ECPE 174 – Lab 6
ALU

1. **Objectives**
   Design an arithmetic logic unit. Implement and test the data sub-modules. Implement the control unit and test the system.

2. **Problem Statement**
   We want to build an ALU (arithmetic logic unit), which is the heart of any processor. Over the two weeks, we will design both the datapath and control units.

   The ALU will operate on 3-bit numbers (A and B). It will:
   - Add
   - Subtract
   - Compare 2 values to determine if A equals B, is strictly greater than B, or is strictly less than B
   - Check if A==0

   The ALU must be a hierarchical design, utilizing correct digital design principles. Each component needs to be individually tested and verified. At a minimum, you should have a 3-bit adder, a 3-bit logic unit, and a FSM-based control unit. In implementing your system, you can use any technique you want:
   a. Write your own VHDL module.
   b. Use example code in the book.
   c. Use the Megawizard plug-in manager inside Quartus.

   The ALU will use the following instruction set:

```
Add = 000
Subtract = 001
Equal = 100
Greater than = 101
Less than = 110
A equal 0 = 111
```

   We will enter instructions on toggle switches SW0-SW2, 'A' values on switches SW3-SW5, and 'B' values on switches SW6-SW8. KEY3 will indicate the instruction and data are ready. Output the instruction on the red LEDs. Output 'A' values, 'B' values, and the result on the 7-segment displays. For the logic operations, you should display '1' if the case is true and '0' if false.

3. **Pre-Lab**
   Each week you need to turn in a pre-lab. For the first week, complete the following steps before lab and turn in by 2pm:
   1. Design a block diagram of your ALU. Identify which modules are part of the datapath and which are part of the control.
   2. Outline your plan for design, implementation, and test of the ALU. Explicitly detail what you will do in the first lab and what you will do in the second lab. This should be several paragraphs of text along with bulleted lists of procedures.
   3. For the code you complete before the first lab, explain your design choices, provide the VHDL, and include simulation results demonstrating working VHDL.

Submit electronically via Sakai.

By the second week, your design must be finished. Turn in by 2pm:

1. All VHDL code for your ALU design.
2. An explanation of your design choices. What type of FSM did you use and why?
3. Simulation results verifying component and full system operation (anything not tested by the beginning of the first lab day).

Submit electronically via Sakai.

4. In-Lab

Complete the following steps in lab:

1. Implement the adder/subtractor VHDL design on the Cyclone II. Demonstrate for check-off.
2. Implement the logic VHDL design on the Cyclone II. Demonstrate for check-off.
3. Implement the complete design on the Cyclone II. For the 7-segment display, include seven_seg_hex.vhd as a component in your design (located on Sakai in the Resources section). Demonstrate for check-off.
4. Modify your design to include memory. Add control logic to include a load state where you can load up to 4 instructions into memory and then switch to an execute state where you can step through each instruction (with the number depending on how many you input). It should wrap around such that if you load 2 instructions but try to execute 3, it will return to the first instruction. Remember to keep your design hierarchical. Demonstrate for check-off.
5. Draw a system diagram including the main components. Using the provided constraints file and the Timing Analyzer, analyze the timing of your overall system. Determine the longest path, the propagation delay through the add/subtract unit, the propagation delay through the logic unit, and the maximum clock frequency. Outline the equation for the maximum clock frequency using the information from the analysis. Demonstrate for check-off.

5. Post-Lab

Discuss the following your lab report:

- How did your test procedure work? What if any modifications were necessary to either the design or the test procedure?
- How much space do the units require? What is the clock information for your different units? Include setup time, hold time, and maximum clock rate.
- Compare and contrast the space and time requirements between the add/subtract and logic units. Which requires the most space? Which has the longest setup and hold times? Do these answers make sense to you? Explain.
- What, if any, assumptions did you need to make in order to complete this design? What, if any, additional logic or functional extensions would improve the system?
- Outline the information determined from the timing analyzer. Describe your computation of the maximum clock frequency and longest propagation delay path. Are these reasonable values? How might you modify the design to increase the clock frequency? How might you modify it to reduce the propagation delay?