CompArch Lab 1

William Lu, Kathryn Hite, & Ian Hill

October 27, 2016

1 Introduction

2 Input Conditioner

2.1 Circuit

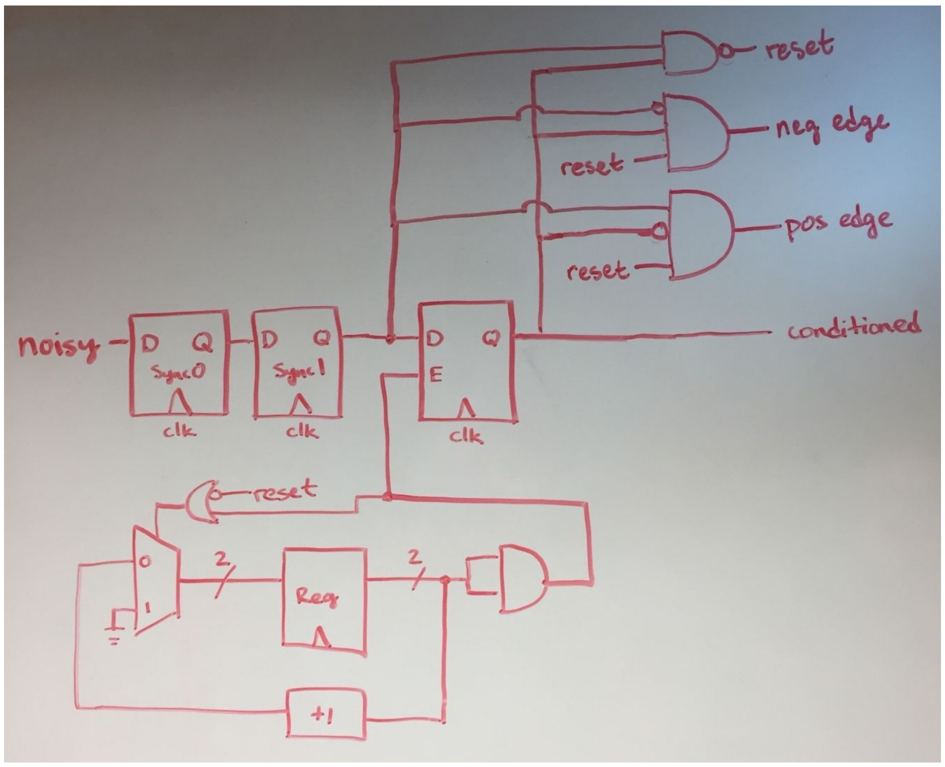


Figure 1: Input conditioner circuit interpreted from our behavioral verilog implemention

2.2 Testing

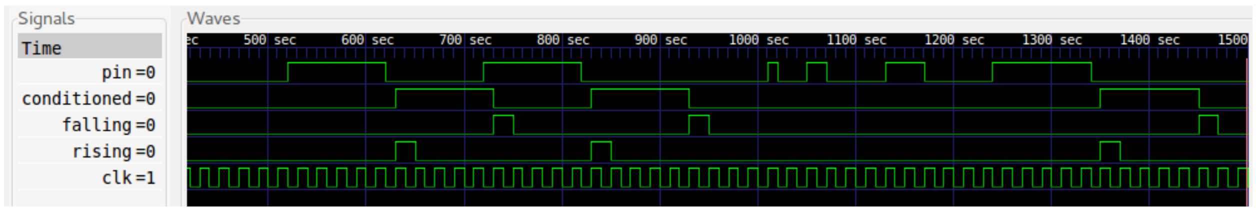


Figure 2: Resulting waveforms from the tests of our input conditioner

To test our input conditioner, we visualized its operation using GTKwave. We input a noisy signal into the input conditioner and checked for whether or not the noise propagated to the conditioned signal output of the input conditioner. Through GTKwave, we could also tell whether or not the rising or falling edge detection of the input conditioner was working. The resulting waveforms matched what is expected from an input conditioner.

2.3 Timing Analysis

Given a 50MHz system clock, where every clock cycle takes 20ns, and a wait time of 10 clock cycles, this design will suppress an input glitch of 200ns. The length of the glitch suppression is determined by the counter guarding the conditioned output which only passes through the output of the dual-ranked D flip flops every 10 clock cycles.

3 Shift Register

3.1 Circuit

The shift register exhibits four main behaviors that are performed synchronously with the main system oscillator clock, which runs at 50MHz.

Behavior one checks to see if the peripheral clock has an edge. If so, the SerialDataIn is placed at the least significant bit position, while all of the original data is shifted up one position. The second behavior assigns the entire shift register memory to the value of parallelDataIn when parallelLoad is high. Behavior three will output the most significant bit of the shift register memory. Finally, the fourth behavior outputs the entire shift register memory. The first two behaviors run based on input variables, while the second two output data in any case.

3.2 Testing

To test this module, we check the first two behaviors individually to test their enable inputs. Both are then enabled to check that only behavior one is called when both pins are high. The following tests a case fully shifting across all bit to verify that the shift register behavior persists over time. The test bench outputs are as follows:

```
------------------------
SHIFT REGISTER TEST CASES
------------------------
peripheralClkEdge parallelLoad | parallelDataIn serialDataIn | parallelDataOut serialDataOut
        0              1       |    10101010         1       |    10101010          1
peripheralClkEdge parallelLoad | parallelDataIn serialDataIn | parallelDataOut serialDataOut
        1              0       |    10101010         1       |    10101010          1
        1              0       |    10101010         1       |    01010101          0
        1              0       |    10101010         1       |    10101011          1
        1              0       |    10101010         1       |    01010111          0
        1              0       |    10101010         1       |    10101111          1
        1              0       |    10101010         1       |    01011111          0
        1              0       |    10101010         1       |    10111111          1
        1              0       |    10101010         1       |    01111111          0
        1              0       |    10101010         1       |    11111111          1
peripheralClkEdge parallelLoad | parallelDataIn serialDataIn | parallelDataOut serialDataOut
        1              1       |    00000000         1       |    00000000          0
```

4 SPI Memory

4.1 SPI Structure

The SPI memory module is constructed using the components and connections shown in Figure 3. This scaffolding utilizes the shift register we previously created, and

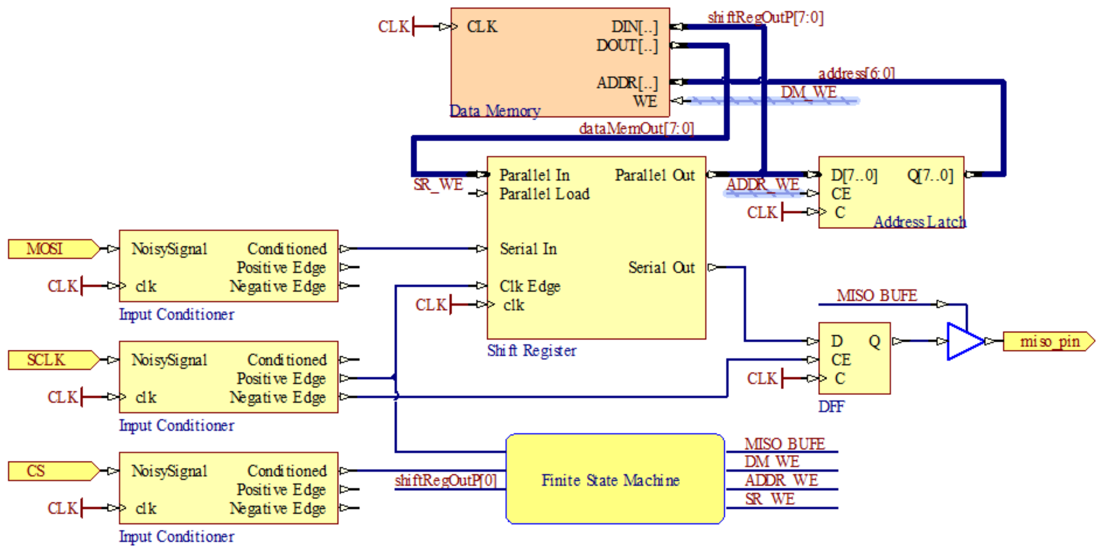
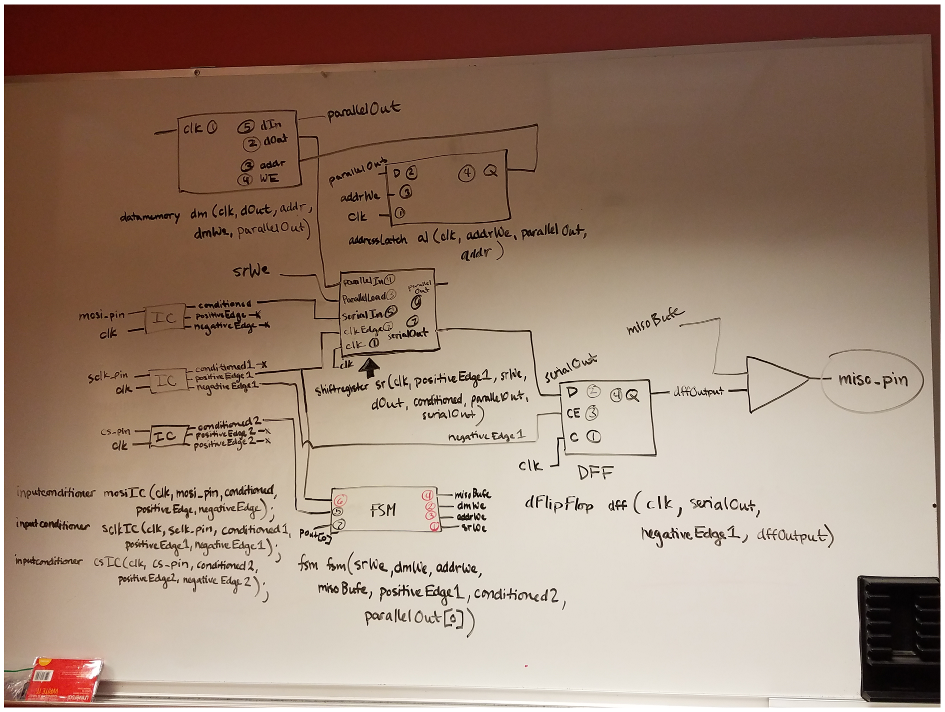


Figure 3: SPI structure layout

Our verilog SPI module was based off of the structure layout as shown in Figure 4. Every module call is directly correlated with the variables used in our module.



parallelOut
datamemory dm (clk, dOut, addr, dmWe, parallelOut)
addressLatch al (clk, addrWe, parallelOut, addr)
srWe
mosi_pin
sclk_pin
cs_pin
IC
shiftregister sr (clk, positiveEdge1, srWe, dOut, conditioned, parallelOut, serialOut)
serialOut
misoBufe
dffOutput
miso_pin
DFF
dFlipFlop dff (clk, serialOut, negativeEdge1, dffOutput)
FSM
inputconditioner mosiIC (clk, mosi_pin, conditioned, positiveEdge, negativeEdge);
inputconditioner sclkIC (clk, sclk_pin, conditioned1, positiveEdge1, negativeEdge1);
inputconditioner csIC (clk, cs_pin, conditioned2, positiveEdge2, negativeEdge2);
fsm fsm(srWe, dmWe, addrWe, misoBufe, positiveEdge1, conditioned2, parallelOut[0])

Figure 4: Layout correlating the SPI diagram to our module code structure.

4.2 SPI Testing Strategy

4.3 SPI Test Output

The test output values we are getting are currently incorrect after following the structure described in the lab for both the FSM and SPI modules. We will update to the correct output after checking in with ninjas on Monday.

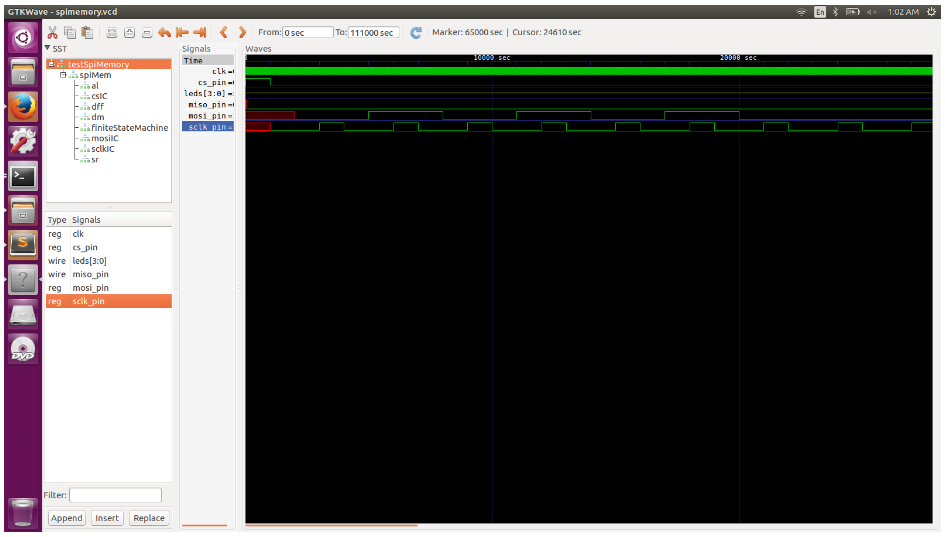


Figure 5: Layout correlating the SPI diagram to our module code structure.

To test the SPI memory, we first wrote a number to an address within the memory and then read that same address to check whether or not we correctly stored the number at that memory address.

5 Work Plan Reflection

5.1 Work Plan

```
-------------------------
CompArch Lab 2 Work Plan
-------------------------

(5) Input Conditioning
   (1) Complete module
   (1) Complete test bench
   (1) Complete test script
   (1) Circuit diagram for structural circuit of input conditioner
   (1) Timing analysis of glitch suppression
(5) Shift Register
   (2.5) Complete module
   (1.5) Complete test bench
   (1) Describe test bench strategy
(2) FPGA Implementation

--------Midpoint--------

(1) Create SPI Memory
```

```
(1) FPGA Implementation
(1) SPI Memory Testing
(3) Report
```

5.2 Reflection

The total times spent on each portion of the work plan were approximately as follows. In general, our implementing our input conditioner took longer, implementing our shift register took less time, and implementing our SPI memory took longer.

```

------------------------
CompArch Lab 2 Work Plan
----- Actual Times -----
------------------------


(6.75) Input Conditioning
   (3) Complete module
   (2) Complete test bench
   (.25) Complete test script
   (1) Circuit diagram for structural circuit of input conditioner
   (.5) Timing analysis of glitch suppression
(3.5) Shift Register
   (1) Complete module
   The verilog for this module was more trivial than we estimated, though some unknown
       procedural verilog attributes took time to debug
   (2) Complete test bench
   (0.5) Describe test bench strategy and complete report section
(0) FPGA Implementation

--------Midpoint--------

(4) Create SPI Memory
(0) FPGA Implementation
(4) SPI Memory Testing
(2) Report
```