Lab 2 – ALU Design

ENGIN 341 – Advanced Digital Design
University of Massachusetts Boston

Overview

In this lab you will design an Arithmetic and Logic Unit (ALU) that implements 11 functions, as shown in Figure 1. Output will be displayed as hexadecimal values on a seven-segment display.

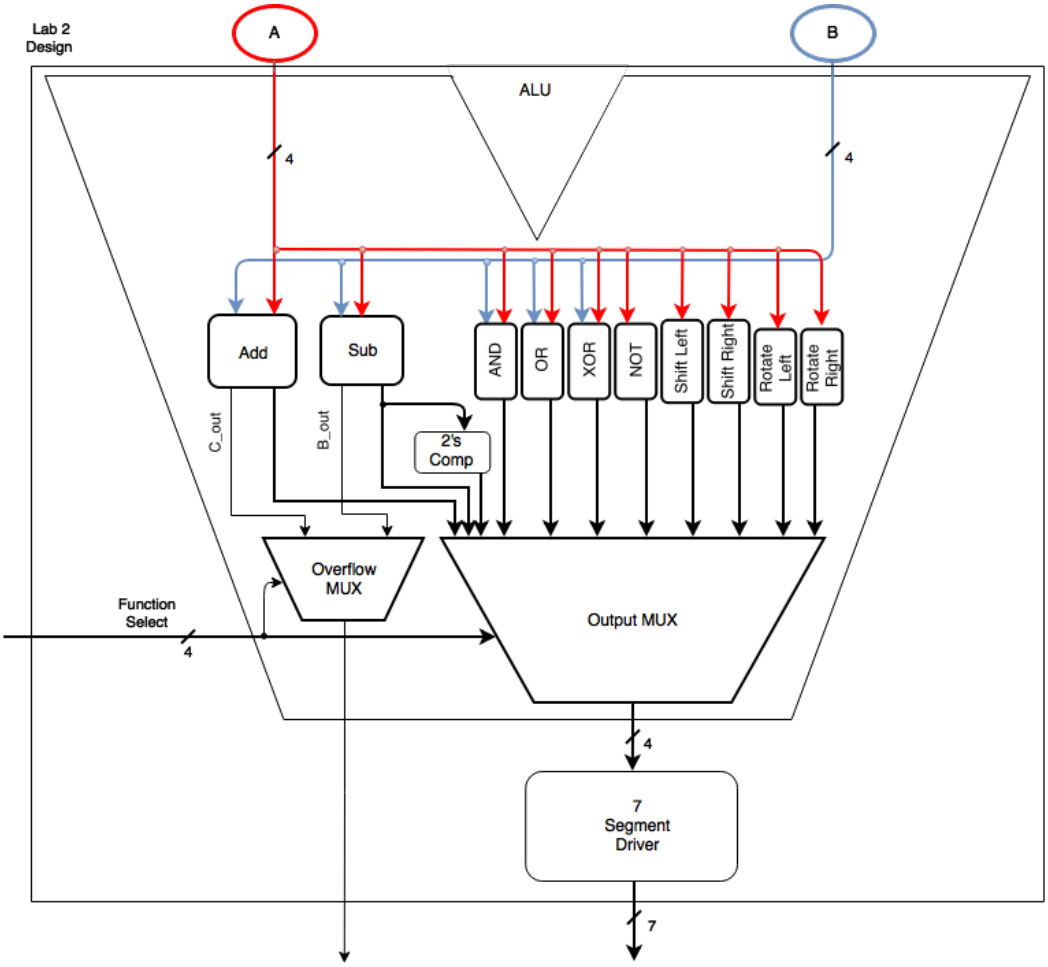

Figure 1: Lab 2 Block Diagram

Lab Work

1. Modify your 2-bit Adder design from Lab 1 to a *N-bit Ripple Carry Adder* using **generic** and **generate** statements.
    a. Write a testbench for the Adder module. Verify functionality before proceeding.
2. Design a VHDL module for an *N-bit Ripple Carry Subtractor*, follow the same design practice as in the design of the N-bit Ripple Carry Adder.
    a. Write a testbench for the Subtractor module. Verify functionality before proceeding.

3. Design a VHDL module for an 11-to-1 multiplexer that will use the *Function Select* input as select. This is the *Output MUX* in Figure 1.
    a. Use this same approach to then design the 2-to-1 MUX for the Add and Subtract overflow. The select line of this MUX should be the least significant bit of the *Function Select* input.
4. You have been given a working module for the *Two's Complement* used within the ALU. Create an *ALU.vhd* file with an *ALU* module that has the same ports as the ALU shown in Figure 1, except that the inputs A and B, and the output R should be defined as generic N-bit wide vectors. Declare components for the N-bit Ripple Carry Adder and Subtractor, the MUX-es and for the Two's Complement modules.
    a. Instantiate each one of the components within the *ALU architecture* once. Connect the outputs of the Adder, Subtractor, and Two's Complement internal functions to the inputs of the Output MUX. Make sure to follow the assignment order shown in Figure 1 (Table 2 in the detailed lab description). Connect the overflow bits of the Adder and Subtractor modules to the 2-to-1 MUX, and make sure to assign the least significant bit of the *Function Select* ALU input as the select input to the 2-to-1 MUX.
    b. For the remaining binary and unary operators, use dataflow assignments to internal *ALU-architecture* signals, and then connect these signals to the output MUX in the order shown in Table 2/Figure 1.
    c. Write a testbench for the ALU. Verify that it works as intended before proceeding.
5. Write a *Lab2.vhd* file that instantiates and interconnects, as shown in Figure 1, one ALU and one 7-Segment Driver, a module which was provided to you. Map the instantiated ALU's *generic* to 4 bits.
    a. Synthesize the design. Review the elaborated schematic to make sure the RTL design synthesized mimics the RTL design shown in Figure 1.
    b. Use the *I/O Planning* tool to design the constraints file. Refer to the *Zybo Reference Manual* and Figures 4 and 5 when assigning the pins.
6. Program the ZYBO with the completed ALU design and verify its functionality using the following tests. Also, in your lab write-up, calculate and show the expected results of the following operations by hand:
    a. Add: 1001 + 1010
    b. Subtract: 1001 – 0010
    c. Two's Complement: 0010 – 0101
    d. AND, OR, XOR: 0101, 1101
    e. NOT: 1010
    f. Shift & Rotate Left: 1011
    g. Shift and Rotate Right: 1101
7. In your lab write-up, answer the following questions:
    a. You have been provided a Two's Complement module for your design. Look over it and describe it's function.

Deliverables

Your Lab 2 project directory, containing all sources, simulation waveforms, bit files, and lab report are to be turned in as a zip file labeled *"ENGIN341_LAB2_LastName_FirstName.zip".*