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 Arithmetic-Logic Units (ALUs) by writing VHDL code to implement and test a 16-bit ALU with four functions: addition, subtraction, the logical AND, and the logical OR.

**Approach**

The ALU took in two 16-bit vectors as inputs for the functions and one 2-bit selector to choose which function to perform. The result was outputted as a 16-bit vector, and the carry-out for addition/subtraction was outputted as a 1-bit number. Also, three values were outputted indicating whether the first input value was (1) greater than, (2) less than, or (3) equal to the second value.

**Experimentation**

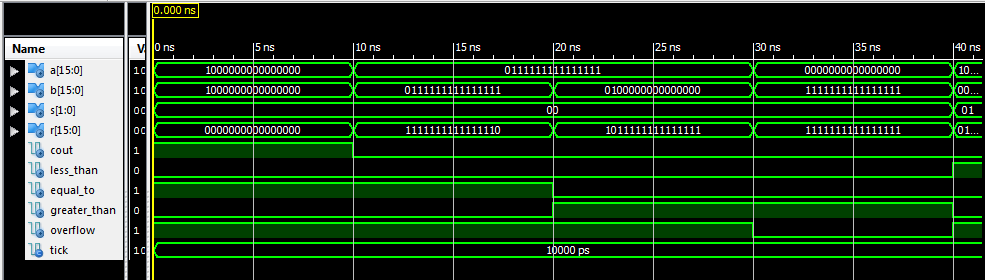
For the ALU, the two inputs were added and subtracted using the ieee.std\_signed.all library. In class, we described the ALU as being similar to a multiplexer. The select was used to choose which function to complete on the inputs. In order to calculate the carry-out, the inputs were changed to 17-bit integers, and the sum and difference of the two were saved as 17-bit signals. The carry-out was assigned to the most significant bit. If there was no carry-out, or the operations didn’t allow carry-out, then the carry-out was assigned to 0.

Whether overflow occurred was also outputted. If the two 16-bit inputs were the same sign and the sum of them had a different sign, then the overflow was set to one. For subtraction, if the first input had the same sign as the not of the second, and their difference had a different sign, then overflow occurred.

**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.

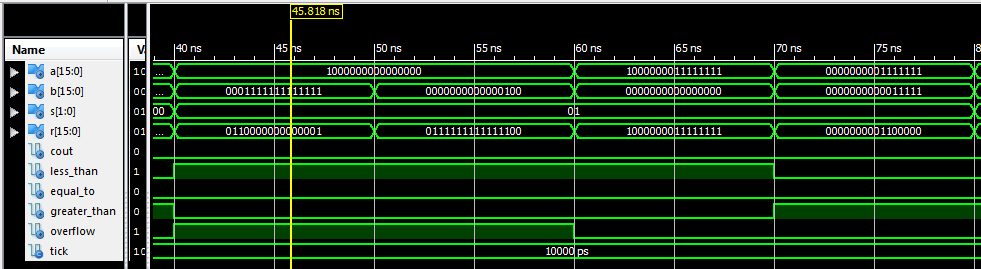
The first function, addition, occurred when the select equaled “00”. The waveform for this function is shown in Figure 1 below.



**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.

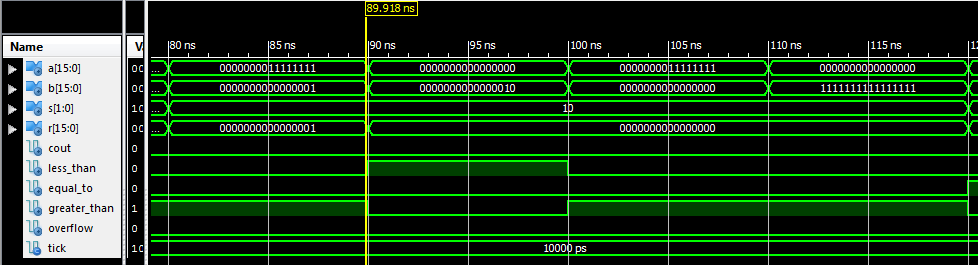
The second function was subtraction. The waveform for this function is shown in Figure 2 below.



**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.

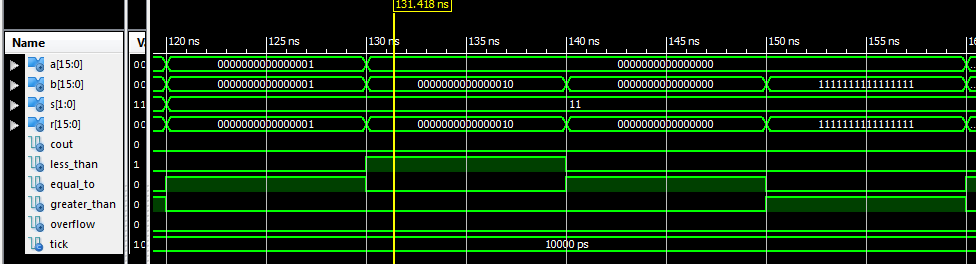
The third function was the logical AND function. The waveform testing this is shown in Figure 3 below.



**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.