CompArch Lab 3

Corey Cochran-Lepiz
Kristtiya Guerra
Adi Sudhakar

Novemeber 17, 2019

1 Introduction

The goal of this lab is to create Serial Peripheral Interface (SPI) memory and instantiate it onto an Zybo FPGA board.

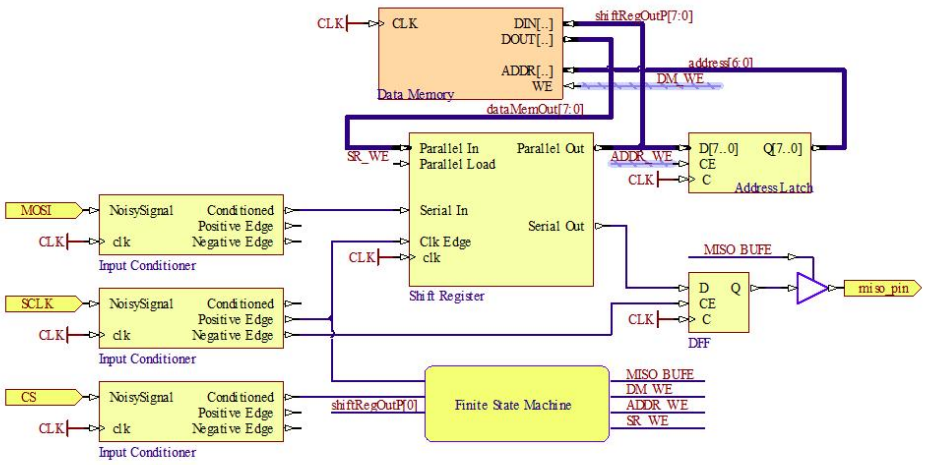

Figure 1: Block Diagram of SPI

2 Implementation

2.1 Input Conditioning

The input conditioning is meant to clean input signals and send out 3 signals to the main shift register- the clean conditioned signal, a positive signal at the positive edge of the conditioned signal, and a positive signal at the negative edge of the conditioned signal.

The input is controlled by a button or switch. This module works at the clock cycle and whenever a button is pressed the user isn't ensured they will press right at the clock cycle. In order to combat this, input synchronization needs to be implemented. This method uses a pair of D flip-flops that take the signal and synchronize it. There is also a counter that sets a certain amount of clock cycles before the module will identify the input as a real signal and not noise.

This module also produces 2 additional outputs: One that is dedicated for indicating the falling edge of the conditioned output and one dedicated for indicating the rising edge of the conditioned output. The signals

each only last for a single clock cycle.

2.1.1 Input Conditioner Testbench

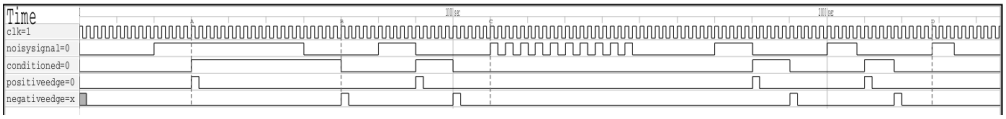

Figure 2: GTK Wave of the Input Conditioner Testbench

At marker A, the positive edge is triggered at the very start of the conditioned output for a single clock cycle.

The Input Conditioner test bench tests if the module correctly interprets when a signal is being sent in. It should ignore signals that are shorter than the specified wait-time (3 clock cycles). It also should only give signals at the positive and negative edges for the conditioned signal.

2.2 Shift Register

The shift register is the core functional unit of SPI memory - at every positive clock edge, the shift register sends out the MSB of its 8 bit storage and takes in a new bit from serialIn, in this case, conditioned MOSI input. Depending on the MOSI instruction, the FSM will dictate whether the shift register will keep its current contents or parallel load a new set of values from the data memory.

2.3 State Machine

The state machine is a large decoder that interprets the MOSI instruction from the shift register and dictates whether the SPI memory is reading from its data memory or writing to its data memory.

2.4 Output w/Enable

If the MISO buffer is enabled, then the NOT control is enabled. The input value will be output. If the buffer is not enabled, then the input value will not be output.

2.5 Address Latch

The Address Latch sends an address to data memory. The Address Latch takes in parallel output from the Shift Register, an address write enable from the FSM, and works at the positive edge of the clock. It takes the last 7 bits of the inputted data and excludes the MSB.

2.6 D Flip-flop with enabler

The d flip-flop with enable takes in 3 inputs - the clock, an enable signal, and input signal. When the enable signal is high (in our case, negative edge of the SCLK) the data in will be pushed to the output. If the enable is not high, then the current data that is held in the d flip-flop will remain being the output and not be replaced.

2.7 Loading to the FPGA Board

Loading the SPI memory onto the FPGA board turned out to be more of a challenge than initially expected.

2.8 SPI Memory

The SPI memory compiles all of the modules together. As shown in figure 1, it takes in 3 inputs plus a clock. The 3 inputs are the SPI clock, the MOSI pin, and the Channel Select pin. Its output is the MISO pin.

3 SPI Testing

Overall, it takes about (150) clock cycles for the MISO to output any sort of value.

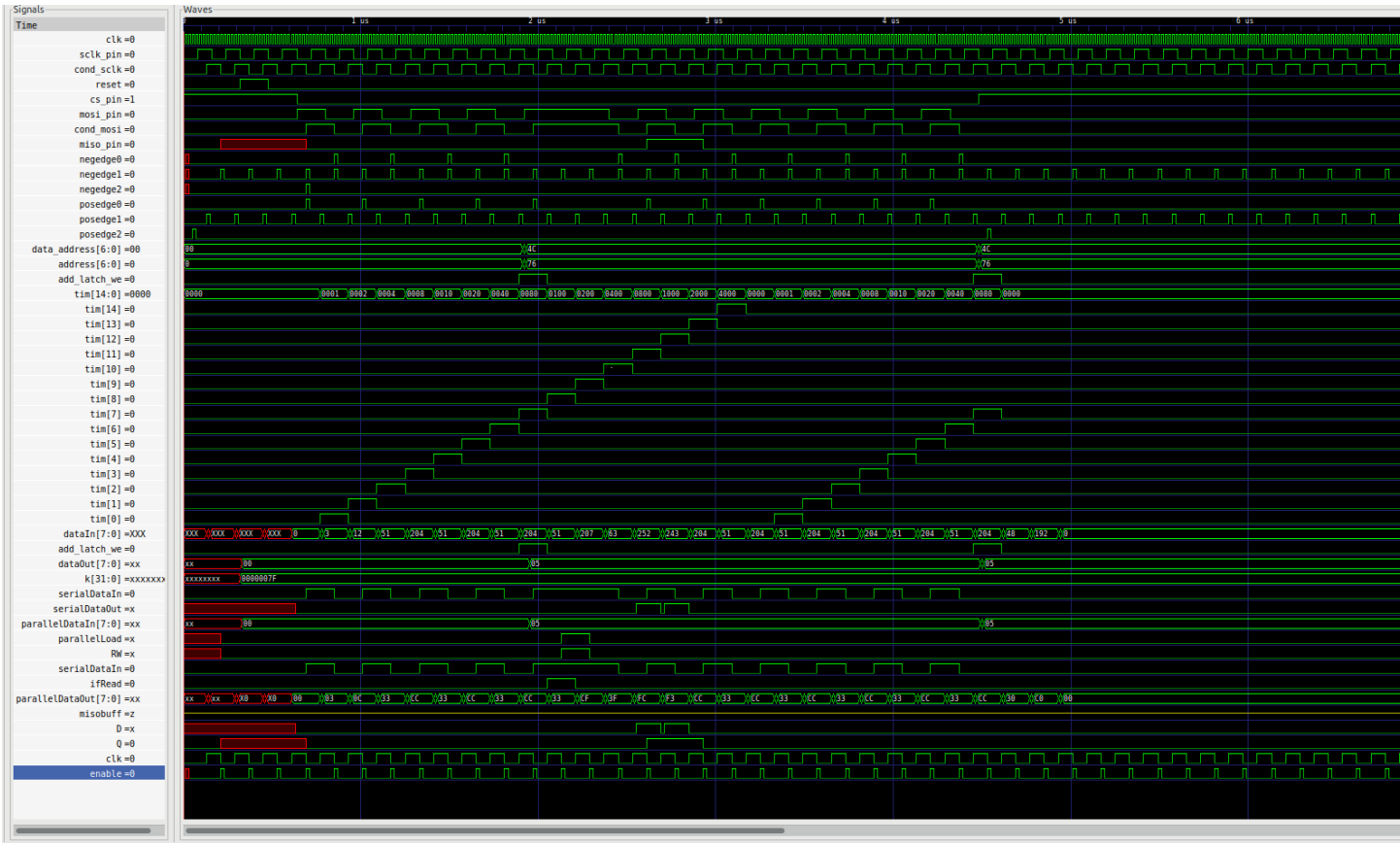

Figure 3: Final waveform of our SPI testbench