

Duke ECE152 – Spring 2012 – Project Part 4: ALU

75 Points. Due electronically by 11:59pm on Weds, Feb 22.

In this part of the project, you will implement the integer Arithmetic Logic Unit (ALU) that you will use in your processor.

Project Part 4: Arithmetic Logic Unit

The ALU implements all of the arithmetic and logical operations specified in the Instruction Set Architecture, including addition, 2's complement subtraction, bitwise logical and, bitwise logical or, left shift, and zero-extending right shift. In addition to the above operations, the ALU also generates a signal `isEqual` when both operands are equal to each other; this is used for the `beq` (branch-if-equals) instruction. Lastly, the ALU generates a signal `isGreaterThan` when `data_operandB` is greater than `data_operandA` and the subtract operation is selected (behavior when the subtract operation is not selected is undefined / don't-care); this is used for the `bgt` (branch-if-greater-than signed 2's complement) instruction. Be sure to check for under- / over-flow and generate the correct output when calculating `isGreaterThan`. The ALU receives a 3-bit ALU operation code (ALU opcode) from the processor's control logic that denotes which operation should be performed. The shift operations require only the lower 5 bits of input `data_operandB`.

You will implement your ALU in Structural VHDL using the Quartus II software. You should create one VHDL (`alu.vhd`) or Block-Diagram (`alu.bdf`) file that has exactly the same format as the diagram in Figure 1. This top-level `alu.vhd` or `alu.bdf` file is likely to refer to other lower-level files (e.g., `adder`, `shifter`, etc.). The top-level file `alu.vhd` or `alu.bdf` file is what you will then use later in the semester when you need an ALU for your processor. Figure 1 is a screenshot of the `alu` component in Quartus, and it shows the signal names that you MUST use in your design to facilitate testing and grading.

After implementing your ALU, you should test it thoroughly to verify that it works correctly. One test waveform is provided for your ALU at http://people.ee.duke.edu/~sorin/ece152/project/test_alu.vwf. In addition, this assignment will be graded by running additional tests that are not provided, so do not assume that you can ignore bugs that do not manifest themselves on the one test that is provided.

Submitting This Assignment

To submit this assignment, create a Quartus Archive (Project → Archive Project) named `project4.qar` of all the files needed to implement your design. Make sure that your top-level file is named `alu.vhd` or `alu.bdf`. Names of lower-level files are unrestricted, but be sure to include them along with your top-level design entity in the Quartus Archive file. Email your Quartus Archive file as an attachment along with all group members' names and NetIDs to duke.ece152.spring2012@gmail.com.

Table 1: ALU Opcodes

Operation	ALU Opcode
add	000
subtract	001
and	010
or	011
sll	100
sra	101

Figure 1: alu VHDL and BDF screenshots

