ECE 385 Lab Report

Spring 2023

Lab 2: Data Storage

Name: Zhu Hanggang, Hong Jiadong

Student ID: 3200110457, 3200110970

Prof. Chushan Li, Prof. Zuofu Cheng
ZJU-UIUC Institute
April 6, 2023

1 Introduction

In this lab, we build a simple 2-bit, four-word shift register storage unit. The main functionality of the circuit includes using MUX to decide whether to store, fetch and load data. There's also a control logic to decide when read/write data in shift register using a counter and a comparator.

2 Operation of the memory circuit

For the addressing method. We use counters and comparators to perform the addressing operation. We set the comparator's comparison level to the address level. The circuit should read only when the right-most address in shift registers matches the address in SAR. The circuit should write only when the left-most address in shift registers matches the address in SAR.

For the write operation, firstly the new data is input into the DIN input. When LDSBR signal is set, the control block sends signal to permit new data go through 3 to 1 MUX and store DIN into the SBR. When STORE signal is high, the 2 to 1 MUX will choose the data in SBR and write the data into shift registers. If and only if both the addressing operation signal AND the STORE signal are high, the new data can be written. Basically data flows from DIN to SBR and then finally to shift registers.

For the read operation, the data should be loaded to the SBR when the FETCH signal is set. When FETCH signal is high, 3-to-1 MUX will load the data from shift registers to SBR .If and only if both the addressing operation signal AND the FETCH signal are high, the new data can be read. Basically data flows from shift registers to SBR.

3 Written description and block diagram of memory circuit implementation

3.1 High-level description

Our circuit can be roughly divided into four parts.

1. Addressing. Use SN7493 4 bit ripple counter and only uses to QB and QC outputs to count 00,01,10,11 for each clock cycle. Use SN7485 4 bit comparator to judge if output counter matches address in SAR. For convenice, we denote the output of this part as EQUAL.

2. SBR Logic. Use SN74156 Multiplexer to use three kinds of input for the SBR. 1:data from rightmost shift-register(read), 2:DIN(load) and 3:itself(do nothing). The SELECT is determined by LDSBR, FETCH, and EQUAL. So we use some AND gates deal with the logic. The details of this will be introduced later.

3. Register Memory. Use two SN74LS194A shift registers to store bits. One serves as bit 0 and the other serves as bit 1. Use right shift strategy. A SN74157 Multiplexer to use two kinds of input for registers in leftmost position. 1:data from SBR (write), 2:data from rightmost position(cyclic shift). The SELECT is determined by EQUAL and STORE. In our design, only one additional ADD is required.

4. SBR. Use SN7474 D Flip-flop to represent SBR. The input controlled by the SBR Logic part. Output of the flip-flop represents the data in SBR.

3.2 High-level Block Diagram

See figure 1.

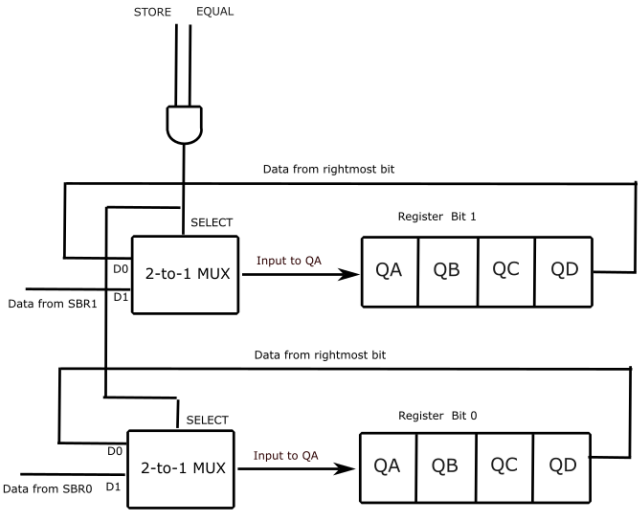


Figure 1: Memory circuit block diagram

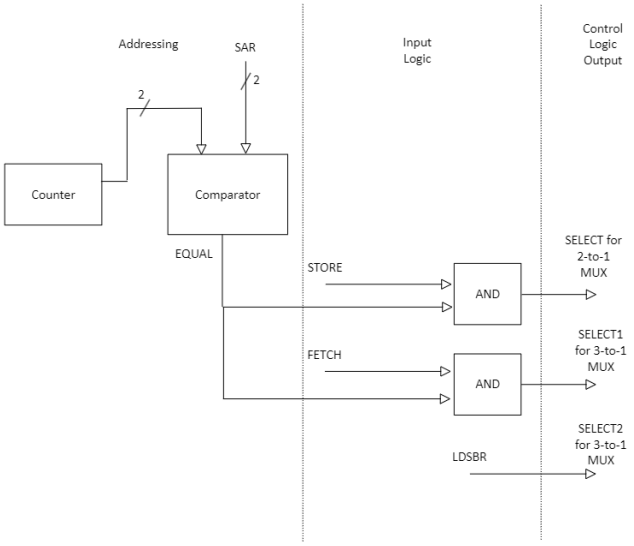


Figure 2: Control unit block diagram

4 Control Unit

The role of control unit is to set SELECT input to MUX with the given inputs. For the 3-to-1 mux,the SELECT is presented as S1S0 with 00 meaning do nothing, 01 meaning LDSBR, 10 meaning reading data. The 2-to-1 mux is simper, only read data from SBR if both STORE and EQUAL are 1. The detailed design of AND gates is in the next section.

For the block diagram of control unit, refer to figure 2.

5 Design steps taken and detailed circuit schematic

5.1 K-map and Truth Table

We make k-maps for the 3-to-1 mux. The main logic of 3-to-1 mux is as follows. Note that we make the assumption that you can do LDSBR, FETCH, STORE one at a time(You can't do two operations at

LDSBR	FETCH	STORE	EQUAL	S1S0
0	0	0	0	00
0	0	1	0	00
0	1	0	0	00
1	0	0	0	01
0	0	0	1	00
0	0	1	1	00
0	1	0	1	10
1	0	0	1	01

Table 1: 3-to-1 MUX Truth Table

ST/EQ \ LD/FT	00	01	11	10
00	0	0	×	1
01	0	0	×	1
11	0	×	×	×
10	0	×	×	×

(a) K-map for S0

ST/EQ \ LD/FT	00	01	11	10
00	0	0	×	0
01	0	1	×	0
11	0	×	×	×
10	0	×	×	×

(b) K-map for S1

Figure 3: 3-to-1 MUX K-map

the same time). Please refer to table 1 and figure 3.

Note that we use abbreviation here. STORE(ST),EQUAL(EQ),LOSBR(LD),FETCH(FT). According to the K-map we can get logical expression for S1 and S0:

$$S0 = LDSBR$$

$$S1 = FETCH\ AND\ EQUAL$$

5.2 Design consideration

In terms of the design considerations taken, we use the simplest possible way. For other considerations, we can use synchronous counters instead of ripple counters.If we use ripple counters, it's simper to build and use but there may be glitches. Since we are only doing the simulation, we believe this can be ignored. If we use synchronous counter, it's more complex but we can remove the glitches in real design.

5.3 Detailed Circuit Schematics

See figure 4 and figure 5

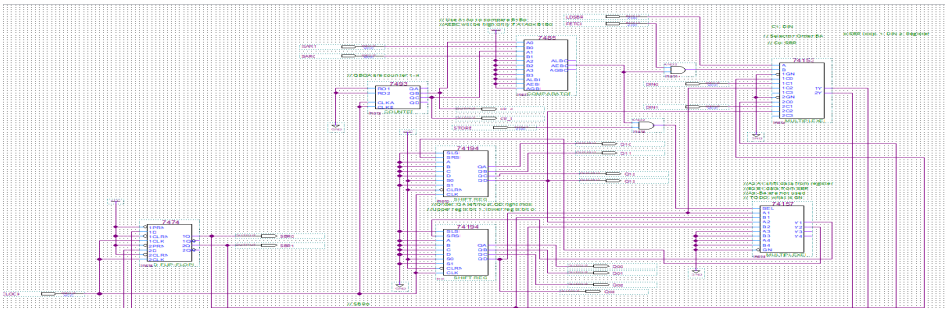
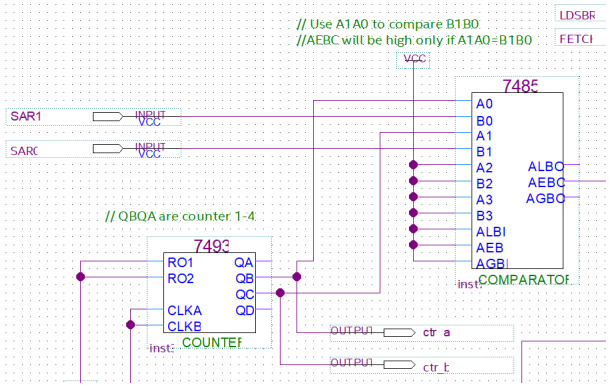
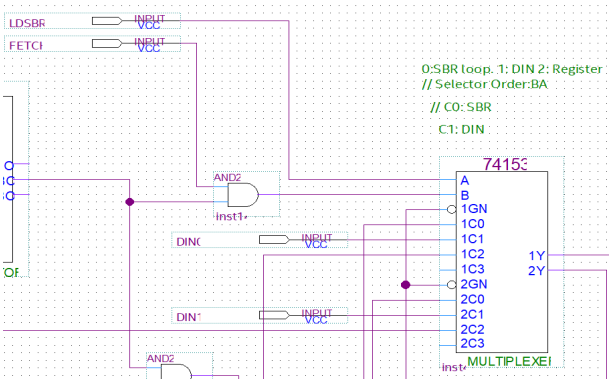


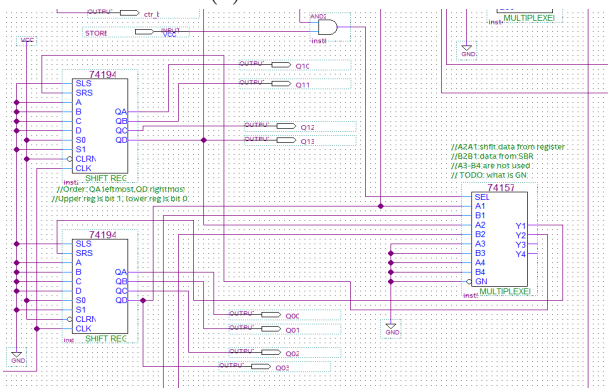
Figure 4: Circuit Overview



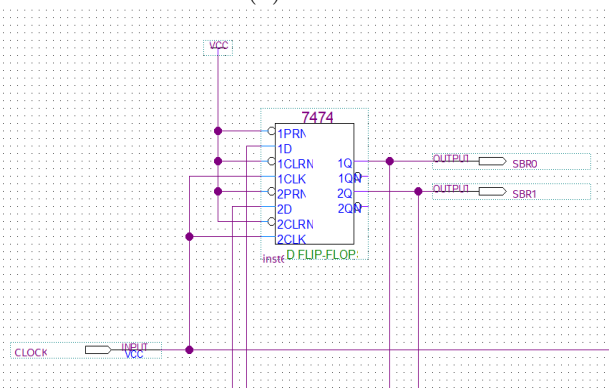
(a) Address Controller



(b) 3-to-1 MUX



(c) Shift Register and 2-to-1 MUX



(d) SBR

Figure 5: Detailed Circuit Schematic

6 Description of all bugs encountered, and corrective measures taken

1. Output at SBR is not correct. The LDSBR is mistakenly connected to CLK pin in D flip-flop. This occurs because we see from the truth table that when CLK is high, data is loaded into the flip-flop. The corrective way is to connect clock signal to CLK pin and output from 3 to 1 MUX to input D.

2. Data loaded into registers are not correct. This is caused by the wrong choice of output of the counter. At first we choose QA and QB as output but they always produce the same result. When

we try to use QB and QC, it produces what we want.

3. Data not written for the second STORE command. This is caused by the wrongly positioned bit 0 and 1 of SAR. These two bits are exchanged and put into the comparator. The corrective way is to make sure bit position in SAR and counter matches.

7 Conclusion

7.1 Summary

In this lab, we design a 2-bit four-word shift register storage unit. It includes some basic logic implementation and the analysis of different circuit operations like read and write. We also learned some tradeoff between different designs and learn how to build good circuits.

7.2 Post-lab Questions (including prelab questions)

1. **Why is clock gating a bad practice in digital design?**

   Because it may cause delay of the clock signal. A gating clock design makes components out of sync, which may cause glitches or wrong instructions

2. **Why does only the clock input need to be de-bounced to strep through the circuit?**

   Because the clock input is important to keep the timing and synchronization of the system. We need to de-bounce the clock to make sure everything is synchronized. For other inputs, they are not necessarily required as they may not have affect on the synchronization of the system.

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, one must wait for some clock cycles to access or store the data. It's just a simple implementation for storing data. But for SRAM, you don't need to wait for these clock cycles. SRAM supports fast access of data. So overall our shift register is less efficient in storing memory.

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 can use synchronous counter or ripple counter. Synchronous counter is more accurate as it removes glitches but more complex. Ripple counter has possible glitches but simpler. We choose ripple counter as we are doing the simulation and we can ignore the impact of glitches.

   For the shift register chips, we use right shift mode as this is more intuitive than left shift. We can't use parallel load mode as we don't need it to store 2 bits.