ECE 385

Spring 2024

Experiment # 2

Lab2: Data Storage

Name: Jie Wang, Shitian Yang
Student ID: 3200112404, 3200112415

Prof. Chushan Li, Prof. Zuofu Cheng
ZJU-UIUC Institute
Jan. 26, 2024, Friday D-225
TA: Jiebang Xia Demo: 5/5

2.2

I. Introduction

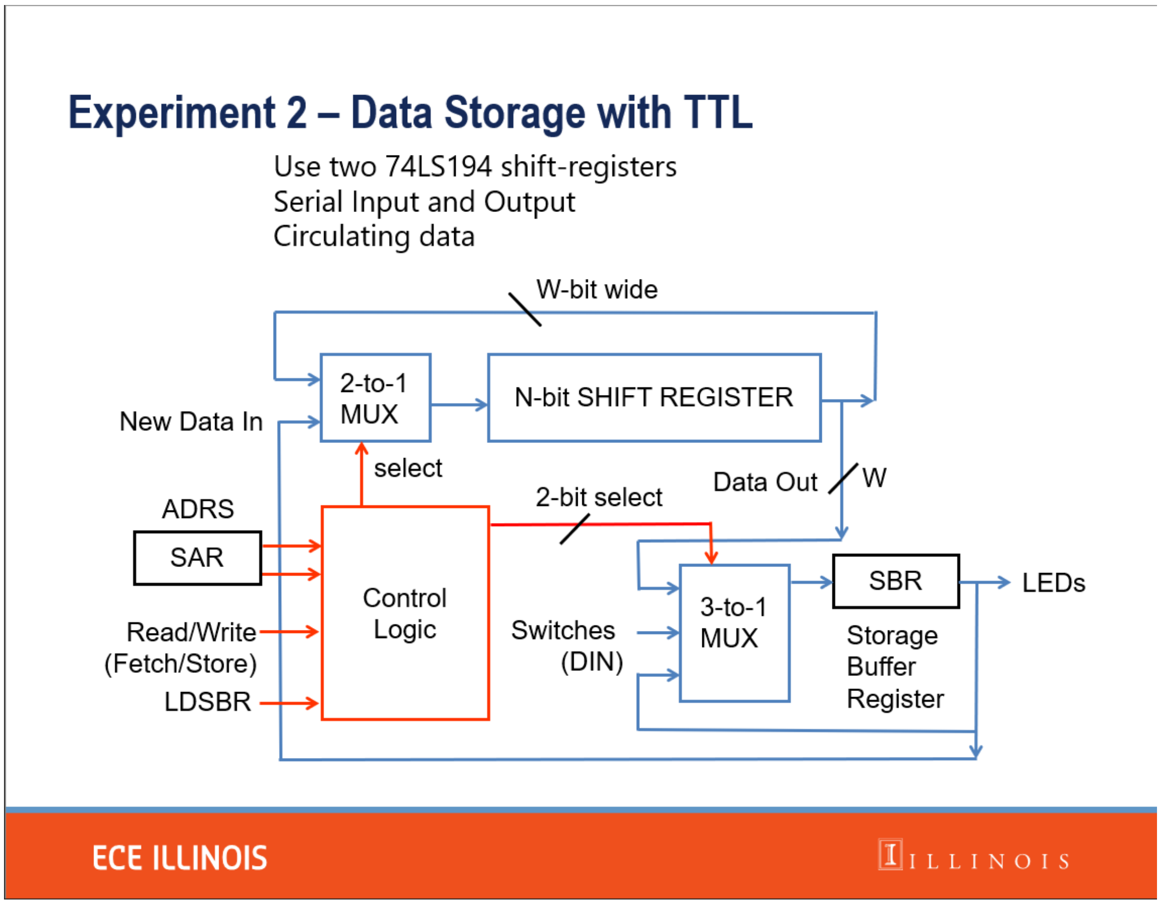
In digital systems, how to store and manipulate data efficiently is a fundamental problem . In this experiment, we design and construct a simple 2-bit, four-word shift-register storage unit on the Quartus Prime + ModelSim Simulation. As shown in the logic diagram below, we need to implement 4 components:

(1) Address controller on the control logic of deciding ‘read/write’ data with counter and comparator.

(2) 3-to-1 MUX to decide the operating mode from ‘store’, ‘fetch’ and ‘load’ states.

(3) 8-bit shift register functioning as the core storage element, which is configured using two 74LS194 data selectors to accommodate the given data.

(4) Storage Buffer Register (SBR)



Experiment 2 – Data Storage with TTL

Use two 74LS194 shift-registers
Serial Input and Output
Circulating data

Figure 1: Logic Diagram for the who Data Storage System

II. Data storage mechanism

To store binary data in the circuit, we typically utilize the coordinated operation of an 8-bit shift register and associated multiplexers. The shift register serves as the primary repository, holding and sequentially moving data. Multiplexers, governed by the control logic, manage data flow by selecting the appropriate input based on the current mode—'store', 'fetch', or 'load'. An address controller, leveraging a counter and a comparator, directs the system's read or write actions, ensuring data is accurately placed or retrieved from the shift register. The Storage Buffer Register provides a temporary hold for data during transfer operations, ensuring consistency and stability in the system's performance. This architecture enables reliable storage and retrieval of 2-bit binary data across four distinct words, demonstrating the fundamental principles of digital data management.

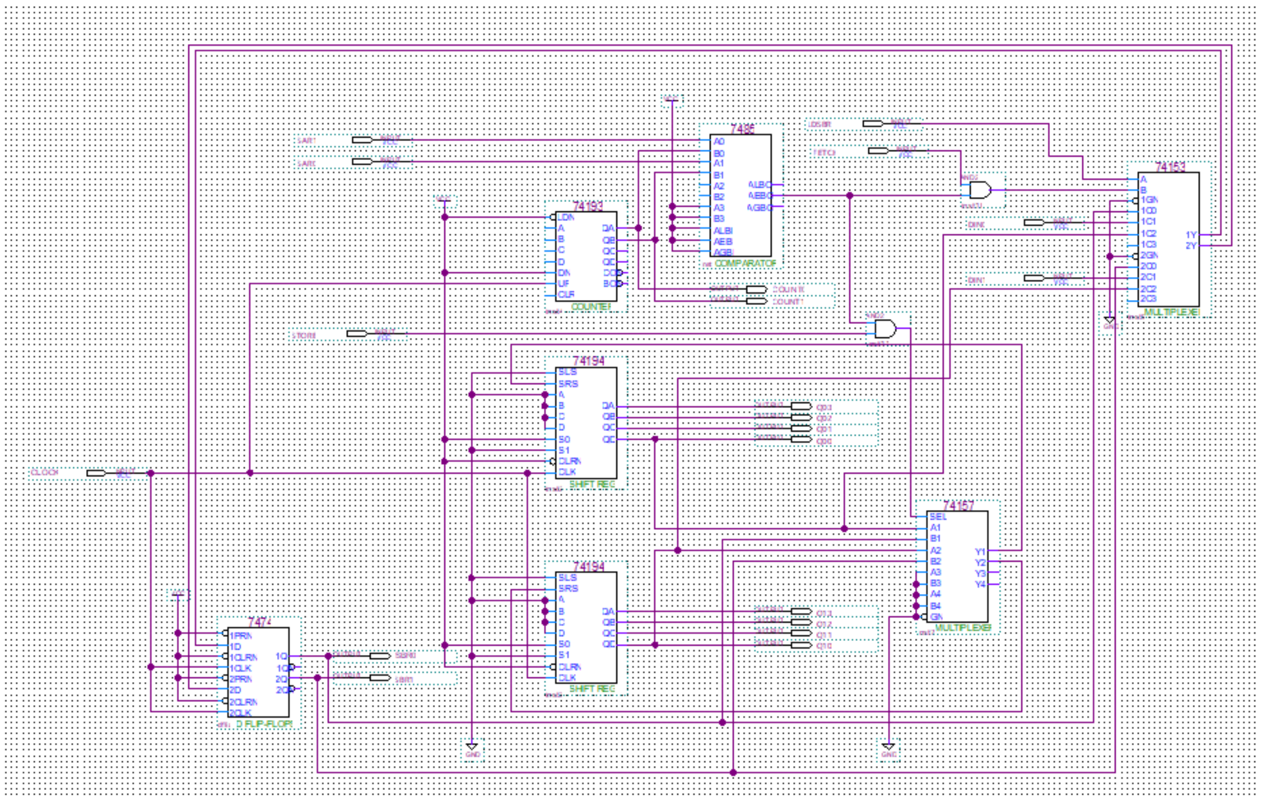


Figure 2: Overview of Our Data Storage Circuit

III. Operation of the memory circuit

a. The addressing function will be opened when the fetch(read) or store(write) flag is on. The counter(74193) will count the number by the input from the clock and return the result to the comparator(7485). If the number of counter equals to the input of SAR(SAR1,SAR2) , the comparator will send signal to the AND gates to activate the fetch or store function.

2.4
b. When store flag is on and the comparator(7485) return activated signal, SBR will send its stored information to the 3-1 MUX and 3-1 MUX will choose these signals to store in the registers.

c. When fetch flag is on and the comparator(7485) return activated signal, 3-1 MUX will choose the current signals from the registers and output them to the SBR.

IV. Written description and block diagram of memory circuit implementation

[1] High-level description

1. Addressing, which includes SAR, counter(74193), comparator(7485). SAR provides which address to be written or read. Counter(74193) will output number by QA, QB basing on the clock input. Comparator(7485) will compare the SAR and counter and output "equal signal" to 3-1 MUX(74153). Then 3-1 MUX will load the output from registers instead of DINs.

2. FETCH, STORE or LDSBR. 3-1 MUX(74153) will load the data from registers if FETCH, STORE, or load the data from DINs if LDSBR. Then it will send result to SBR(7474).

3. Shift registers. The shift registers use two registers(74194) to store data and use 2-1 MUX(74154) to decide to repeat previous memory or load new data from SBR to the registers, which is control by the "store" signal.

[2] High-level block diagram

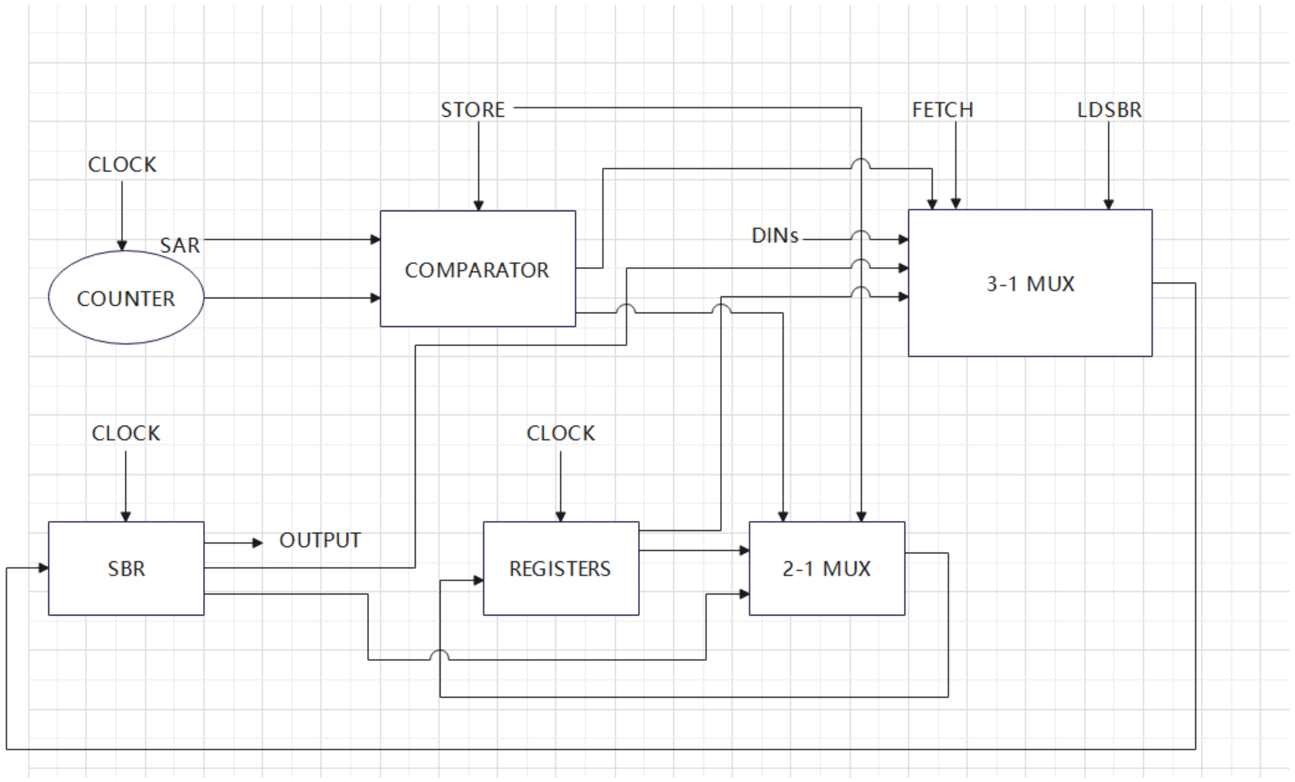


Figure 3: Memory – Circuit Block Diagram

V. Control Unit

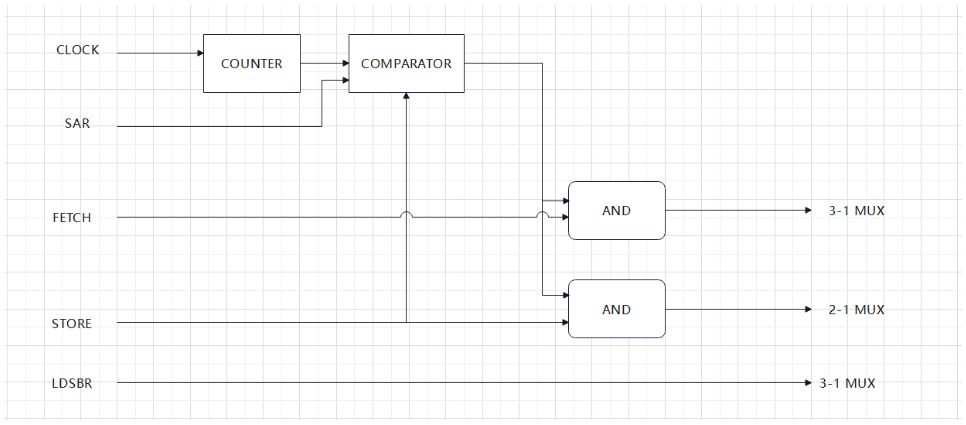


Figure 4: Control Unit Block Diagram

The control unit contains a counter(74193) which is controlled by clock, a comparator(7485) which is controlled by store signal to decide to compare the counter and clock, a 3-1 MUX(74153) which is controlled by fetch, LDSBR and comparator's output signal to design to load DINs, SBR or REGISTERs signal, a 2-1 MUX(74154) which is controlled by store signal to decide return previous memory or SBR signal to registers this turn. (another figure see IV.b)

VI. System Design

[1] Truth Table

Given the condition that only one signal (FETCH, STORE, LDSBR) is active at any time, we can write a Truth table as below. The EQUAL state from the comparator indicates when the address from the counter matches the target address (SAR), allowing data storage operations to proceed.

FETCH	STORE	LDSBR	EQUAL	ACTION	S1S0
0	0	0	0	No operation	00
0	0	0	1	No operation	00
1	0	0	0	No operation	00
1	0	0	1	Fetch data	01
0	1	0	0	No operation	00
0	1	0	1	Store data	00
0	0	1	0	No operation	10
0	0	1	1	Load data into SBR	01

Figure 5: Truth Table of Input Signal Effect

2.6

[2] Detailed Circuit Schematic

As shown in Figures below, we used the synchronous counters to control the data IO cycle instead of ripple counters. Because ripple counter involves glitches and more complex design.

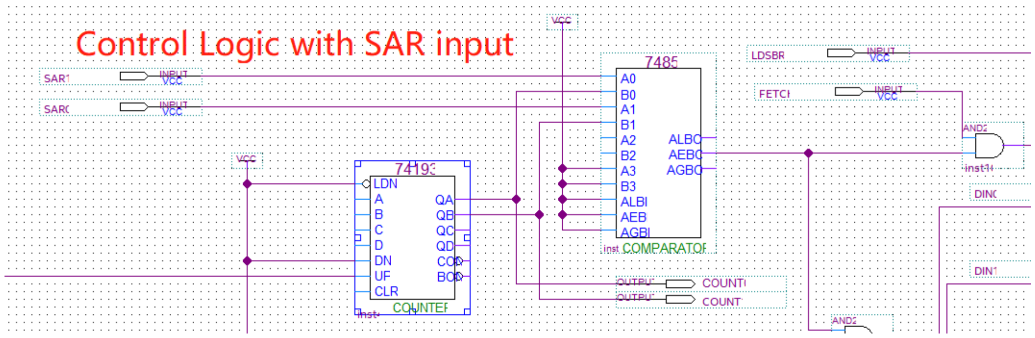


Figure 6: Address controller

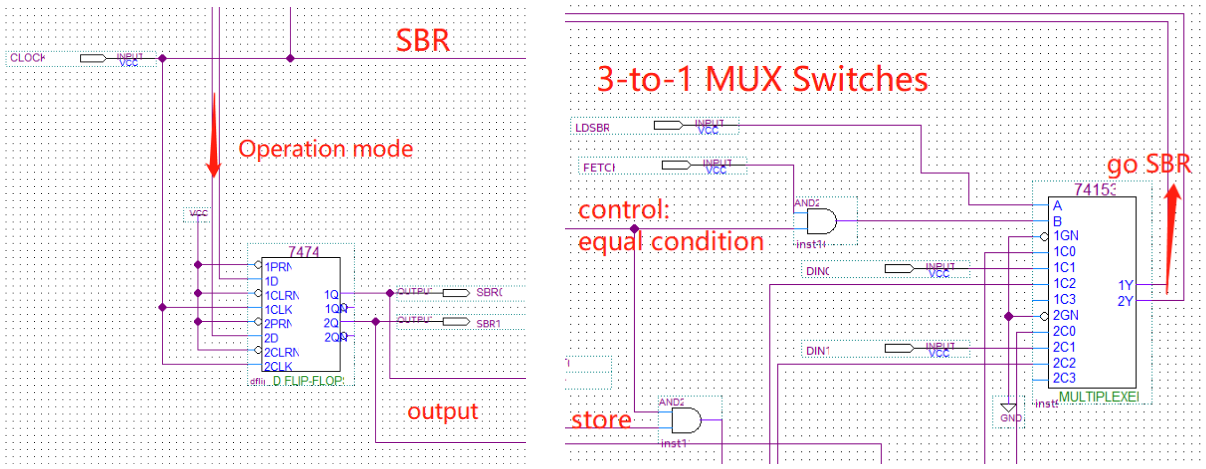


Figure 7,8 : Shift Register for Output, Operation State Switch(right)

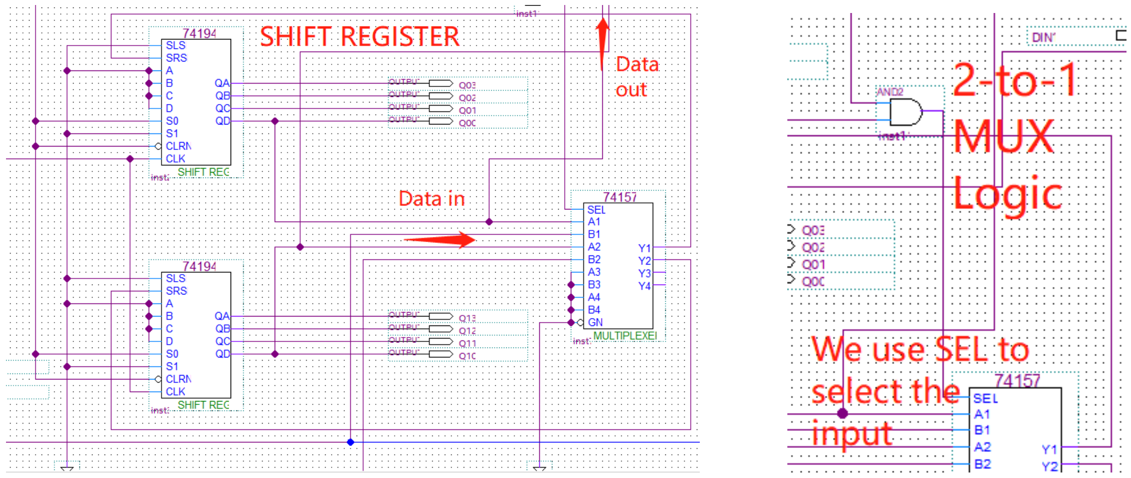


Figure 9,10: Storage Buffer Register (left) and 2-to-1 MUX(right)

Finally, our system works appropriately, the following is result of the given test file 'waveform2.vwf'.

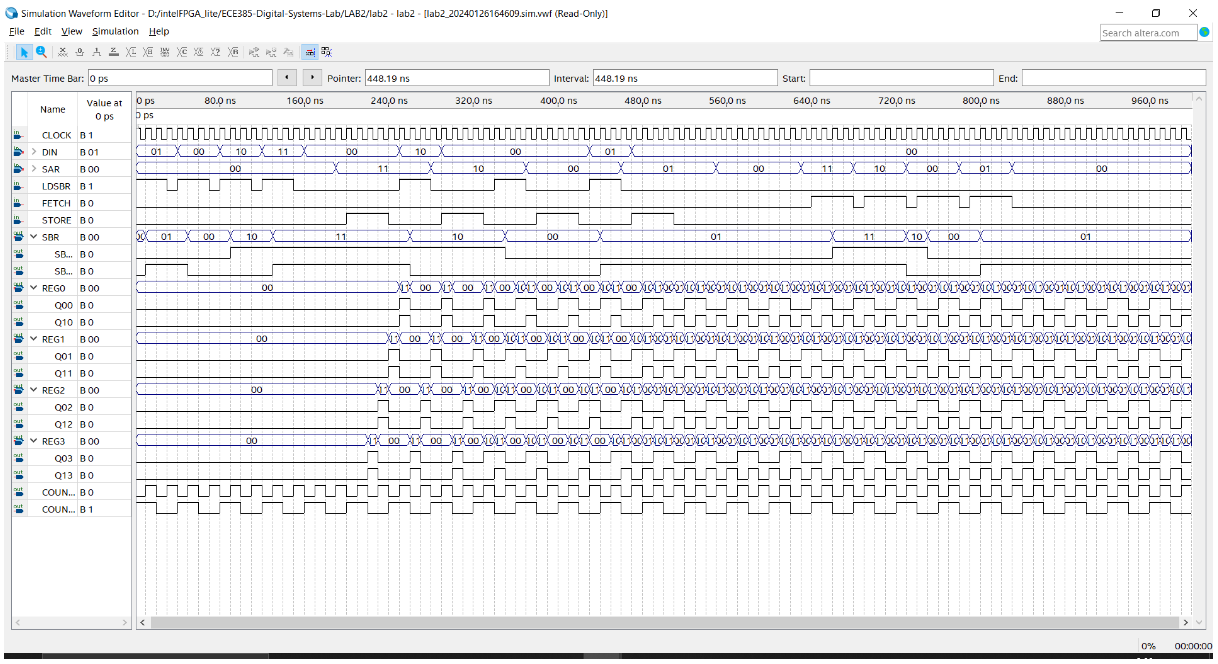

Simulation Waveform Editor - D:/intelFPGA_lite/ECE385-Digital-Systems-Lab/LAB2/lab2 - lab2 - [lab2_20240126164609.sim.vwf (Read-Only)]
File Edit View Simulation Help
Master Time Bar: 0 ps
Pointer: 448.19 ns
Interval: 448.19 ns
Start:
End:
CLOCK
DIN
SAR
LDSBR
FETCH
STORE
SBR
REG0
REG1
REG2
REG3
COUN...

Figure 11: Waveform Result, simulating correctly.

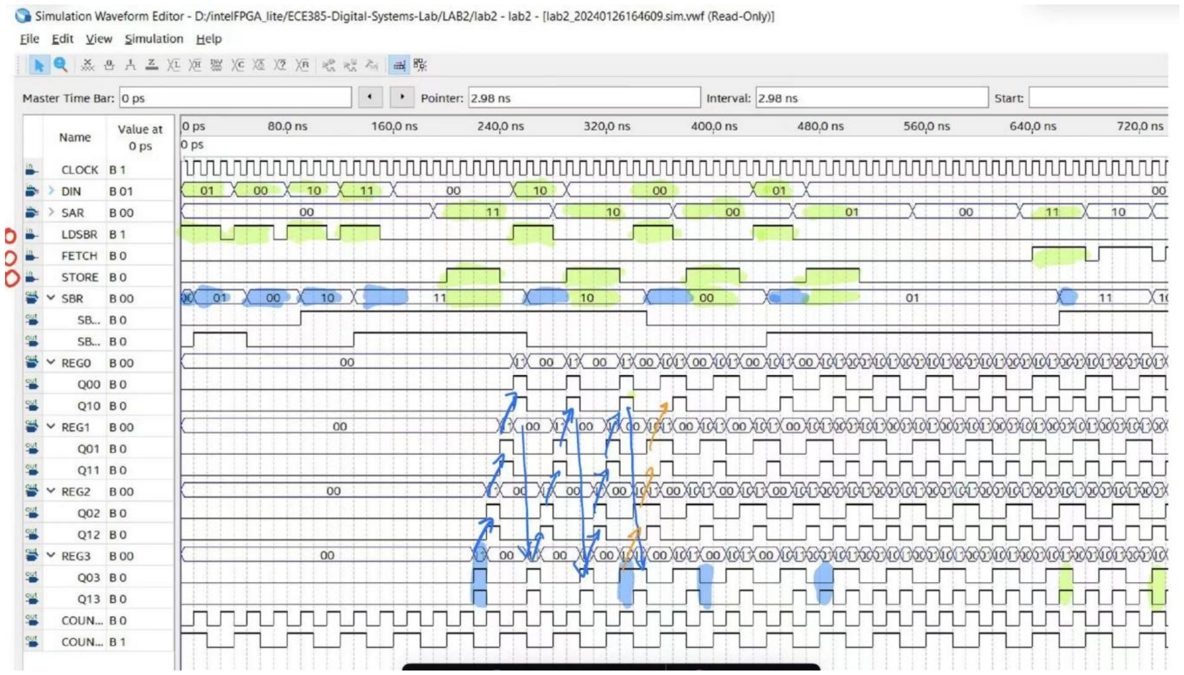

Simulation Waveform Editor - D:/intelFPGA_lite/ECE385-Digital-Systems-Lab/LAB2/lab2 - lab2 - [lab2_20240126164609.sim.vwf (Read-Only)]
File Edit View Simulation Help
Master Time Bar: 0 ps
Pointer: 2.98 ns
Interval: 2.98 ns
Start:
CLOCK
DIN
SAR
LDSBR
FETCH
STORE
SBR
REG0
REG1
REG2
REG3
COUN...

Figure 12: Our manual verification of system

As shown in Figure 12, the Green highlight is Source, and the Blue highlight is Result. During demo, we proved our system works correctly according to our system design.

VII. Bug Log

Below is a brief analysis of each problem we met during the circuit construction:

1. **Incorrect Component Connection**: There was a short circuit created between VCC and GND due to incorrect wiring of components. Additionally, the clock signal was not input correctly, leading to the circuit not functioning as expected. This type of error can cause significant damage to the circuit components and prevent the entire system from operating.

2. **Component Selection Error**: A misunderstanding of the datasheet led to the wrong selection of a counter component, resulting in issues with the counter's input. After replacing the counter with a model that better matched the circuit requirements, based on a correct interpretation of the datasheet, the circuit began to function normally. This highlights the importance of thoroughly understanding component specifications and ensuring they match the circuit's needs.

3. **Mode Selection and Component Linking Issue**: When attempting to select between fetch and store modes, the multiplexer (MUX) did not correctly choose the intended option due to an error in how it was connected to other components. This issue would affect the circuit's ability to correctly perform data storage and retrieval operations, as the MUX plays a crucial role in directing the flow of data based on the selected mode.

VIII. Conclusion

[3] Summary

In this lab, we design a circuit to store 2-bit 4-word data. It shifts the memory and use SAR to find the target address. Read, write function are controlled by fetch and store signal. While building the circuit, we learn how to find the correct and appropriate electrical elements and make the draft clean.

[4] Post-lab problems

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

In digital circuits, gating the clock is akin to an orchestra conductor unpredictably changing tempo, resulting in disarray. This practice disrupts the synchronized operation of the system's components, analogous to the miscoordination among orchestra members, leading to overall disharmony. The clock signal orchestrates the timely and orderly execution of operations, ensuring system coherence.

Moreover, gating the clock can induce clock skew, a detrimental phenomenon that undermines system reliability. Instead, utilizing Clock Enable is a preferred approach. This method allows the clock signal to remain continuous while selectively enabling or disabling operations in parts of the circuit through an enable signal. This control mechanism ensures operations only occur when appropriate, maintaining stability and synchronization without interrupting the clock's continuous flow.

2. Only the clock input needs to be de-bounced to strep through your circuit (why?).

The need for de-bouncing the clock input arises because, at the moment of switching the clock signal, there can be noise and high-frequency fluctuations due to unstable connections. This may cause electronic components to mistakenly believe that multiple clock signals have passed, leading to mismatches. Meanwhile, non-clock signals might require specific handling, and manually defining de-bouncing for them could alter their intended function. Additionally, non-clock signals can be managed through other means to avoid adverse outputs, as demonstrated in operations like those in Lab 1.

3. What are the performance implications of your shift register memory as compared to a standard SRAM of the same size?

For our shift register, we need to use whole cycle time to make sure we can read/write data in correct address, which is time-consuming. While SRAM will read/write data to the certain address almost directly.

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

For the counter, we chose a ripple counter to overlook the impact of electrical noise. The counter options include synchronous counters and ripple counters. Synchronous counters are more precise as they eliminate electrical noise but are more complex in structure. Ripple counters, on the other hand, may have electrical noise, but their structure is simpler.

For shift register chips, we choose two 4-memory registers and choose the left shift mode. Because we need to accomplish 2-bit 4-word shift memory circuit. Four 2-memory registers will be quite complex to be organized.