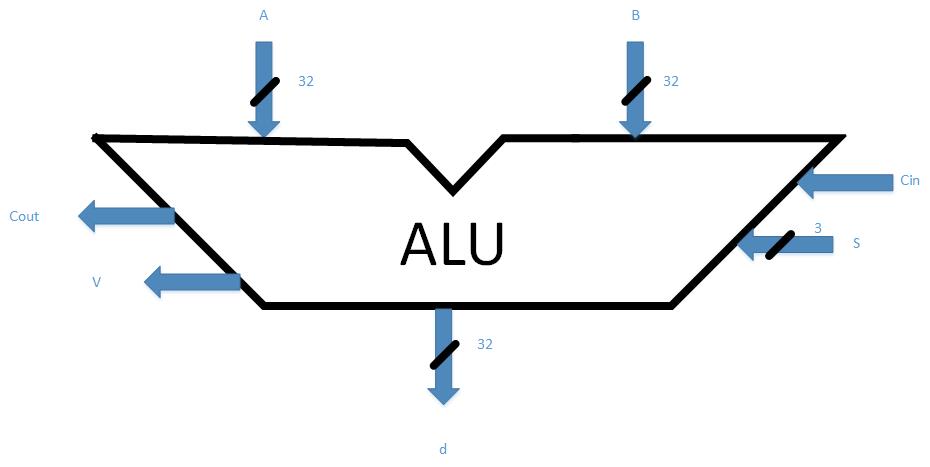
Arjun Gupta

EECE 3324

Lab 1

7/3/2018

1. Description
   1. The purpose of this assignment was to demonstrate how to make a more efficient ALU. This was done by using a Look-Ahead Carry adder structure (LAC) that broke down each Fuller Adder module into a smaller description, and passed information down a chain from one Look-Ahead Carry adder block to another. This resulted in a significantly reduced delay in synthesizing a design – on the order of log N rather than N. The full ALU was built up and simulated using the Vivado IDE, and a testbench was created to run the simulation. Each function of the ALU was thoroughly tested and the expected results matched each given input.
2. Block Diagram
   1. Figure 1 represents a block diagram for the top-level module of the ALU, showing the input/output structure. The input output structure for the ALU is modeled.



1. Steps taken to complete Assignment
   1. The first step taken was to analyze the given structure of the LAC. I looked at the input/output structure of the LAC and implemented the leaves of the “lac3” module. On each iteration of a LAC (LAC0, LAC1, LAC2, …), I noticed that the size of the input/output structure (A, B, and D) doubled as the number of inputs doubled as well. Each LAC has two leafs of the previous iteration and one root. The leaves each have half of the data for a given input or output. For example, LAC3 will have 2^3 number of bits, so 8 bits in total. One leaf of LAC3 (which is a LAC2) will have 4 bits carried over from the previous iteration and the other leaf will have the other 4 bits. The root will propagate the rest of the bits over to the next LAC level until the final root is instantiated.
2. Lessons learned
   1. From this lab, I learned how much faster a LAC structure is compared to a regular ripple-carry or parallel adder structure. The benefits from a LAC structure are definitely evident as you increase your data size, and once you hit 64-bit or higher adders, you will definitely see the improvement. Adders are present throughout the CPU instruction execution process flow, so it is very evident that having efficient data structures like the LAC are important to having a fast and efficient processor.