Lab 2 Report

ECE 385

Friday (ABB)

11:00AM-1:50PM

Data Storage

Ritvik Avancha and Steven Phan

(rra2 and sphan5)

Table of Contents

Lab 2 Report

Introduction

After the introductory lab from last week, this week's lab asked for something slightly more complex. The circuit designed in this lab implemented RAM (4x2, four addresses each able to hold two bits) using MUXs with select logic, two 4-by-1 shift registers, counter and comparator, and two D-flip-flops to act as buffer registers. In order to load into buffers, read, and store in and from certain addresses, a total of 7 external switches were used: FETCH, STORE, LDSBR, SAR[0] and SAR[1], and D_{in}[0] and D_{in}[1].

Operation of the Memory Circuit

Addressing is implemented by having a counter cycle through 00, 01, 10, and 11. Each clock cycle (not to be gated because it could lead to desynchronized behavior throughout the circuit due to gate delays), the counter outputs two bits, C[1] and C[0] where [1] is the most significant bit of the state. The circuit commits a read or store only when the address that is user specified, SAR[1] and SAR[0] matches the current state. C[1:0] is compared with SAR[1:0] with a comparator, and the output signal sends a 1 out when C[1:0] and SAR[1:0] match. These signals are then AND'd with two other switches representing FETCH and STORE (which act as select signals for MUXs described later in the report). Assuming only FETCH is on or STORE is on during any given clock cycle, an operation is performed if the comparator outputs a 1 (i.e. the user-specified address matches the current register to be written in).

A STORE operation is performed on memory by first loading in D_{in}[1:0] from a 3-to-1 MUX into two buffer registers, implemented with D-flip-flops. That is, the data bits are switches, and once LDSBR is switched on, the 3-to-1 mux selects this D_{in} line. Once the bits are written into the buffer register, the user specifies a certain address by flipping switches SAR[1:0]. Once those are set, the STORE signal is enabled. Then, once the counter matches with SAR, the data from the buffer registers will be written into the address specified by the user. The overall flow: user_data -> LDSBR -> buffer -> user_address -> STORE -> memory (the shift registers).

A FETCH operation is performed on memory by first specifying what address data is to be fetched from with switches SAR[1:0]. Then, the FETCH switch is turned on. Once the counter matches SAR[1:0], the data from the right-most shift register is loaded to the buffer registers. The overall flow: user_address -> FETCH -> buffer register.

<u>Block Diagram/Memory Circuit Implementation</u>

- High-Level Description
    - In order to implement 4-by-2 memory with FETCH/STORE/LDSBR operations, 7 different components (but 8 chips total) were needed. First, there needed to be a counter that tracked where data is written into and read from. In order to ensure the correct address for those two operations, a comparator was used to compare the user's set address and the current address that could be written to and read from. Four additional NAND gates were required for the select logic - FETCH and STORE are AND'd with the output of the comparator and act as select signals for the two 2-to-1 MUXs and two 4-to-1 MUXs (which also use LDSBR as the most significant select signal). In addition to the select logic, two 4 x 1 universal bidirectional shift registers were used (one for the most significant bit, and the other for the least significant bit), and two D-flip-flops were used as buffers. The input into memory is controlled by the 2-to-1 MUX, which selects the output of the shift register when no signal is on (to ensure no loss of data), otherwise it selects data from the buffer when STORE is on. The input into the two buffer registers is controlled by the 3-to-1 MUX which selects the output of the buffer when no signal is on (to ensure no loss of data in buffer), data from user when LDSBR is on, and the output of the shift register when FETCH is on.

- Block Diagram

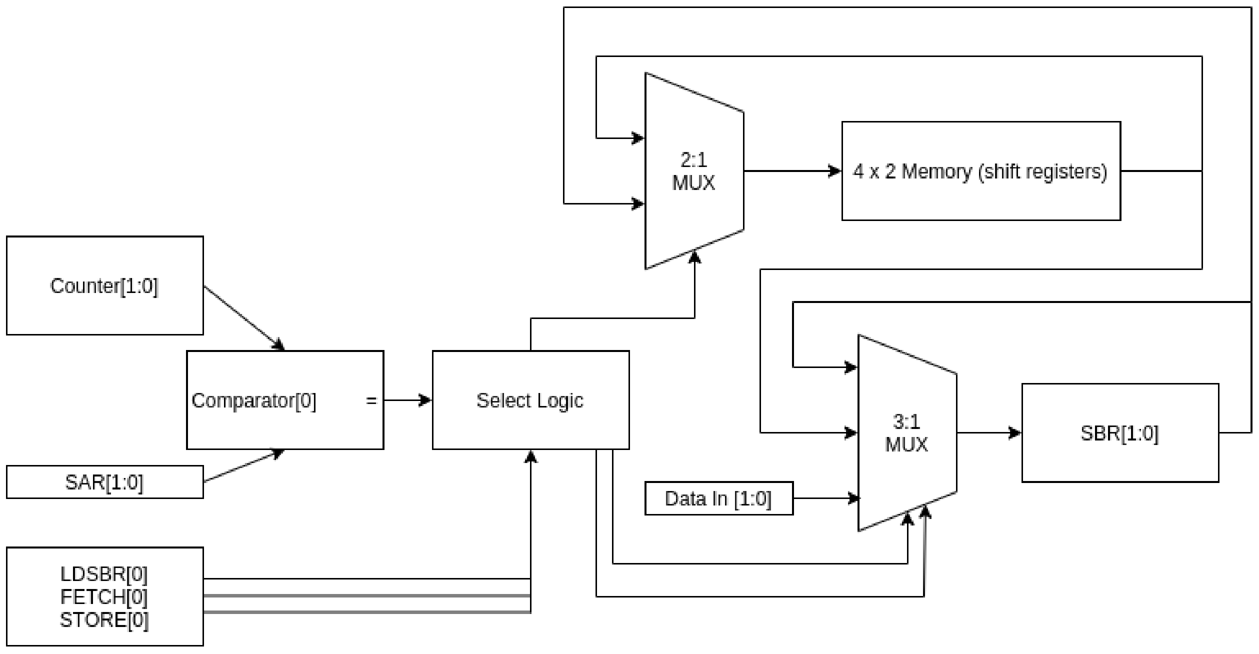

Figure 3.1: High-Level Block Diagram

<u>Control Unit</u>

The control unit in the circuit uses a ripple counter to keep track of the address/shift register that is ready to be either written into or read from. Even though the counter is a 4-bit up counter, only the lower two bits are used since there are only four addresses implemented in this circuit. The output of the counter is the current state, and that output is compared with the address the user sets (SAR[1:0]). When the comparator determines that the state and user address are bitwise identical, it outputs a 1. This output is then AND'd with two other signals, FETCH and STORE, both of which act as select signals for the 3-to-1 MUX and 2-to-1 MUX respectively. Only one is on at the same time. When FETCH is 1 and the comparator's output is 1, that means the current register that the user is trying to read from will be read and put into the buffer register. When STORE is 1 and the comparator's output is 1, that means the current bits in SBR are ready to be written into the shift register specified by the user.

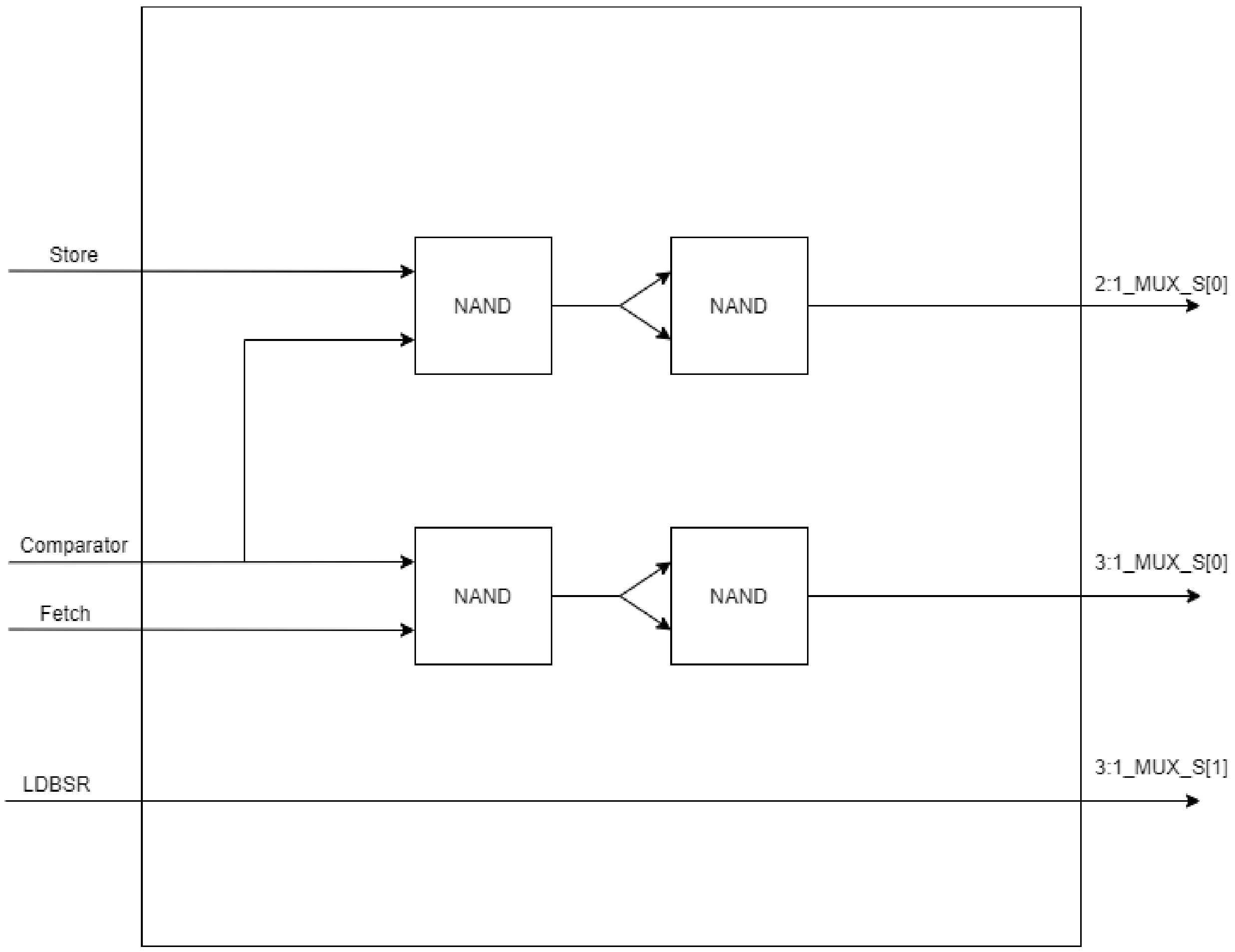

Figure 4.1: Control Unit Block Diagram

Design Steps and Considerations

Initially, a high level circuit schematic was drawn out in order to have an understanding of what general signals go into the unit and what signals come out of the unit. Signals like STORE, and FETCH were used as direct control signals into the select logic components to simplify the design. Another design choice made was using a ripple over synchronous counter. Due to demo conditions, the simplicity of the ripple counter was chosen over the synchronous counter. Even though a synchronous counter is less prone to glitches, the ripple counter could do what the synchronous counter could under the demo testing conditions in a much simpler design. In addition to those design choices, a debounce circuit was made to simulate clock cycles while testing at home to fix the issue of contact bounce simulating multiple rising and falling edges that would lead to non-ideal circuit behavior.

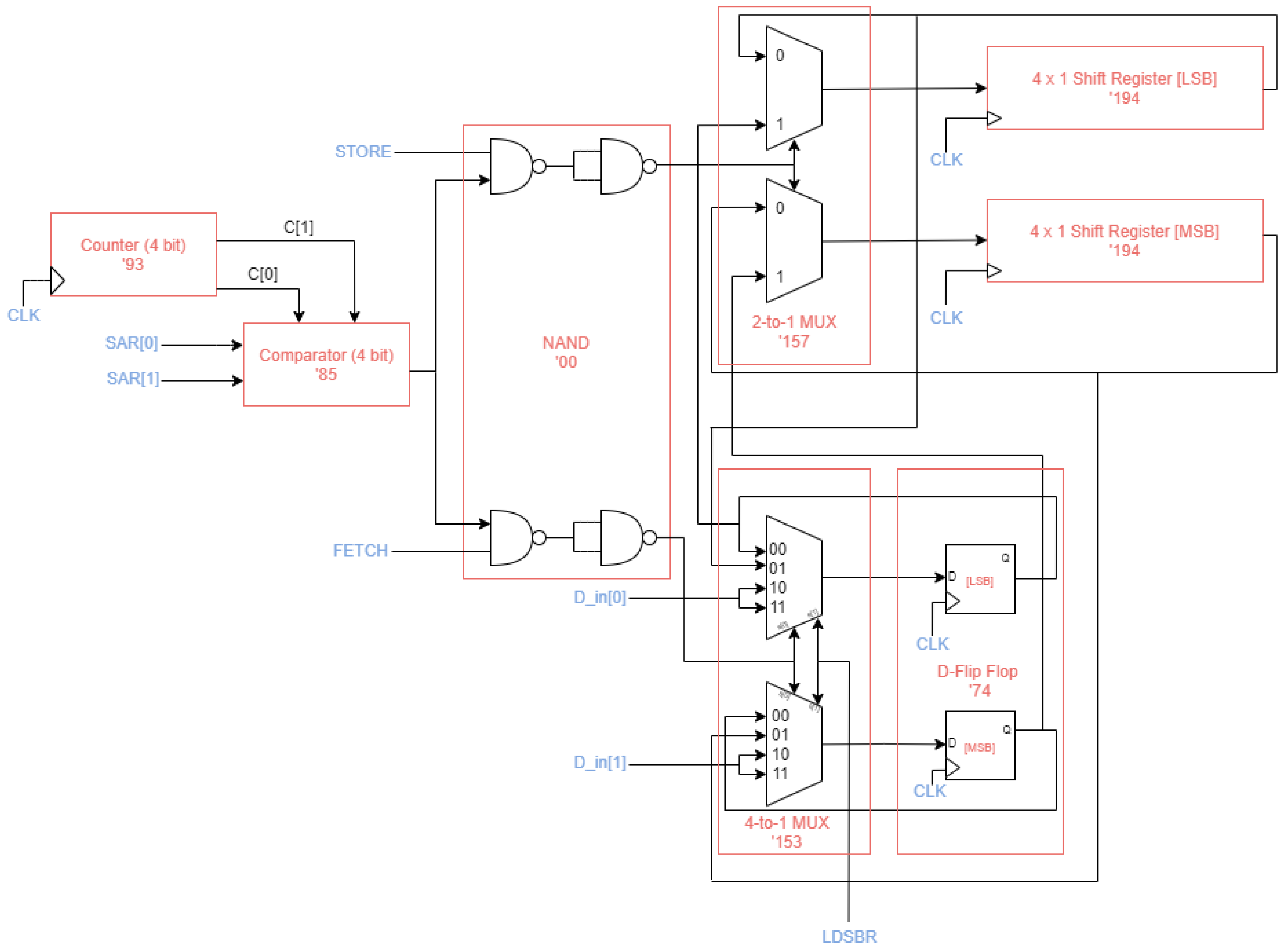

Figure 5.1: Detailed Circuit Schematic

Component Layout Sheet

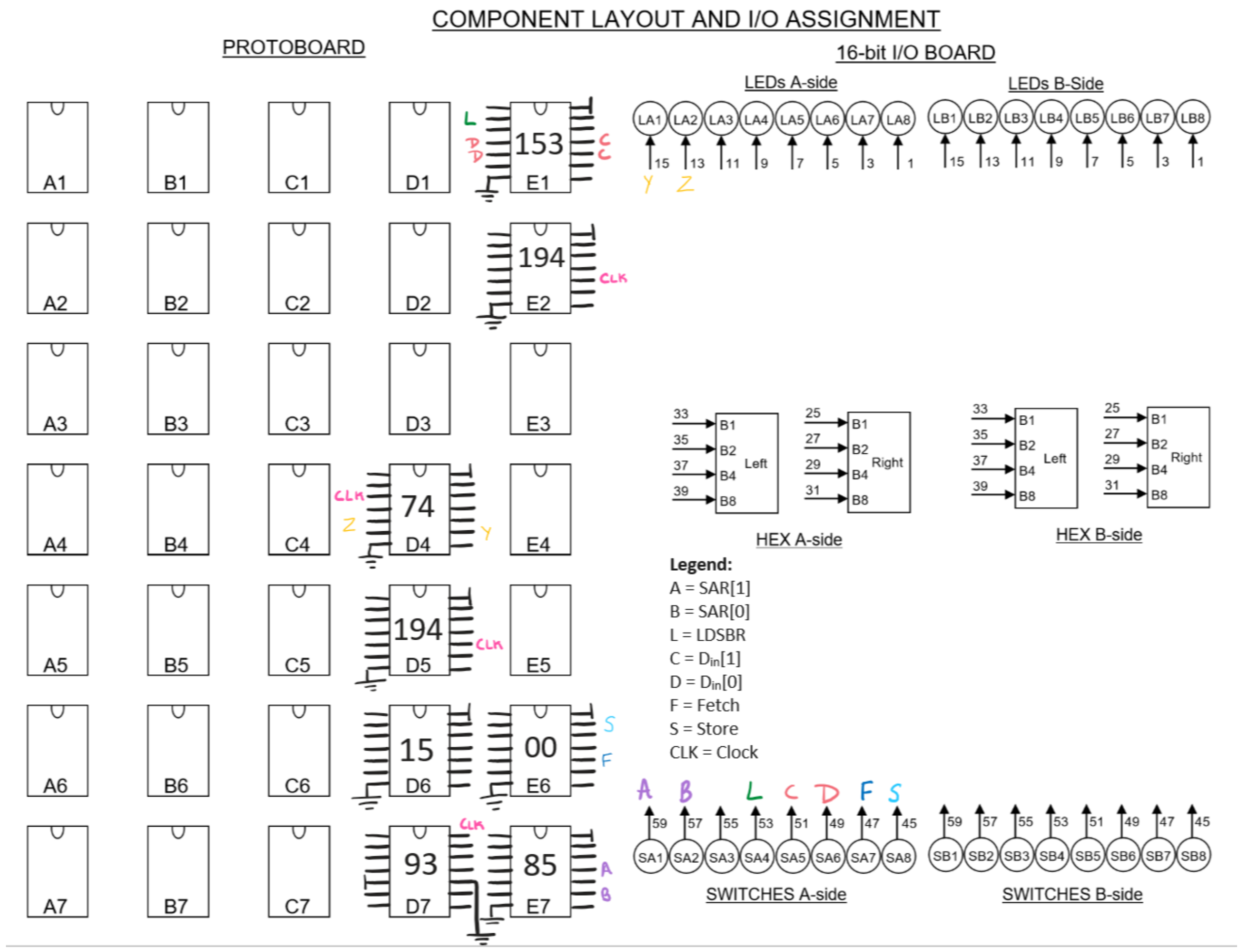

Figure 6.1: Component Layout Sheet

Bugs and Fixes

During the completion of the Data Storage lab, many bugs were encountered. One of the first bugs encountered involved the binary counter IC chip (‘93 listed in Figure 6.1). Taking a closer look at the datasheet revealed the need to connect Q_A (output of the first flip-flop) to the Input B (clock of the second flip-flop) in order to enable the ripple counter to accomplish a standard up count cycle. The second bug encountered involved miswired select logic for the 2-to-1 MUX and the 4-to-1 MUX. More specifically, the select logic for the MUXs had been swapped leading to incorrect STORE and FETCH sequences. Additionally, a major bug encountered in the final stages of the build sequence involved loose IC connections on certain locations on the breadboard (E4 and E5 on Figure 6.1). The fix involved the movement of IC chips ‘153 and ‘194 to the top corner of the board (E1 and E2 on Figure 6.1) in order to ensure reliable connections. The last bug encounter was a loss of LDSBR functionality. After some backtracking on the breadboard, the issue was revealed to be a faulty wire snapped internally. Replacing the faulty wire led to a functional 4 x 2 bit shift register storage unit.

<u>Conclusion</u>

SISO memory is significantly slower compared to SRAM of the same size. SRAM allows for parallel read and write capability leading to a significant increase in speed compared to the serial read and write capability of SISO memory built in the lab. The significant increase in speed does come at a cost. Due to the parallel read and write capability of SRAM, SRAM often takes up significantly larger board space in due to the parallel data paths.

The counter used in the circuit in order to keep track of the addresses was a ripple counter. The implications of using a ripple counter, as opposed to a synchronous counter, was that the ripple counter had a ceiling for the clock cycles it could handle and clock speed past higher than the ceiling could lead to inconsistent outputs. The rationale for using the ripple counter had to do with the 1kHz max clock speed constraint placed on the demonstration as the ceiling for the ripple counter is beyond the relatively slow speed of 1kHz thus the ripple counter offered a much simpler option while being viable given the constraints. The choice of the Universal Bidirectional Shift Register IC was simple as one ‘194 chip included a 4x1bit shift registers thus two ‘194 chips fulfilled our needs of a 4x2 bit shift with a few MUXs for bit selection.

This lab was meant to be an introductory to RAM. In this lab, 4 x 2 RAM was implemented using two 4 by 1 shift registers to hold the bits written into memory, and a counter was used to keep track of what address was the current available address. Signals like LDSBR, FETCH, STORE, and so on determined what operations were to be executed on the current clock cycle. At lower speeds, the clear cycling of bits throughout the registers were clear, but as the clock cycle ramped up to 1 kHz, writing and storing were not as clear. To ensure correct writes and read operations, two buffer registers were used to ensure correct data values.