Tara Moses

010630228

Mondays 4:10 – 5:55

Lab #3: Register File

February 22, 2016

**Introduction**

The objective of this lab was to familiarize students with a register file for a 16-bit register. The register was treated as an array with 16 slots, since the addresses taken into the register file were 4 bits and the decimal number range that 4 bits can represent in binary is [0, 15].

**Approach**

The register file was meant to take in a 16-bit input, a\_data, and load that input into the specified register address, a\_addr, for it when the load input was high and the clock’s edge was rising. The register file also outputted the data stored in two register addresses, b\_addr and c\_addr, to 16-bit outputs b\_data and c\_data, respectively. A\_addr, b\_addr, and c\_addr were all 4-bit inputs. The register also implemented an asynchronous clear which, regardless of the clock edge, set all registers to 0, except for register 1 which stayed set to “0000 0000 0000 0001”. Register 0 was set to “0000 0000 0000 0000”, and it (as well as register 1) did not change even when a\_addr was set to load a\_data into those registers.

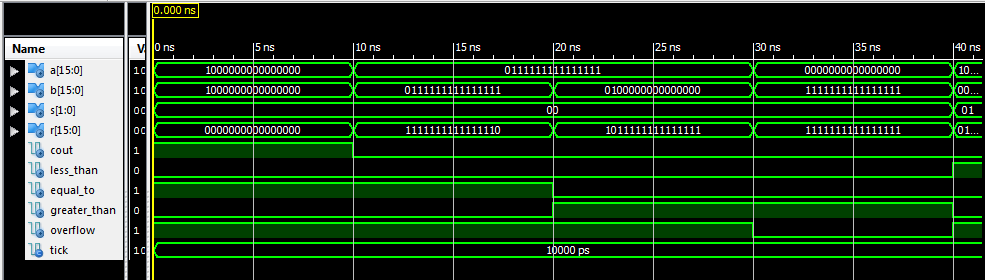
**Experimentation**

The register file’s register was an array of size 16. Register 0 was set to all 0s. Register 1 was set to “0000 0000 0000 0001”. Every time load was high and a\_data was being loaded into register a\_addr, the values of registers 0 and 1 were reset to the above values. Because of this, registers 0 and 1 couldn’t be changed.

**Results**

The ALU was created, and a test-bench was written to test each of its four functions. Each function was tested with four sets of inputs.

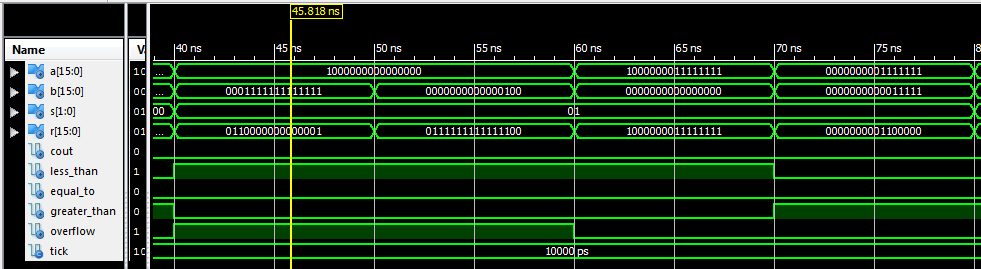
The first function, addition, occurred when the select equaled “00”. The waveform for this function is shown in Figure 1 below.



**Figure 1**: waveform for ALU addition

For the first part of this waveform, two very large negative numbers were added together. There was carryout and overflow. The two numbers were equal to each other. For the second part, there was no carryout, but there was overflow. For the third part, the first input was more than the second input. There was no carryout, but there was overflow. For the fourth part, the second number was added to 0. There was no carryout or overflow.

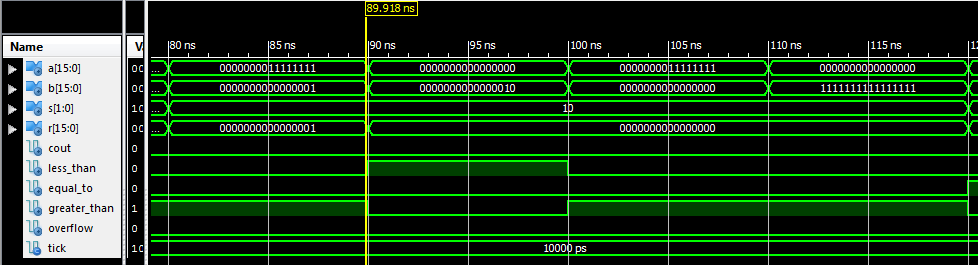
The second function was subtraction. The waveform for this function is shown in Figure 2 below.



**Figure 2**: waveform for ALU subtraction

For the first test case of this function, a large positive number was subtracted from a large negative number. There was no carryout, and the first number was less than the second one. There was overflow. The second test case involved a small positive number subtracted from a large negative number. There was overflow, but no carryout. The third test case subtracted 0 from a negative number. The result was the same negative number. No carryout, no overflow, and the first number was less than the second one. The fourth test case involved a smaller positive number subtracted from a larger one. There was no carryout or overflow.

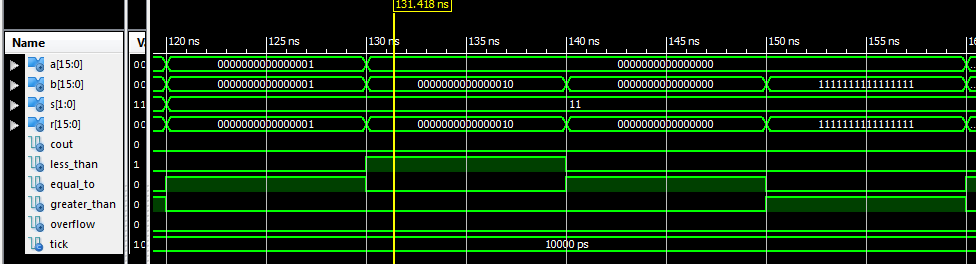
The third function was the logical AND function. The waveform testing this is shown in Figure 3 below.



**Figure 3**: waveform for ALU AND function

This function had no carryout or overflow. The first test case for this function anded two nonzero numbers together. The result was 1. The next three test cases anded a number with 0, so the result was 0 for each of them.

The last function was the logical OR function. The waveform for this function is shown in Figure 4 below.



**Figure 4**: waveform for ALU OR function

The first, second, and fourth test cases for this function ORed a number with a nonzero number. The nonzero number was the result of each of these test cases. In the third test case for this function, two zeros were ORed together, and the result was 0.

**Conclusion**

The VHDL code for the ALU compiled and worked as expected. The test-bench included helpful test inputs that correctly demonstrated the working carry-out, overflow, result, and greater than/equal to/less than outputs. In this lab, I learned how an ALU works, as well as how to add and subtract signed integers.