

# Lab 1: ALU Lab

EE312 Computer Architecture

Professor: Minsoo Rhu

TA (Lab 1): Ranggi Hwang, Head TA: Youngeun Kwon

(EMAIL: kaist.ee312.ta@gmail.com)

## 1. Overview

*Lab 1* is a toy project that teaches you the basics of Verilog language. You are going to learn how to write a Verilog code and how to simulate it. In *Lab 1*, you are required to implement an Arithmetic Logic Unit (ALU) by using Verilog language. You are able to complete *Lab 1* based on what you have learned from the Verilog tutorial. Have fun!

## 2. Files

You are given two files:

1. *alu.v*: This is where you implement the ALU module.
2. *ALU\_TB.v*: A Testbench file which you can test and grade your ALU module with

You only need to change *alu.v*. You don't need to change *ALU\_TB.v*.

## 3. ALU Lab

In *Lab 1*, you are required to make an ALU that has the following specifications:

1. Inputs and outputs are 16-bit signed binary numbers.
2. Operations are 4-bit binary numbers.
3. Overflow must be detected.

For Add & Sub operation, the ALU should be able to handle overflow. *Cout* must be one if overflow happens; otherwise, *Cout* should be zero.

The detailed explanations of the operations that your ALU module is required to handle are shown in Table 1.

OP	operation	description
0000	A + B	16-bit <i>addition</i>
0001	A – B	16-bit <i>subtraction</i>
0010	A and B	16-bit <i>and</i>
0011	A or B	16-bit <i>or</i>
0100	A nand B	16-bit <i>nand</i>
0101	A nor B	16-bit <i>nor</i>
0110	A xor B	16-bit <i>xor</i>
0111	A xnor B	16-bit <i>xnor</i>
1000	A	<i>Identity</i>
1001	~A	16-bit <i>not</i>
1010	A >> 1	<i>Logical right shift</i>
1011	A >>> 1	<i>Arithmetic right shift</i>
1100	A[0]A[15:1]	<i>Rotate right</i>
1101	A << 1	<i>Logical left shift</i>
1110	A <<< 1	<i>Arithmetic left shift</i>
1111	A[14:0]A[15]	<i>Rotate left</i>

**Table 1. Operations**

#### 4. Grading

The TAs will grade your lab assignments with *ALU\_TB.v*, which is already given to you. If you pass all the tests in *ALU\_TB.v*, you will get a full score. Your score is determined by how many tests you pass.

#### 5. Lab Report Guidance

You are required to submit a lab report for every lab assignment. You can write your report either in Korean or English.

Your lab report **MUST** include the following sections:

1. Introduction
  - a. *Introduction* includes what you think you are required to accomplish from the lab assignment and a brief description of your design and implementation.
2. Design
  - a. *Design* includes a high-level description of your design of the Verilog modules (e.g., the relationship between the modules).
  - b. Figures are very helpful for the TAs to understand your Verilog code.
  - c. The TAs recommend that you include figures because drawing them helps you how to *design* your modules.
3. Implementation
  - a. *Implementation* includes a detail description of your implementation of what you design.
  - b. Just writing the overall structure and meaningful information is enough; you don't have to explain minor issues that you solve in detail.
  - c. **Do not copy and paste your source code.**
4. Evaluation
  - a. *Evaluation* includes how you evaluate your design and implementation, and simulation results.

**b. *Evaluation* must include how many tests you pass in *ALU\_TB.v*.**

5. Discussion

- a. *Discussion* includes any problems that you experience when you follow through the lab assignment or any feedbacks for the TAs.
- b. Your feedbacks are very helpful for the TAs to further improve *EE312* course!

6. Conclusion

- a. *Conclusion* includes any concluding remarks of your work or what you accomplish through the lab assignment.

## 6. Requirements

You **MUST** comply with the following rules:

- You should implement the lab assignment in **Verilog**.
- You should only implement the **TODO** parts of the given template.
- You should name your lab report as **Lab1\_YourName1\_StudentID1\_YourName2\_StudentID2.pdf**.
- You should compress the lab report and source code, then name the compressed zip file as **Lab1\_YourName1\_StudentID1\_YourName2\_StudentID2.zip**, and submit the zip file on the KLMS.