flip flops each are an output to a 3-to1 mux that takes the register, DIN and the flip flop's output as an input.

b.



Control Unit:

Our control unit consisted of a 4-bit counter, a comparator, and NAND gates. The control unit operates by having the 4-bit counter be utilized as a 2-bit counter. Whenever the 3rd bit of the counter would become a "1", the counter would reset itself. Therefore, the counter acted as a 2-bit counter. The counter would keep counting and its outputs would be fed to the comparator along with the inputs of the SAR switches. When the output of the counter and the inputs of the SAR switches were equal, the comparator would recognize that and send out a high signal from the "A=B output". This signal would be fed into a NAND gate where the NAND gate would NAND this signal with both the FETCH and STORE signals to determine whether or not the user wants to fetch or store data from their desired memory location. This output would then be inverted as we want the inverse of this signal. This signal is then passed onto the MUX and then the flip-flops which takes care of the actual memory reading and writing.



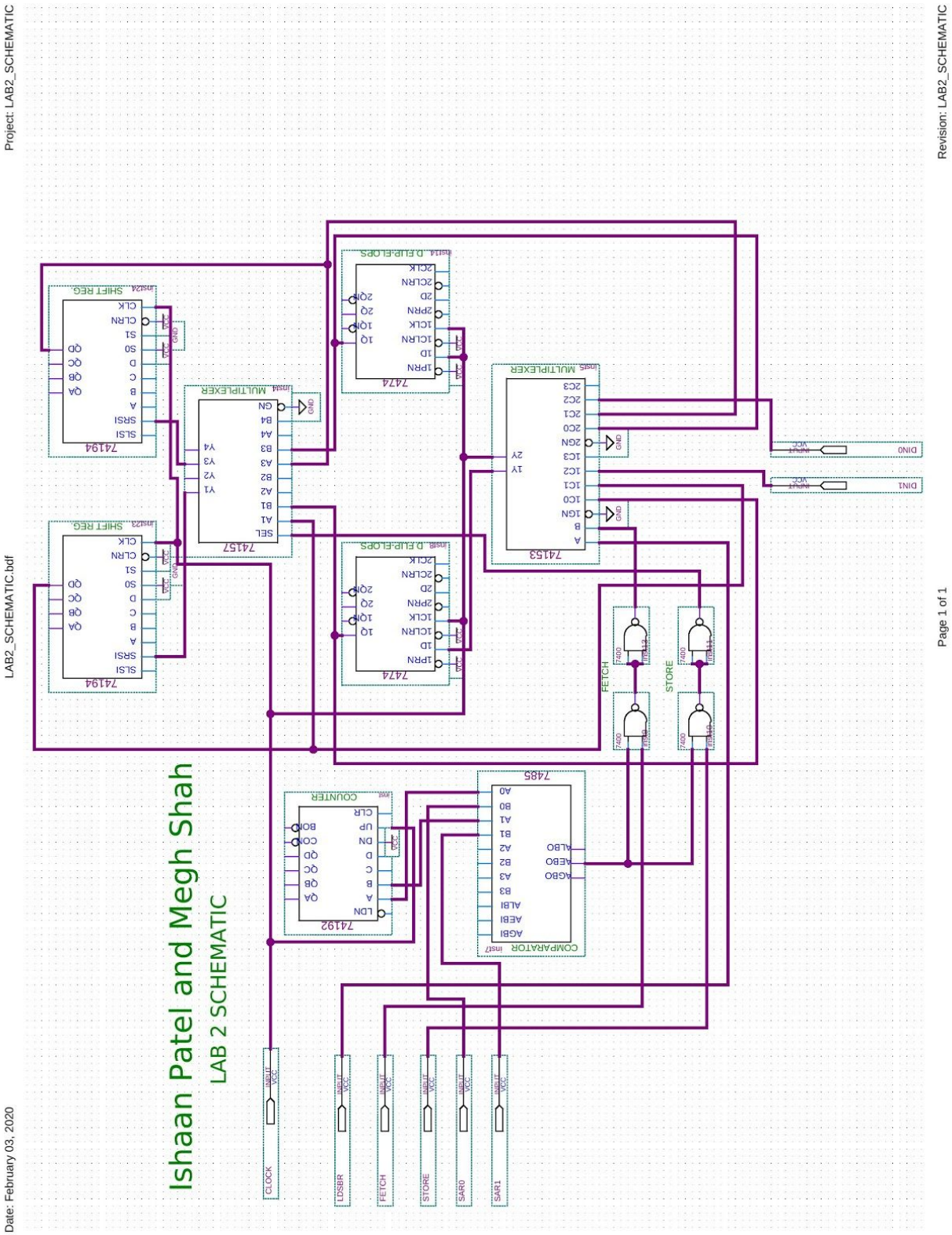
Design steps taken and detailed circuit schematic:

A. We didn't have to use any k-maps or truth tables during design. This lab was relatively simple in design complexity.

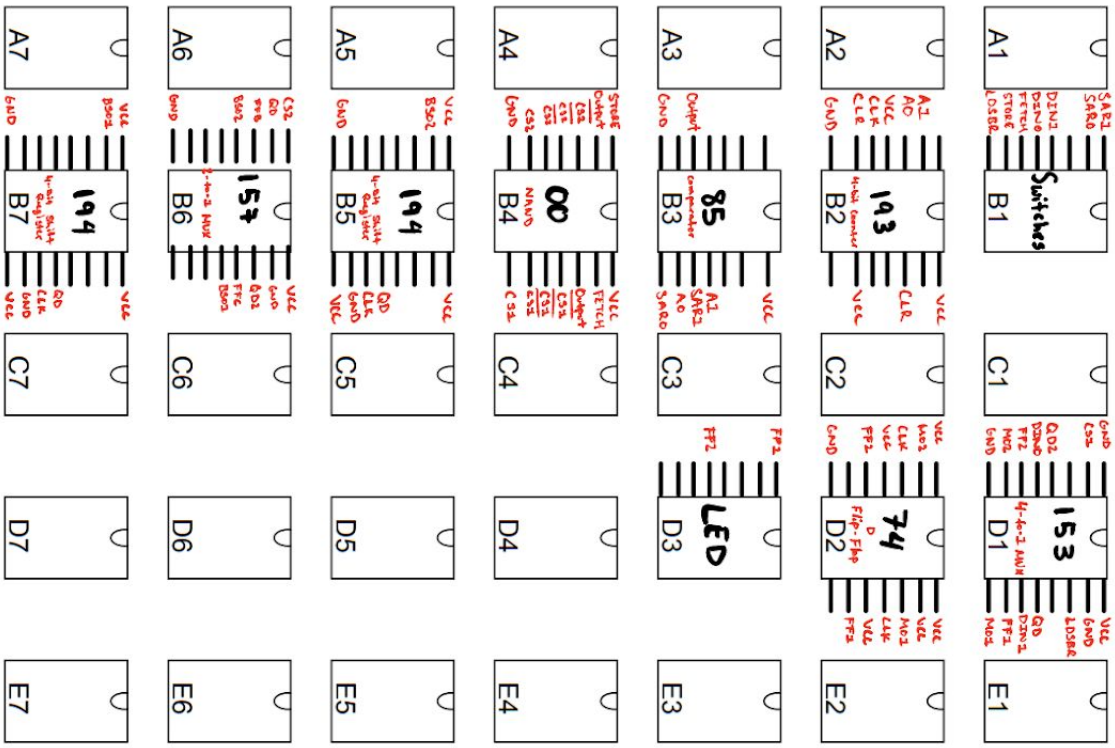
B. Design Considerations

   a. The first consideration that we took into account was which flip flop to use. We decided to use the D flip flop since it is very simple and has a single input, compared to the JK flip flop which has two. Next, we decided to use the synchronous 4-bit counter instead of the ripple counter since we wanted to ensure there was no delay in data processing. This delay could have added up and might have added unusual artifacts to the design. The last significant design consideration was whether to use a 2 to 1 mux or a 4 to 1 mux as input to the d flip flop(s). After some time we decided to use the 3 to 1 mux since we could put LDBSR on one input and FETCH on the other. This reduces the complexity of our logical circuit.

D. Detailed Circuit Schematic



Component Layout Sheet:



Description of all bugs encountered, and corrective measures taken:
The only problem that we encountered was a pin on a TTL chip coming off and not completing the circuit. After we replaced the chip, the circuit worked first try. Our success is due to careful precautions and 3-4 hours of preparation. For a good 3 hours we tested each IC to ensure they work as intended and then after that we began to carefully wire the circuit together. Every time we placed a wire we would double check with our wiring diagram to make sure it was in the right pin. This goes to show that with diligence, you can significantly reduce your debug time and in our case have only a small problem.

Conclusion:

A. This lab taught us that memory isn't simply just made from capacitors like DRAM or from a two-inverter loop like SRAM. It can also be made from shift registers, given additional control logic such as a comparator and a counter. The concept of data storage has a lot of flexibility and ideas like memory addressing and addressability can all be done using a shift register and a few additional components.
B. What are the performance implications of your shift register memory as compared to a standard SRAM of the same size?
    a. A standard SRAM would be able to immediately access the data (Ignoring propagation delay) since it utilizes a decoder for memory addressing. Our shift register memory on the other hand has to wait for the correct memory address from the counter which would take longer to access.

    What are the implications of the different counters and shift register chips, what was your reasoning in choosing the parts you did?
    b. We chose the SN74192 4-bit up/down counter since it featured a data clear pin that would allow us to limit the addresses to 00,01,10 and 11. The reason why we chose the SN74194 bi-drectional shift register was also because it features a clear pin that would allow us to quickly clear the data if needed to be.

Pre-Lab Questions:

A. Pre-Lab Questions
    a. Do not gate the clock (This is bad practice in digital design, why?)
        i. Putting a gate on the clock can mess up other parts of your circuit without knowing about it.
    b. Only the clock input needs to be debounced to step through your circuit (Why?)
        i. The only reason why the clock needs to be debounced is because the circuits react the the "bounced" effect instantaneously and will still "run" while the clock is bouncing. Switches don't usually need to since the circuit won't read the switches' value instantaneously.