Katy Kahla
James Kress
3/17/12

Part 2 – Adder

Design

This part of the project asked us to create a carry lookahead adder that had the ability to add or subtract two 16-bit values. The inputs to our register module were given to us in the project description and included inputA, inputB, and subtract which when high would tell the adder to subtract the two given values. The outputs from the claAdder were AandB, AorB, sum, and carryOut. The claAdder does not have the ability to detect overflow.

We began this portion of the project by creating three modules. We started with the single-bit adder that simply calculated the sum, AandB and AorB values of a single bit. These equations used the logic that was in the class lecture notes. From here, we designed our fourBitAdder module. It is in this module of the project that we calculate all generate and propagate values for the carry lookahead logic.

We considered implementing this logic in the top-level code however; we determined this wouldn't work because the actual delay in calculating these values is the same even when done in the fourBitAdder module. We derived our generate and propagate equations from in class and lecture notes. We also researched CLAs online to insure that we were implementing the logic correctly.

To implement subtract, we look for a high subtract signal in the top module. Originally we attempted to look for this signal in the singleBitAdder but it became clear quickly that this wouldn't work properly as it creates an adder that calculate the values and then added 1 to it. Instead, we placed it in the top module and checked for the high subtract signal first so that the correct values in inputB would be given to the fourBitAdder. To implement this we simply took the inverse of inputB and passed that on to our module.

Schematic

Below, we have included the RTL schematic for the project. The first figure is the top level, and the second schematic is the pushed in view of the top level, showing the major components of the adder file. The third figure shows a pushed in view of the fourBitAdder and the fourth figure shows a pushed in view of the singleBitAdder.

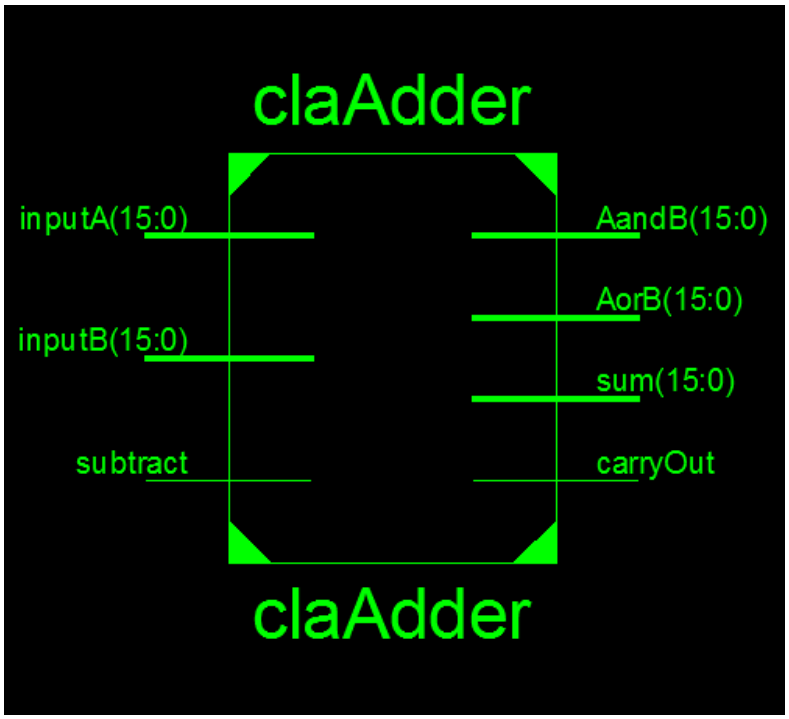

Figure 1: RTL Schematic of Top Level

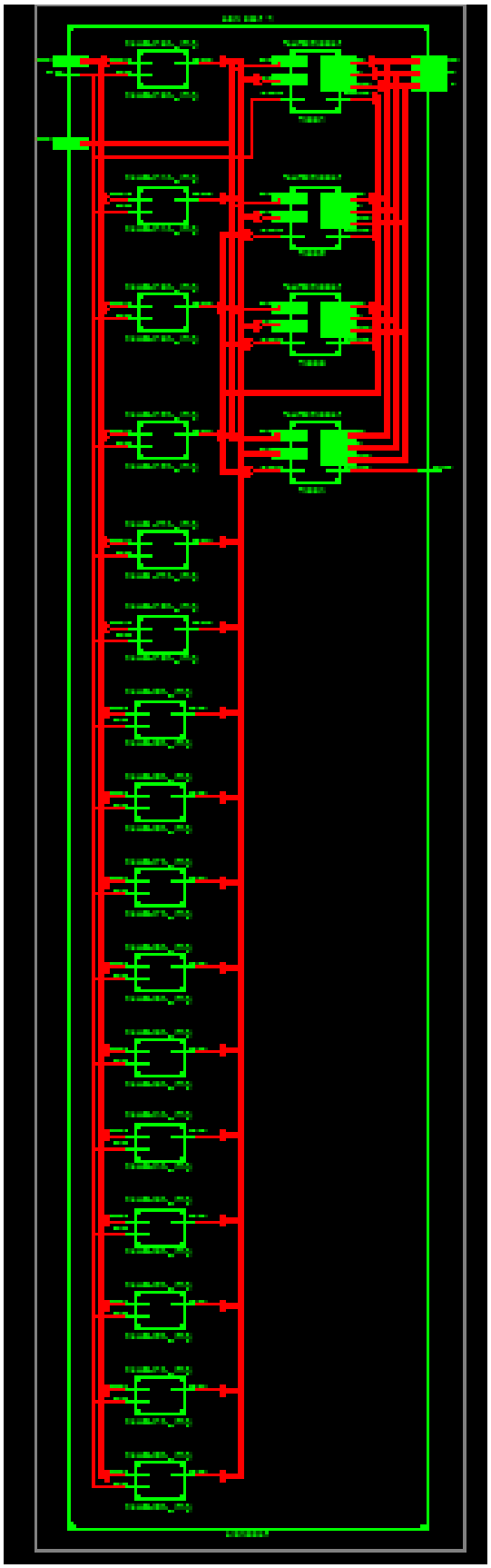

Figure 2: RTL Schematic of pushed in top-level

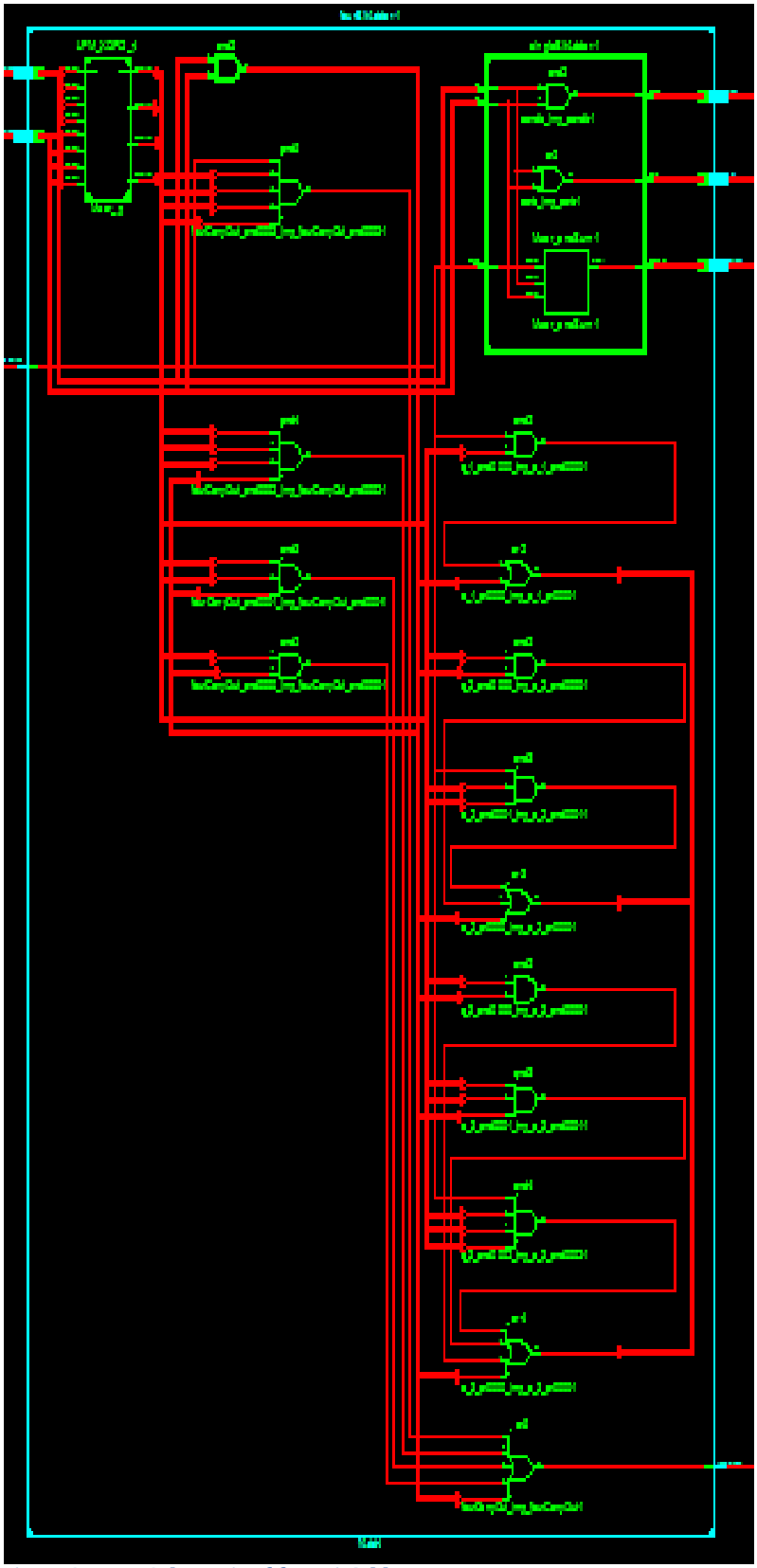

Figure 3: RTL Schematic of fourBitAdder

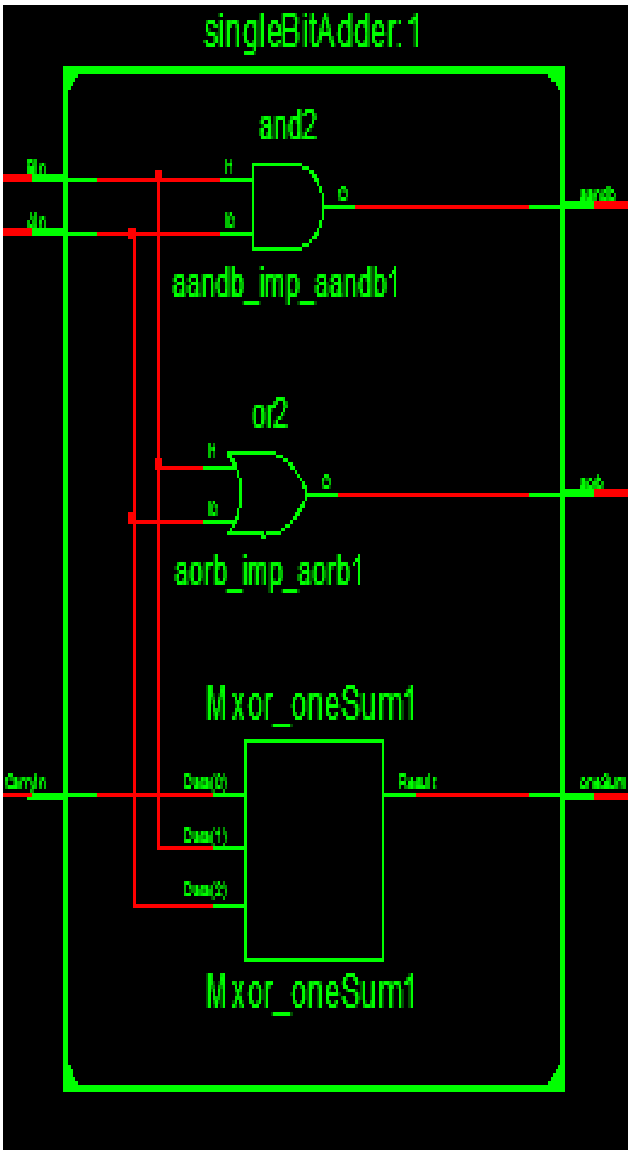

Figure 4: RTL Schematic of singleBitAdder

Testing

We tested out project extensively to ensure that all of the program requirements were met. To start off, we tested using the simple testing script that had been provided with the program. Once we passed all of the tests it contained, we decided to add to the given tests to come up with a file that would test for all of the required functionality.

In the first block of testing, we checked our addition functionality. We attempted to make sure that we tested all edge cases for the addition functionality. Some of the edge cases in this module were adding two 0's, adding the maximum value of 65,535 to 0 and to 1 as well as to itself. We made sure that adding a 0 to anything would give the correct result regardless of what register 0 was in as well as adding 1 to anything including itself. After the edge cases, we tested some general cases including a single digit plus a single digit and double digits plus double digits. The last thing that we realized with addition is that we could actually add a negative number to a positive number (because a negative number is a large positive 16-bit value) and get a correct value. The value is correct because of rollover in the addition. All of our test cases passed without exception.

In the second block of testing, we checked our subtraction functionality. In this portion, we also tested as many edge cases as we could think of. Most of the cases in the section of testing we derived from the addition portion. We tested to make sure the anything having a 0 subtracted from it would retain the same value and 0 minus anything would give that number in a negative representation. We also made sure that a small number minus a larger number would give a negative value.

All tests may be view in the supplied testing module.