Lab 2 Report

by Changjun Lim and Sungwoo Park

< <u>Input Conditioner</u> >

Input Conditioning Test

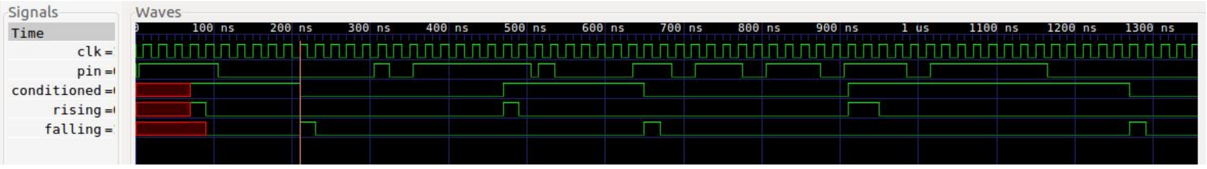


Fig.1: Waveform of Input Conditioner Testing

In this waveform, we can check that the ‘conditioned’ output only changes at the positive clock edge.(Input Synchronization) And short glitches on input does not affect ‘conditioned’.(Input Debouncing) The rising edge and falling edge of ‘conditioned’ are well detected in ‘rising’ and ‘falling’ signals (Edge Detection).

We tested the maximum length of glitch suppressed by the input conditioner. At the length of 70 ns, glitch is recognized as input. Since input conditioner recognizes input which is kept during 4 clock cycles, the theoretical maximum length is 79 ns(the value smaller than 80 ns).

Circuit diagram of structural input conditioner

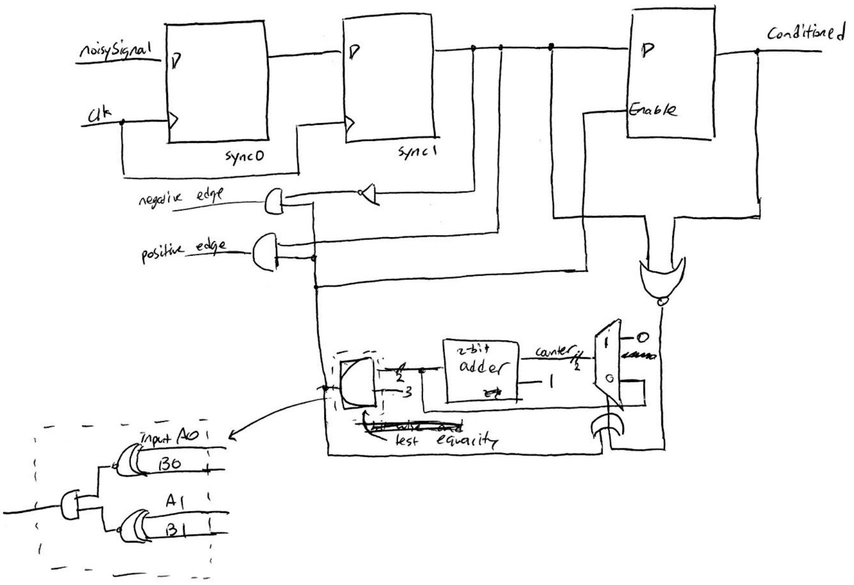


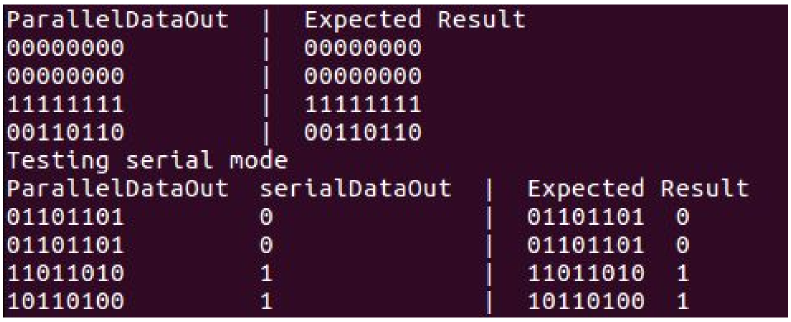
Fig 2: Circuit diagram of structural input conditioner

Maximum length input glitch analysis

If we have a system clock cycle of 50MHz and waittime of 10, then the maximum length of input glitch this system can handle is 200 ns. This is 10 times the length of the duration of a single clock cycle, which is (1 second divided by 50 * 10^6 = 20 * 10^9 sec) 20ns.

< Shift Register >

Shift Register Test Strategy



```
ParallelDataOut    |  Expected Result
00000000           |  00000000
00000000           |  00000000
11111111           |  11111111
00110110           |  00110110
Testing serial mode
ParallelDataOut  serialDataOut   |  Expected Result
01101101         0               |  01101101  0
01101101         0               |  01101101  0
11011010         1               |  11011010  1
10110100         1               |  10110100  1
```

Fig. 3: Shift Register test output

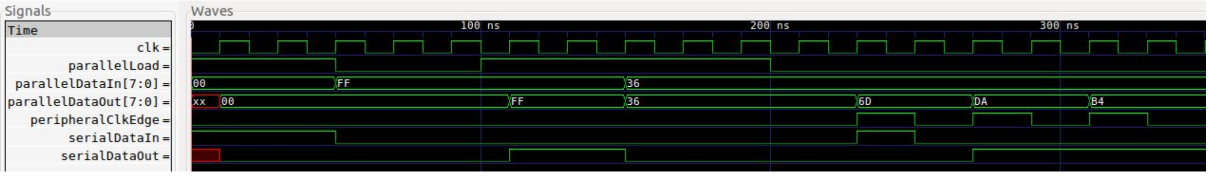


Fig. 4: Waveform of shift register testing

We test the parallel and serial mode of the shift register. In the parallel mode test, we input three value(00000000, 11111111, 00110110) and confirm that we get the same value from the parallel output of the shift register. When we change ‘parallelLoad’ signal to 0, we confirm that parallel output does not change even when the parallel input changes.

In the serial mode test, we check that only when ‘peripheralClkEdge’ has an edge, the shift register advances one position and loads the serial input bit. Parallel output and serial output show the result that we expect.

Finite State Machine Diagram

Refer to Appendix A at the end.

< SPI Memory Test Strategy >

1. Write ‘10100111’ to address ‘1111110’
2. Read data from address ‘1111110’
3. Write ‘01000011’ to address ‘1100010’
4. Read data from address ‘1111110’
5. Read data from address ‘1100010’
6. Read data from address ‘1111111’

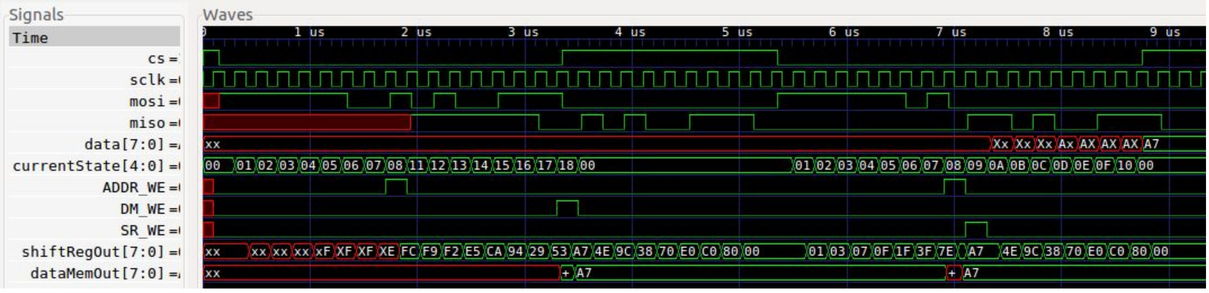


Fig. 5: Waveform of SPI Memory for test case 1 and 2

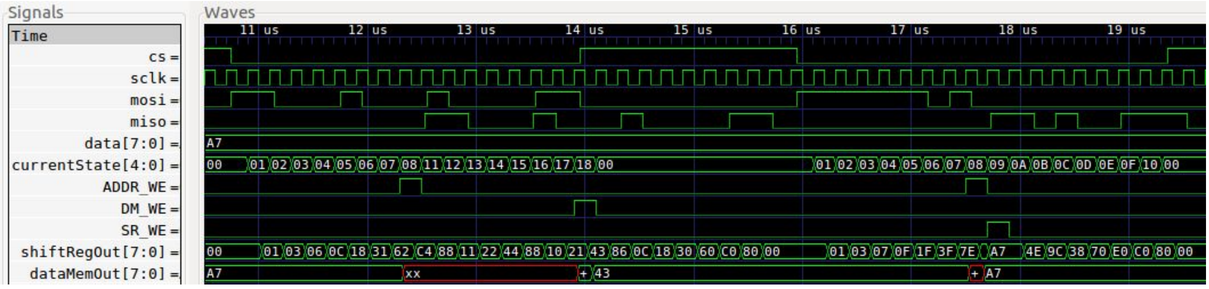


Fig. 6: Waveform of SPI Memory for test case 3 and 4

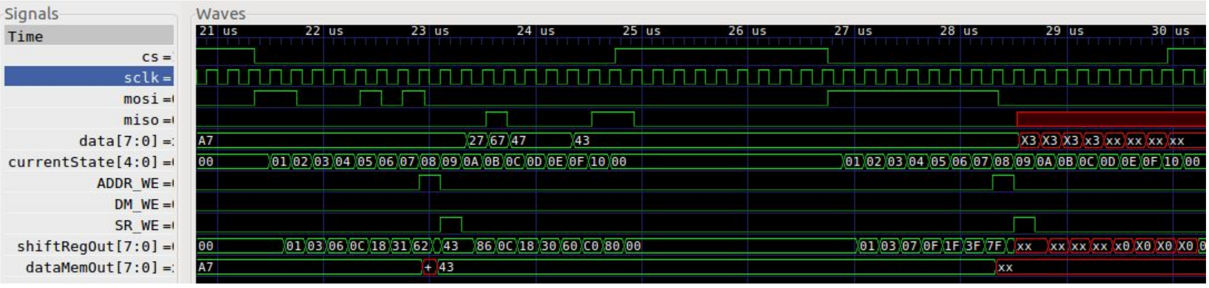


Fig. 7: Waveform of SPI Memory for test case 5 and 6

We chose test cases that can cover the two main behaviors of the SPI memory: read and write. We first started out by writing data to a specific address and reading it from the same address, confirming that we got the value that we just wrote. We then wrote and read from the different address, checking that our SPI memory can read from and write to different addresses. Lastly, we read data from the first address we wrote data to, checking that our SPI memory correctly retains data. If attempting to read data from a memory address that we have not written yet, we got an undefined value, which is an expected behavior.

< Work Plan Reflection >

Scheduled work plan

- Input Conditioning (4.5 hrs) ~10/28
- Shift Register (3.5 hrs) ~10/29
- Midpoint Check In (1.5 hrs) ~10/25
- SPI Memory (3.5 hrs) ~10/30
- Write Report (1.5 hrs) ~11/1

Total 14.5 hrs

Actual time spent

- Input Conditioning (3 hrs) ~10/30

    - complete module in inputconditioner.v (1.5 hrs) ~10/25
    - make test bench (1 hr) ~10/30
    - waveform (0.5 hrs) ~10/30
- Shift Register (1.5 hrs) ~10/31
    - complete module in shiftregister.v (1.5 hrs) ~10/25
    - make test bench (1hr) ~10/31
- Midpoint Check In (2 hrs) ~10/25
- SPI Memory (8 hrs) ~11/1
    - complete SPI memory (1 hrs) ~11/1
    - make test sequence (3 hrs) ~11/1
    - revise code (4 hrs) ~11/1
- Write Report (1 hrs) ~11/2

Total 15.5 hrs

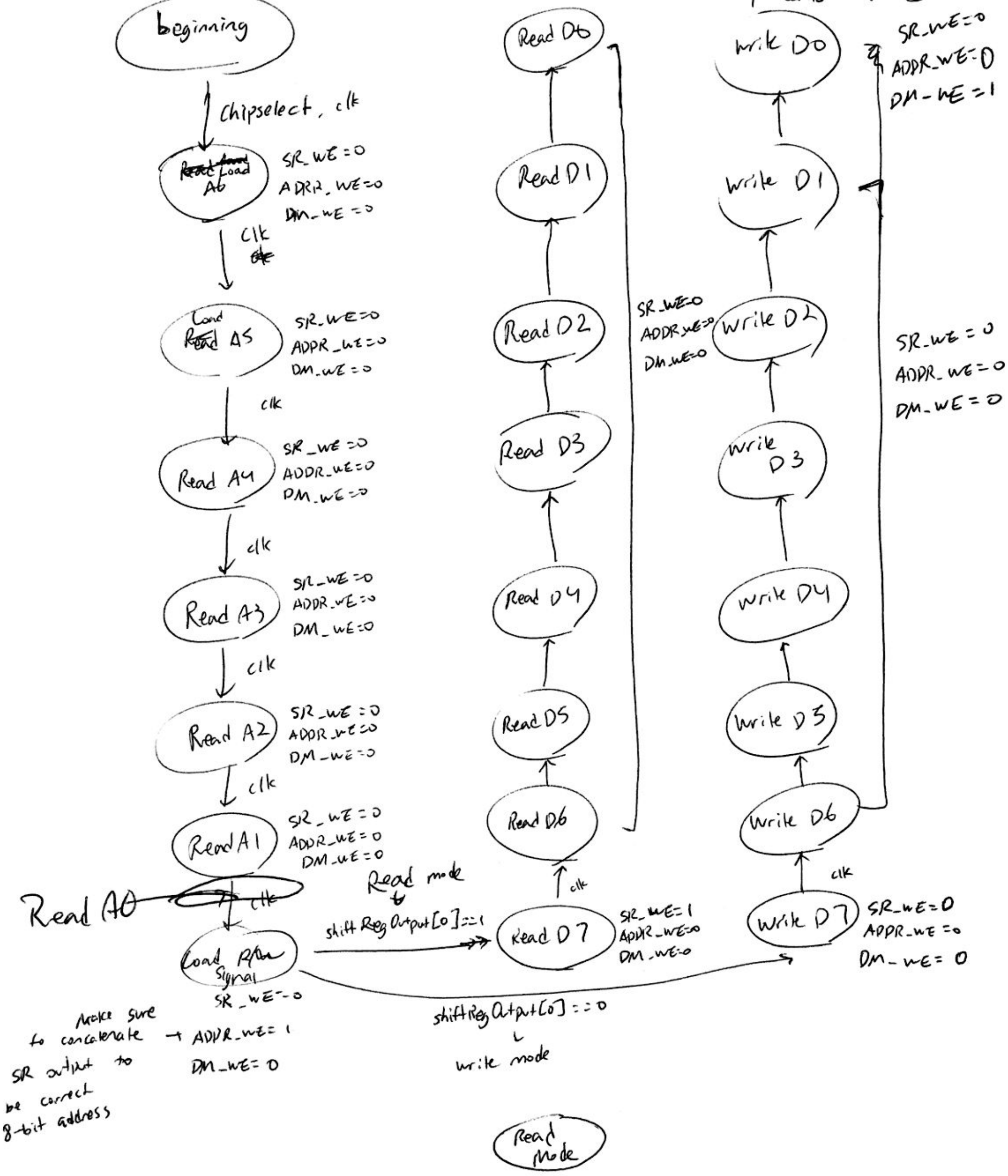
The total time we actually spent(15.5 hrs) is very similar to the total time scheduled for Lab 2(14.5 hrs). It took much more time on ‘SPI Memory’ part, especially making test sequence and revising the code.

Appendix A: Finite State Machine Design

Lab 2 Finite state machine

✲: Serial Data In → ~~[illegible]~~ bit loaded into least significant bit.

Serial Data out → presents MSB



Make sure to concatenate SR output to be correct 8-bit address