# Duke ECE152 – Spring 2011 – Project Part 4: ALU

75 Points. Due electronically by 11:59pm on Friday, March 4.

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, subtraction, bitwise and, bitwise or, left shift, and arithmetically-sign-extending right shift. In addition to the above operations, the ALU also generates a signal isNotEqual when both operands are not equal to each other; this is used for the bne (branch-if-not-equals) instruction. Lastly, the ALU generates a signal isLessThan when data_operandA is less than data_operandB and the subtract operation is selected (behavior when the subtract operation is not selected is undefined / don't-care); this is used for the blt (branch-if-less-than signed) instruction. The ALU receives a 5-bit ALU operation code (ALU opcode) from the processor's control logic that denotes which operation should be performed. The shift operations require only one input operand, data_operandA.

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 DukeECE152Spring2011@gmail.com.

**Table 1: ALU Opcodes**

| Operation | ALU Opcode |
|---|---|
| add | 00000 |
| subtract | 00001 |
| and | 00010 |
| or | 00011 |
| sll | 00100 |
| sra | 00101 |

**Figure 1: alu VHDL and BDF screenshots**

```
ENTITY alu IS
    PORT (  data_operandA, data_operandB    : IN STD_LOGIC_VECTOR(31 DOWNTO 0); -- 32bit inputs
            ctrl_ALUopcode  : IN STD_LOGIC_VECTOR(4 DOWNTO 0);  -- 5bit ALU opcode
            ctrl_shiftamt   : IN STD_LOGIC_VECTOR(4 DOWNTO 0);  -- unsigned 5bit shift amount
            data_result : OUT STD_LOGIC_VECTOR(31 DOWNTO 0);    -- 32bit output
            isNotEqual, isLessThan  : OUT STD_LOGIC);
END alu;
```