ECE 385

Fall 2020

Experiment #2

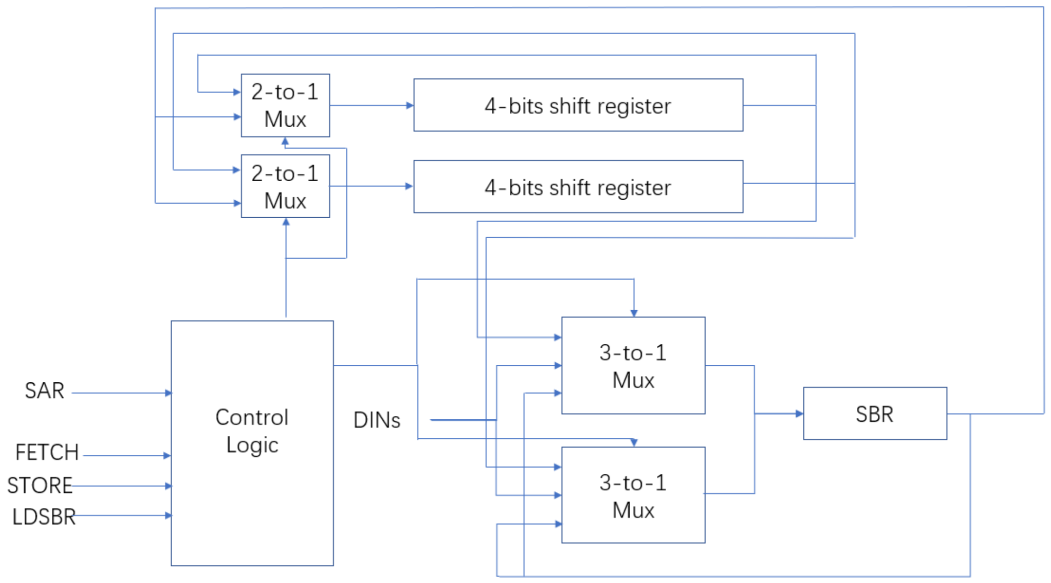
Lu Yicheng
Xiao Shuhong

Pre-lab writeup

0,question
The registers must be shifted on each clock pulse. The clock must run continuously – do not gate the clock (this is bad practice in digital design, why?)

If gate the clock, the rising edge of the clock may not be same for every flip-flop or other component. This may cause delay or even glitch.

1.Block diagram



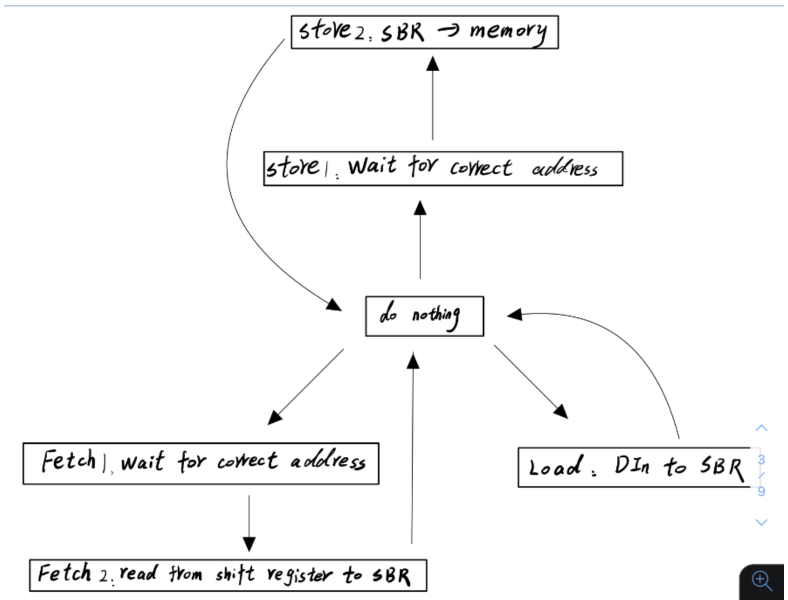
2.operation of the controller& circuit operation

We have six kinds of input. They are: SAR,FETCH,STORE, LDSBR,DINS

We have two kinds of output. They are: SBR, data in shift register(just for illustration)

We have three operation and one normal state as you can see clearly in the picture below:

NOTE: there is no need to build an FSM though we draw the picture below


store2: SBR → memory
store1: wait for correct address
do nothing
Fetch1: wait for correct address
Fetch2: read from shift register to SBR
Load: DIn to SBR

For load operation, we just need to load data to SBR from DINS. To be specific, when LOAD signal is high, we set "2 bit selects" to be 11. So that the mux will choose the DINS.

For write operation, we need to wait until the we meet the expected address and write to that address. To be specific, when WRITE and COMPARE signals are all high, set "select" to 1.

For read operation, we need to wait until the we meet the expected address and load data to SBR from shift register. To be specific, When READ and COMPARE signal are all high, set "2 bit selects" to be 01

For normal state, just set all signal to zero. This is because we want to do nothing but only shift the memory.

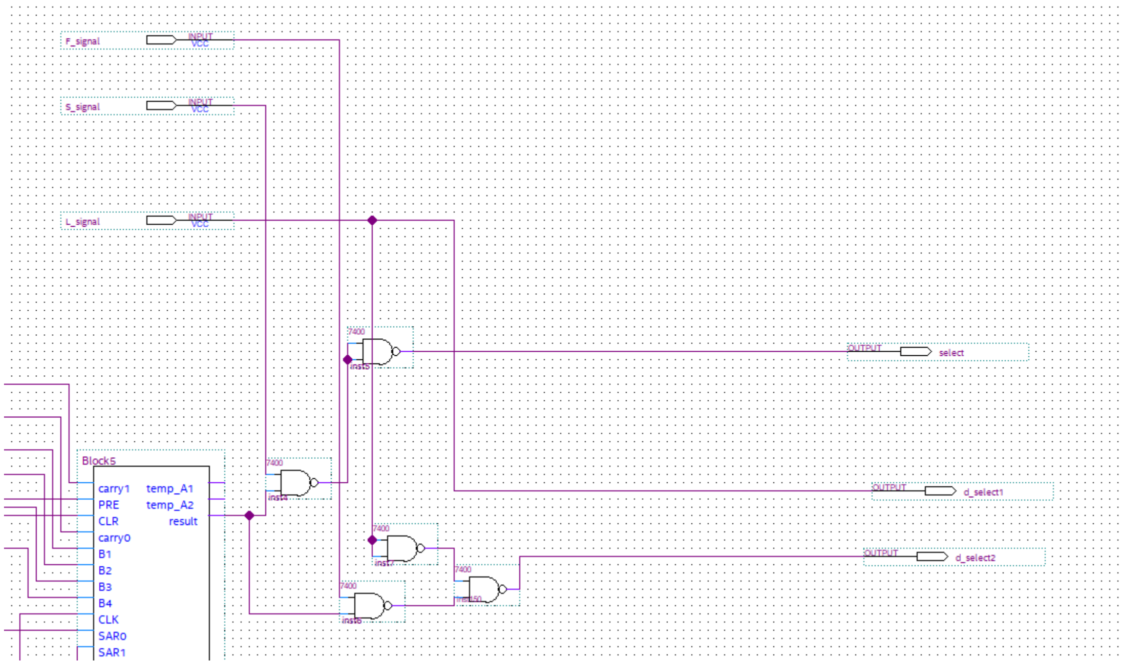
Here are the Truth Table and Boolean Equation for these operations:

signal:

fetch	store	LDSBR	Compare	select	2 bits select
0	0	0	0	0	00
1	0	0	0	0	00
1	0	0	1	0	01
0	0	1	0	0	11
0	1	0	0	0	00
0	1	0	1	1	00

Select=SC
2 bits select[0]=L
2 bits select[1]=FC+L
where comparer result signal is C, FETCH is F, SRORE is S, LOAD is L.

logic diagram(signal result is just the compare signal)



1, Introduction

In this lab, we designed and constructed a 2-bit, four-word shift-register storage unit. This unit can do three things:

1. store the data to the expected address in memory
2. read the data from expected address in memory.
3. Load the data to storage buffer register, and this data will finally load into memory after a "store" operation

2. Operation of the memory circuit

a. actually, for our 2-bit four-word memory, the purpose of the address is to help us locate the stored data, so that we can use the address as a medium to accurately select the data we want to read. Since we cannot know the currently accessible address from the Shift Register itself, we use an external counter to synchronize the address movement. For four words, we can use 2 bits to represent four different addresses, defined as 0,01,10,11. in our standard lab kit, however, we only have 4 bit ripple counter, the solution is to only use least Significance for 2 bits of the counter. The state of the counter represents the leftmost address that can currently be written in. as all the flip flop we use are Dual D Positive Edge Triggered Flip Flop, all the change happened at the rising edge of our clock, and when the fetch signal is high and the state of counter become the same as what we input in SAR, the read operation begin; when store is high, the state of counter become the same as input in SAR, write operation begin. however, one thing to mention is in our design, because of the circuit delay, the read or write finish at the next clock, of course, it will not affect our operations.

b. for a write operation, the input from DINs should be load to SBR first, that is, LDSBR should be set to 1, and at rising edge of our clock, input is loaded. also, we need to decide which address for the data to be stored in, that is, a data address is need by set input SAR, switch the STORE to 1, the counter run until it match the address we claimed in SAR, and at rising edge of clock, data successfully stored, then we can switch SRORE to 0.

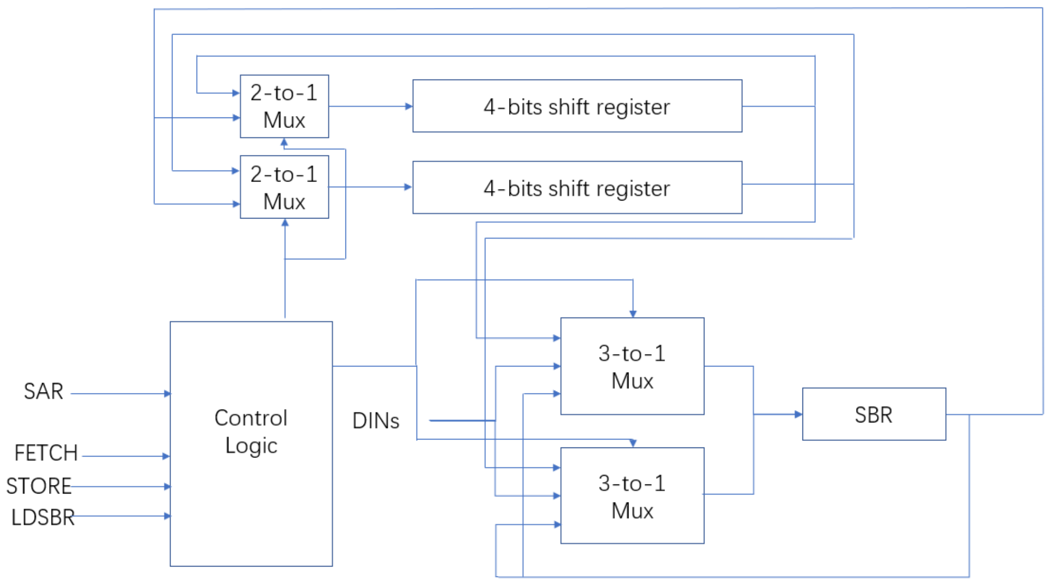
c. for a read operation, first is to decided which address we are looking for, that is, claimed our address by set SAR, then we set FETCH to 1, counter run until it matches what we have in SAR, then the data we need goes to SBR, then output.

3. Written description and block diagram of memory

a. High-level description

for such 2 bit 4-words memory storage, we need two 4-bits shift register (74194) for shift data; two 4-bits full adder (7483), one as counter and another used to compare SAR and counter for matching; four 4-to-1 mux (74153) and 3 D flip-flops (7474) and several NAND(7400) and NOR(7404) gates.

b. high-level block diagram



4, Control Unit

4.1 Provide an intuitive written description of your control unit.

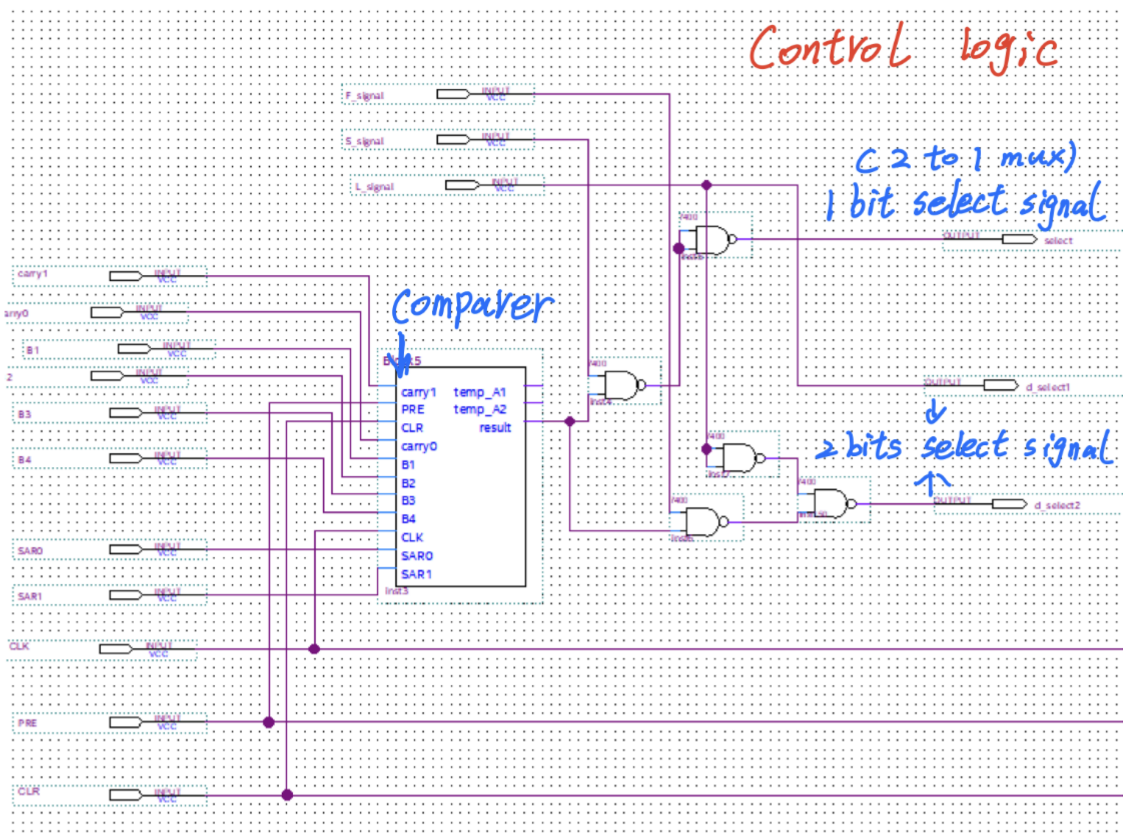
Control logic has four tasks. First task is Control logic should always compare the current address in shift register and the address in SAR. The second task is that when executing the STORE operation, Control logic need to load data to shifter register from SBR if two addresses are same. The third task is when executing the FETCH operation, Control logic need to load data to SBR from the head of the shift register if two addresses are same. The last task is to load DIN data to SBR.

In order to fulfill the first task, we first construct a 2 bits counter by using four flip-flops and 4 bits adder. And then we construct +6* consisting this counter and another adder called A. The outputs of the counter are considered as the current address in shift register. The adder A receives the outputs of the counter and the 2's complement of the desired address from SAR. As a result, the output of the adder A is 0 if two addresses are same. For convenient, we add a inverter to the output. So, two addresses are same if output is 1.

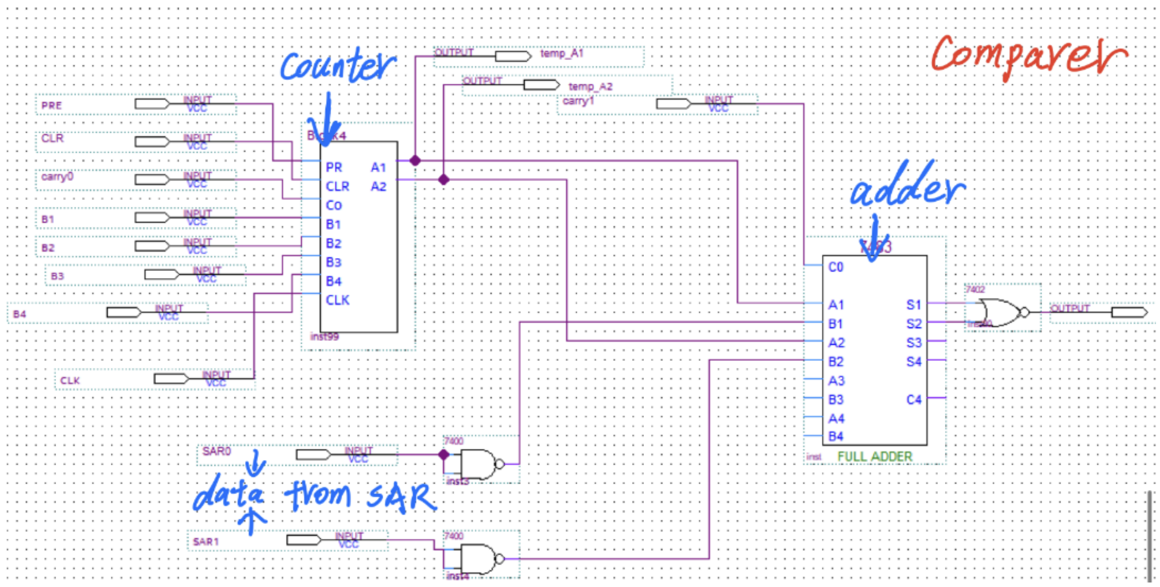
To fulfill the second task, we have to output a 1 bit select signal to a 2-to-1 mux to control the input of the shift register. That is, when STORE signal and compare result signal are high, select signal have to be 1 in order to put SBR data to the tail of shift register instead of the data in the head of the shift register.

To fulfill the third and fourth task, we have to output a 2 bit select signal to a 4-to-1 mux to control the input of the SBR. We define that SBR holds if select==00, SBR receive the data form DIN if select==11, receive the data from shift register if select==01. So, when LOAD signal is high, select==11. When FETCH signal and compare signal are high, select==01. For another cases, select==00.

4.2 Include a block diagram of your control unit.

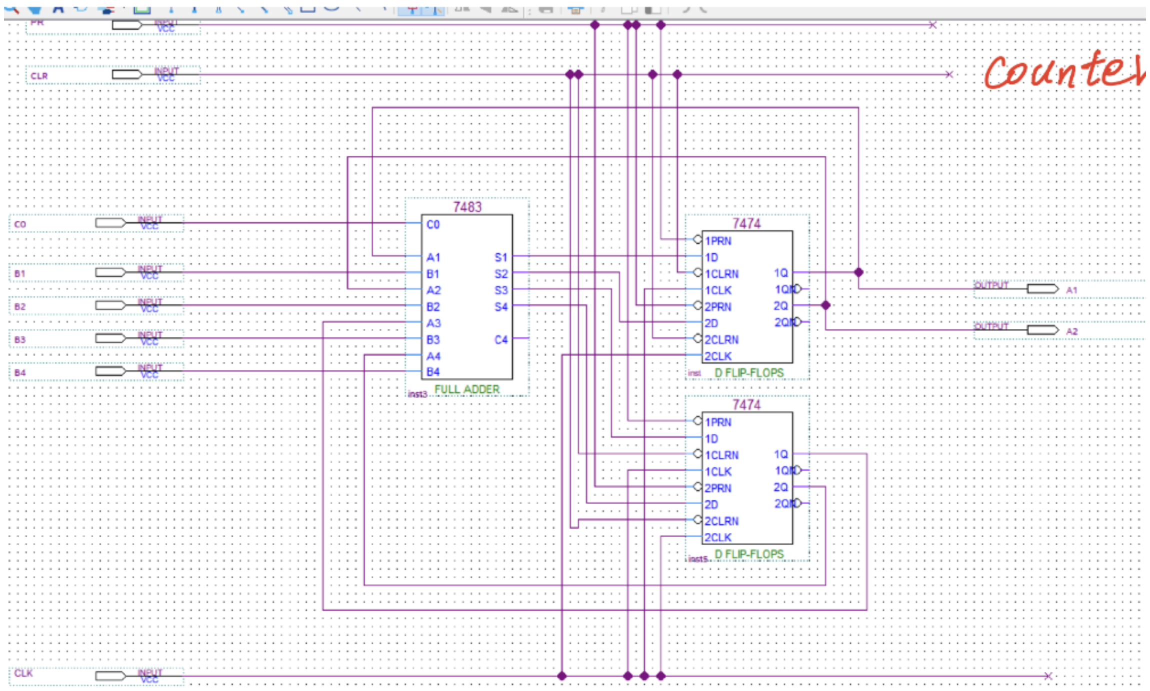


Above is the control logical.
Select=SC
2 bit select[0]=L
2 bit select[1]=FC+L
where comparer result signal is C, FETCH is F, SRORE is S, LOAD is L.



above is comparer(part of control logical).
Output=counter + 2’s complement of SAR.

above is counter(part of comparer).



Output+ = output + 0001

5,Design steps taken and detailed circuit schematic

5.1 truth tables

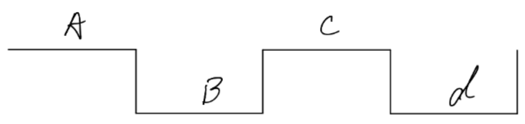
Signal:

fetch	store	LDSBR	Compare	select	2 bits select
0	0	0	0	0	00
1	0	0	0	0	00
1	0	0	1	0	01
0	0	1	0	0	11
0	1	0	0	0	00
0	1	0	1	1	00

Above is a truth table for control logic block. We can see directly that only when fetch and compare are high, the “2 bits selcect”[1] is high. Only when store and compare are high, the “1 bit select” are high. Only when fetch and compare are high, or ldsbr is high, the “2 bits selcect”[0] is high.

5.2 design considerations

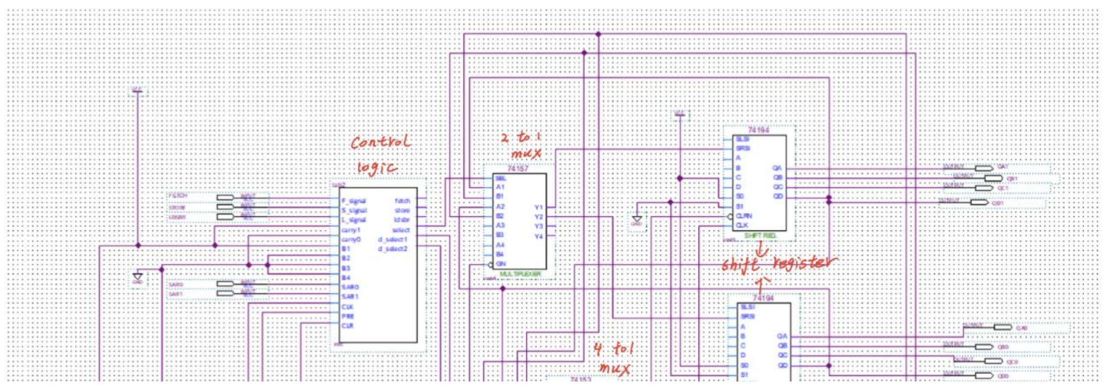
1, when designing counter and comparer, we find that the delay of the adder will have no influence if we connect a Flip Flop to the output of the counter. Actually, even if we use the Ripple counter, as long as the delay is smaller than one clock cycle, there are no influence.
Take the picture below as an example. As long as the output of the comparer is correct before time C, flip-flop will store the correct value and show this correct value at the time the clock changes to C region from B. As a result, all combinational logic has exactly one clock cycle to reach to the correct value if connected with a Flip Flop.

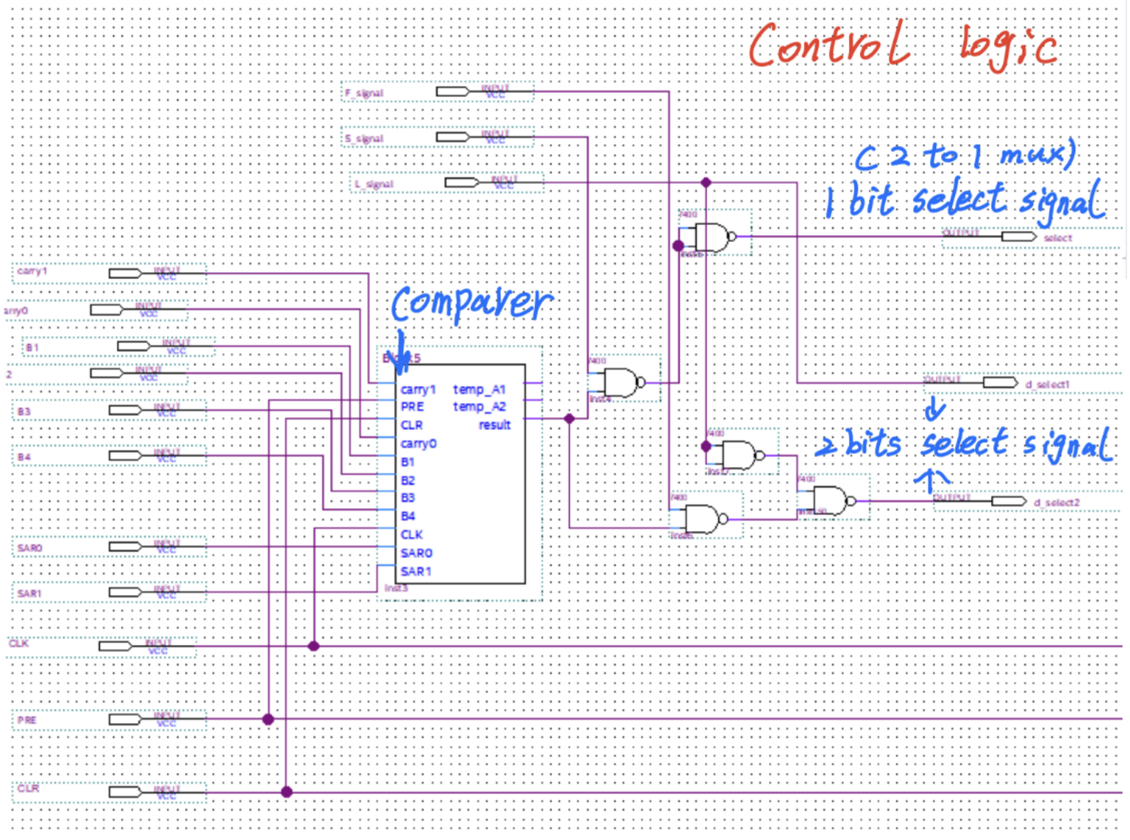


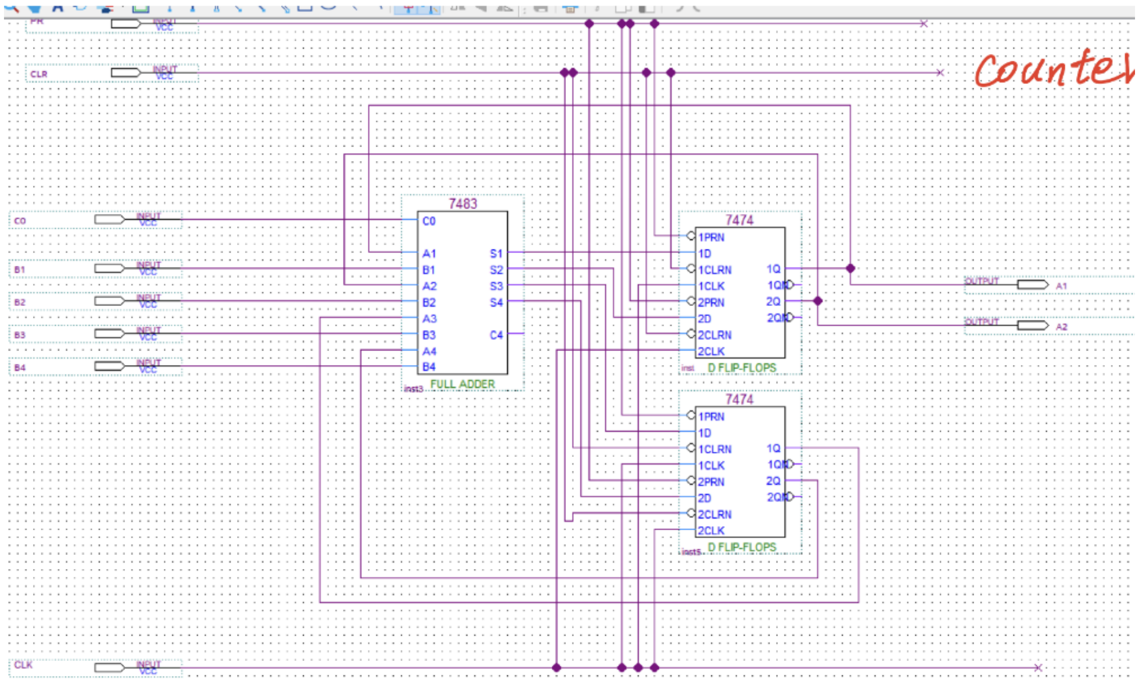
2, When designing the counter and comparer, we can only find the 4-bit Binary Adder instead of 2-bit. Since the addresses should return to 00 after 4 cycles, we get confused for a lone time about how to get 0000 instead of 0100 after 4 cycles if using the 4-bit Binary Adder. Actually, ignoring the left two bits is enough. After 4 four cycles, two bits on the right hand are always zero.

3, When looking at the waveform, we find that if we store value AB into the address CD, Value AB will appear in the shift register at the next clock cycle instead of current clock cycle. The reason is that flip-flop consists of two latches. Value AB is stored into the left hand of the latch at the "current clock cycle" and will only be detected when value shifts to right hand of the latch.

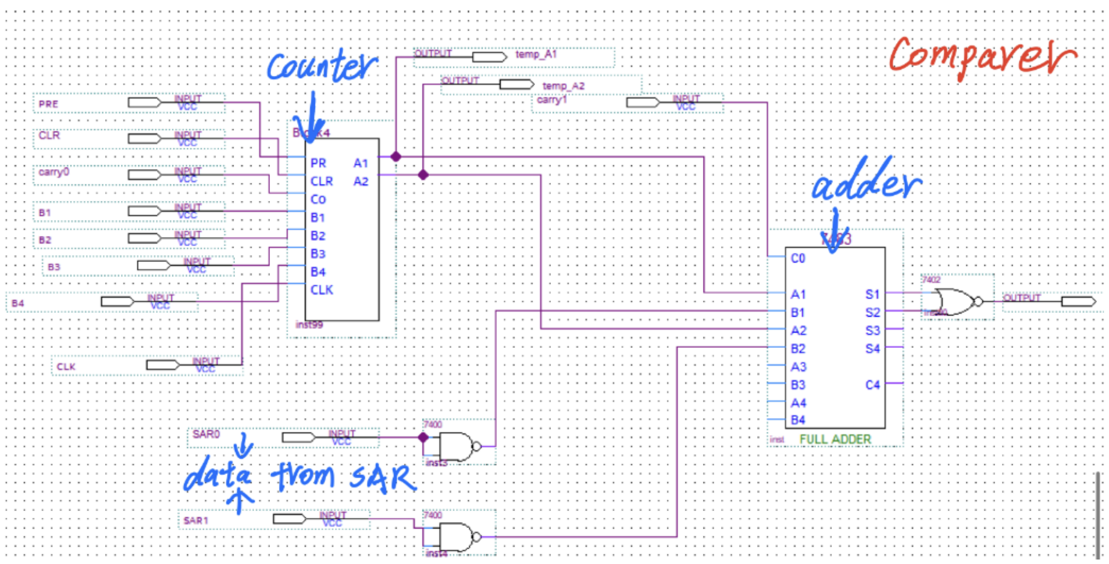
5.3 Detailed Circuit Schematic







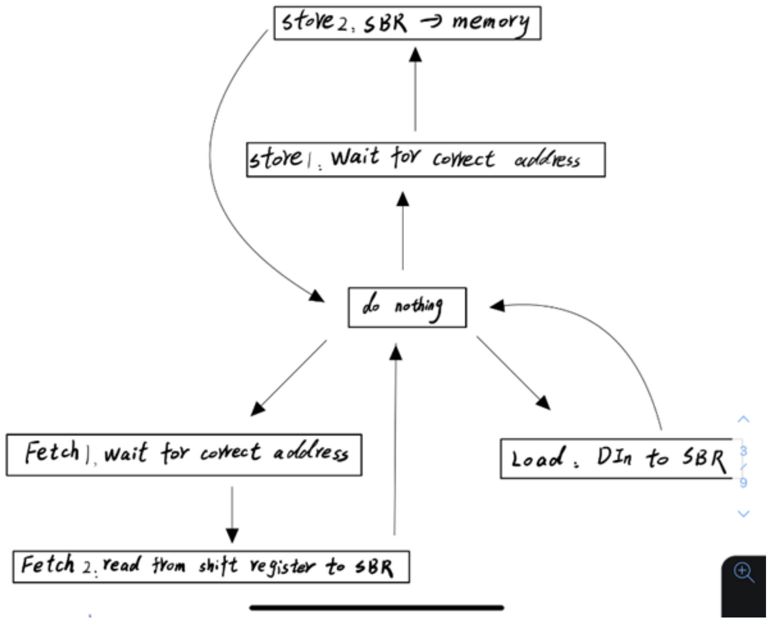
Counter
PR
CLR
C0
B1
B2
B3
B4
CLK
7483
C0
A1
B1
A2
B2
A3
B3
A4
B4
S1
S2
S3
S4
C4
FULL ADDER
7474
1PRN
1D
1CLRN
1CLK
2PRN
2D
2CLRN
2CLK
1Q
2Q
D FLIP-FLOPS
A1
A2



Comparer
Counter
adder
PRE
CLR
carry0
B1
B2
B3
B4
CLK
PR
CLR
Co
B1
B2
B3
B4
CLK
A1
A2
temp_A1
temp_A2
carry1
C0
A1
B1
A2
B2
A3
B3
A4
B4
S1
S2
S3
S4
C4
FULL ADDER
OUTPUT
SAR0
SAR1
data from SAR

6, all bugs encountered, and corrective measures

1, At first, we want to build a FSM having states below:


store2: SBR → memory
store1: wait for correct address
do nothing
Fetch1, wait for correct address
Fetch2: read from shift register to SBR
Load: DIn to SBR

And we choose the “select” and “two bits select” signals(control the muxs) to be part of our state number. In other word, we add flip-flops between the signals and two muxs. This cause a one clock delay. So, when signal finally change the output of the mux, we actually choose the address x+1(x is what we want). The solution is just delete the filp-flop.

2. the second one is not a bug, but a mistake when running our self-design waveform simulation. take FETCH as example, when we want to read the data with address given by SAR, we set FETCH to 1, however, we didn’t notice we set it to 1 for too short time so that the counter didn’t have chance to match with SAR. so, always set the signal we want to debug with longer than 4 clock cycle in lab2 to make sure counter have a chance to match.

3， The delay between detecting a signal and store, when we use the STORE mode, when the address in SAR matches our counter, store operation starts, however, actually, it finishes one clock cycle later. for example, if we set address to 01, that is, we want to store our data in address 01, when store finish, actually it stores at 10, as shift register shift one bit during the time. one method is change a little to our combination logic at the output of counter and SAR, however, actually this delay will not effect our using of this memory storage, as all data address in shift register are defined by our self. if one address corresponding to one data (we can correctly read it), it won't cause trouble.

7, Conclusion

7.1 summary

In this lab, we construct a simple 2-bit, four-word shift-register storage unit which supports three operations: LOAD,WRITE,STORE. To build this, we first construct a counter to store current address and then a comparer to compare the current address with SAR. At last, we use this comparer with several input to build a control logic to control the 2-to-1 and 4-to-2 muxs. By controlling two muxs, we can control when to store or read data into shift reregister so that fit the requirement of the three operations.

7.2 Questions

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

SRAM can read or write data in a very short time compared with shift register. For SRAM, we can directly access data with specified address. However, for shift register, we have to wait for a long time until the current address is same as the SAR.

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

SN7493 4-bit Ripple Counter
SN74LS169A 4-bit Synchronous Up-Down Counter
SN74193 4-bit Synchronous Up-Down Counter
SN74LS194A 4-bit Bidirectional Universal Shift Register
SN5495A parallel-access shift register

In this lab, we choose to use an adder and flip-flops to build the counter. In fact, it is unnecessary for the counter we build is more complex than the counter provided. If we have to choose the one of the provided counters, we will choose SN74193 4-bit Synchronous Up-Down Counter. It is easy to use for we only need to connect its clear and up signal. Also, if we want to change our design later, this counter can provide more functions than a Ripple Counter provides.

We choose the 4-bit Bidirectional Universal Shift Register. This is because in this lab, for shift register, we only need to continue shifting the data without any other operations. As a result, we can just set input S1S0 to 01(meaning shift right) and give a clock and clear signal.

3, Only the clock input needs to be de-bounced in order toto strep through your circuit (why?).

If we do not de-bounce clock input, our circuit may experience several clock cycles without noticing. Other input has to stay in high at rising edge to influence the circuit. In other words, if the input return to lo before the rising edge, it will not affect anything.