**Laboratory #5 P3 ALU**

**EE 310 CUPT Fundamentals of Computer Engineering**

College of Engineering and Natural Sciences

Northern Arizona University

Spring 2017

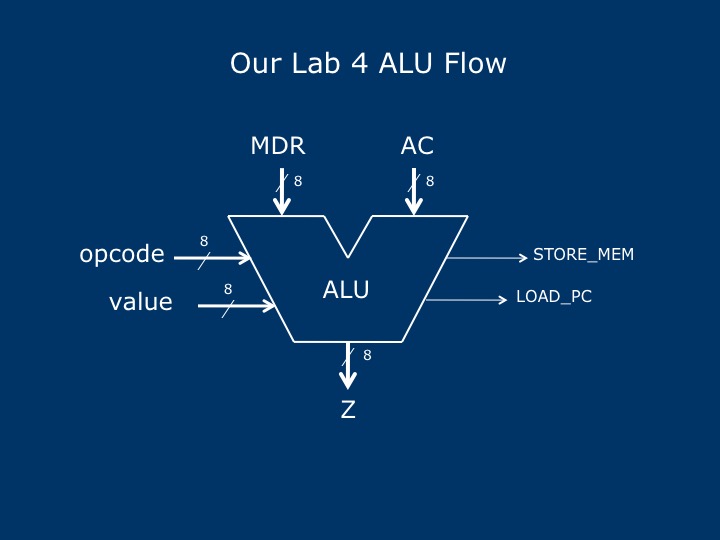
# Objective

At the completion of this lab, the student will be able to complete a VHDL ALU design from a functional specification (write VHDL, use ModelSim to simulate functional behavior, and demonstrate functional behavior on the Cyclone V FPGA board).

# Important Concepts

1. We will introduce the P3 microprocessor in detail in the near future. For now, let’s concentrate on the ALU we will need to prepare. See the block diagram in the figure below. Full details are in the file ‘ALU Tutorial Lab5.ppt’ on BBLearn.
2. In this lab you will develop the ALU component of the P3 processor. There will be two test benches. The first one is a simulation test bench. The second one will be a demonstration test bench since the ALU inputs and outputs will not completely fit on the starter board.
3. You are responsible for keeping your files organized. Make new revisions as necessary.

**ALU Diagram for This Lab**

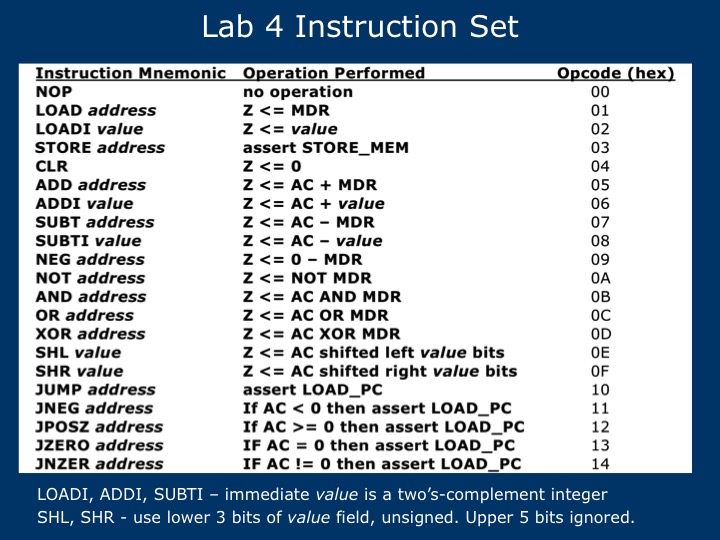


We have no registers in our design yet, so our ALU for this assignment is purely combinational logic. In future assignments, we will modify this ALU slightly as we add registers and other hardware. In the above diagram, inputs are on the left and top, while outputs are on the bottom and right.

The **opcode** and **value** busses will eventually come from the Instruction Register (IR), which we will add later. The data busses **MDR** and **AC** represent the operands coming into the ALU. **MDR** will later come from the Memory Data Register and **AC** will come from the Accumulator. The **Z** output is the data produced by the ALU logic according to the **opcode**, **value**, **MDR**, and **AC** inputs.

For now, we have two single-bit control outputs: **STORE\_MEM** and **LOAD\_PC**. **STORE\_MEM** needs to be activated (asserted) only under the conditions stated in the instruction set. Its purpose will be to control when the processor’s memory (to be added later) must write data from the ALU. The **LOAD\_PC** output must be activated when information needs to be loaded into the Program Counter register (to be added later).

**Instruction Set**



Note: Opcodes not defined in the above instruction set should have benign behavior (that means nothing bad should happen). The best suggestion is to define those unused opcodes the same as the NOP instruction.

# Activity # 1 Initial study

1. (5pts) In the left box below, list the instructions that produce a data result on the Z bus of the ALU. In the right box, list all of the remaining instructions.

|  |  |
| --- | --- |
| Instructions producing results on Z bus:  OpCodes: x01,x02, x04-x0F | Instructions where Z bus does not matter:  X00,x03,x10-x14 |

1. (5pts) There are some instructions that do not require a specific output from the ALU. You are free to choose the value of the ALU output (the Z bus) in these cases. What is it and why did you select it?

We are choosing to use whatever the existing value of Z is. We are choosing this method because it is the most straightforward and won’t require any changes to the Z bus.

1. (5pts) What are the input and output signals of the ALU entity and how many bits are they?

|  |  |
| --- | --- |
| Inputs:  MDR – 8 Bits  AC – 8 Bits  Value -8 Bits | Outputs:  Z – 8 Bits  STORE\_MEM – 1 Bit  LOAD\_PC – 1 Bit |

1. As a class, we will discuss how to choose four good test cases to perform as thorough a test as we can. Once we decide these test cases together, you must determine the correct outputs (**Z**, **STORE\_MEM**, **LOAD\_PC**) for each instruction and each test case.

* Use hexadecimal.
* The test cases should verify the correct operation of each opcode as well as illustrate that connections are correct (correct bus numbering).
* All inputs and all outputs should be ‘0’ and ‘1’ in at least one test case for each instruction.
* For the ADDI instruction, test the carry from least significant bit (LSB) to most significant bit (MSB).
* For the ADDI, SUBTI, and LOADI instructions, test that the *value* input works.
* Some instructions do not need a specific **Z** output value because it is not used. Specify your **Z** outputs for these cases the same way you answered question 2 above.

Note: Four test cases is a ridiculously low number. Many more would be included in a real design.

(15pts) In each box, the format is: **Z bus output** (in hex), **STORE\_MEM**, **LOAD\_PC**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Opcode, Instruction mnemonic** | **Test case #1**  **MDR: 00**  **AC: 55**  **address/value:**  **36** | **Test case #2**  **MDR: 55**  **AC: AA**  **address/value:**  **AB** | **Test case #3**  **MDR: AA**  **AC: 00**  **address/value:**  **49** | **Test case #4**  **MDR: FF**  **AC: 9E**  **address/value:**  **C2** |
| 00 NOP | xx, 0, 0 | xx, 0, 0 | xx, 0, 0 | xx, 0, 0 |
| 01 LOAD | x00, 0, 0 | x55, 0, 0 | xAA, 0, 0 | xFF, 0, 0 |
| 02 LOADI | x36, 0, 0 | xAB, 0, 0 | x49, 0, 0 | xC2, 0, 0 |
| 03 STORE | xx, 1, 0 | xAB, 1, 0 | x49, 1, 0 | xC2, 1, 0 |
| 04 CLR | x00, 0, 0 | x00, 0, 0 | x00, 0, 0 | x00, 0, 0 |
| 05 ADD | x55,0,0 | xFF, 0 0 | xAA, 0, 0 | x9D, 0, 0 |
| 06 ADDI | x8B, 0, 0 | x55, 0, 0 | x49, 0, 0 | x60, 0, 0 |
| 07 SUBT | x55, 0, 0 | x55, 0, 0 | x56, 0, 0 | x9F,0 ,0 |
| 08 SUBTI | x1F, 0, 0 | xFF, 0, 0 | xB7, 0, 0 | xDC, 0, 0 |
| 09 NEG | x00, 0, 0 | xAB, 0, 0 | x56, 0, 0 | x01, 0 , 0 |
| 0A NOT | xFF, 0, 0 | xAA, 0, 0 | x55, 0, 0 | x00, 0, 0 |
| 0B AND | x00, 0, 0 | x00, 0, 0 | x00, 0, 0 | x9E, 0, 0 |
| 0C OR | x55, 0, 0 | x55, 0, 0 | xAA, 0, 0 | xFF, 0, 0 |
| 0D XOR | x55, 0, 0 | xFF, 0, 0 | xAA, 0, 0 | x61, 0, 0 |
| 0E SHL | xA8, 0, 0 | x50, 0, 0 | x00, 0, 0 | x78, 0, 0 |
| 0F SHR | x01, 0, 0 | x15, 0, 0 | x00, 0, 0 | x27, 0, 0 |
| 10 JUMP | xx, 0, 1 | xx, 0, 1 | xx, 0, 1 | xx, 0, 1 |
| 11 JNEG | xx, 0, 0 | xx, 0, 1 | xx, 0, 0 | xx, 0, 1 |
| 12 JPOSZ | xx, 0, 1 | xx, 0, 0 | xx, 0, 1 | xx, 0, 0 |
| 13 JZERO | xx, 0, 0 | xx, 0, 0 | xx, 0, 1 | xx, 0, 0 |
| 14 JNZER | xx, 0 , 1 | xx, 0 , 1 | xx, 0 , 0 | xx, 0 , 1 |

**Instructor’s checklist:**

• Correct outputs for all instructions

# Activity #2 VHDL Code and Simulations

Using the results from Activity #1, create the entity and architecture of the P3 ALU. Below is an outline of the VHDL code.

Create a new project **lab4** with revision **alu\_sim\_tb**. Create a new VHDL file by selecting **File > New...** then clicking on **VHDL File**, then **OK**. Enter the following text. Be careful of punctuation. Use generic parameters to indicate the size of busses, using the default value for this architecture.

-- alu.vhd

-- Implements the instruction set

-- for the uP3 microprocessor in Lab 4

--

-- Author: Your name (pinyin) here, NAU/CUPT EE

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.numeric\_std.all;

library altera\_mf;

use altera\_mf.altera\_mf\_components.all;

entity alu is

generic (

-- **put generic parameter list here**

);

port (

-- **put port list here, use type SIGNED for the data busses**

);

end alu;

architecture behav of alu is

begin

process() -- include necessary signals in sensitivity list

begin

-- **put your code here!**

end process;

end behav;

Create a symbol for this VHDL design. Use **File > Create / Update > Create Symbol Files for Current File**. Insert the symbol in the **alu\_sim\_tb.bdf** file.

Place input and output pins in the bdf file and connect them to the symbol, using names of your choosing. Since this is only for simulation (and not for programming the starter board), you can choose names that indicate the function. Use hexadecimal outputs as much as possible. You can make these “vector” pins using the syntax *signal\_name*[7..0]. This is equivalent to the VHDL vector range 7 downto 0.

Make the **alu\_sim\_tb.bdf** entity the top-level entity. If you don’t make it the top-level entity, it won’t compile. Forgetting this can lead to frustration and lost time. Include all other necessary files and no others. Compile the design.

Create a force file to test the design. Use the four test cases from Activity #1 above and simulate them for each instruction. Do a functional simulation and verify that the results match the expected results from Activity #1. Display all of the input and output vector signals in hexadecimal in the wave window.

Here are some additional force file hints:

* The force file can have a .txt extension so that it will open in Notepad automatically.
* You can specify the time that a signal takes on a value relative to the current simulation time. If the current simulation time is 200ns, these commands will set reset to 0 at 200ns and then a pulse of width 50ns beginning at 1400ns.

force reset 0

force reset 1 1200ns

force reset 0 1250ns

* You can specify values in hex.

force in\_MDF x"FF" 1200ns

**Instructor’s checklist:**

• alu\_tb\_sim.bdf

• alu VHDL code

• Demonstrate functional simulation to instructor

# Activity #3 Demonstration

You must have the simulation completed before the demo. The starter board will not accommodate all the inputs and outputs of the ALU so you will create another test bench for the demo. Use the test bench **alu\_demo\_tb.bdf**. You will need to edit this file. Note how you can label the names of busses (right click on the wire and edit Properties). Use the **seg7** binary to hexadecimal decoder from Lab #3 to display the important information for this demonstration. Program the FPGA target and test all instructions with all test cases.

**Instructor’s checklist:**

• Simulations have been completed.

• Correct operation on the FPGA board.

**Files that need included in lab report:**

* Questions and answers in Activity #1
* VHDL code for ALU
* Block diagram file for simulation and demonstration, respectively
* ModelSim do file for simulation
* Functional simulation results (must be clear and easy to read)

# Activity #4 Instructor questions (in-class quiz) --- 30pts

To show that each of you are actively involved in the projects, a quiz will be given at the beginning of lab6. Questions will be closely related to this lab such as the principles of ALU and VHDL programming. Quiz time would be 20min and the grade will be part of your lab grade!