Computer Organization

Lab 1: 32-bit ALU

Due: 2023/4/2

1. Goal

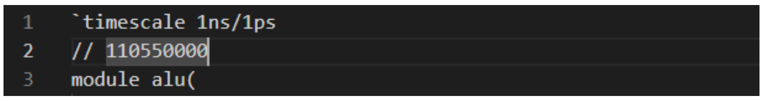
The ALU (Arithmetic Logic Unit) is the basic computing component of a CPU. Its operations include AND, OR, addition, subtraction, etc. The goal of this lab is to implement a 32-bit ALU. This lab will help you understand the CPU architecture.

(Note that future labs will be highly related to previous ones, so please try your best to complete every lab.)

2. Homework Requirement

a. Please use Xilinx Vivado as your HDL simulator.

b. Please **attach your student ID as comments** at the top of each file.
Ex.



```
`timescale 1ns/1ps
// 110550000
module alu(
```

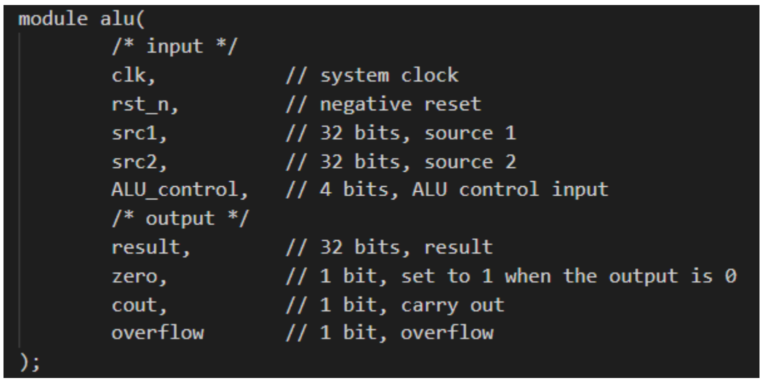
c. Please zip the archives and **name it as "Lab1_ID.zip"** (ex. Lab1_110550000.zip) before uploading it to E3.

d. testbench.v is provided. You can use it to verify the correctness of your design.

e. Any work by fraud will absolutely get a zero point.

f. The names of the top module and the IO ports must be named as follows:

Top module: alu.v



```
module alu(
        /* input */
        clk,            // system clock
        rst_n,          // negative reset
        src1,           // 32 bits, source 1
        src2,           // 32 bits, source 2
        ALU_control,    // 4 bits, ALU control input
        /* output */
        result,         // 32 bits, result
        zero,           // 1 bit, set to 1 when the output is 0
        cout,           // 1 bit, carry out
        overflow        // 1 bit, overflow
);
```

g. In order to have a good coding style, please obey the following rules:

- One module in one file.
- The module name and the file name must be the same.

For example: The file "alu.v" contains the module "alu" only.

h. ALU starts to work when the signal *rst_n* is 1, and then catches the data from *src1* and *src2*.

i. Instruction set: basic operation instruction (60%)

ALU Action	Name	ALU Control Input
And	And	0000
Or	Or	0001
Add	Addition	0010
Sub	Subtraction	0110
Nor	Nor	1100
Slt	Set less than	0111

(Hint: ALU Control Input = {A_invert, B_invert, Operation})

j. zcv three control signal: zero, carry out, overflow (30%)

1. **"zero"**: must be set when the result is 0.
2. **"cout"**: must be set when there is a carry out.
3. **"overflow"**: must be set when overflow.

(Note: You only have to handle carry out and overflow flags in "add" and "sub" operations.)

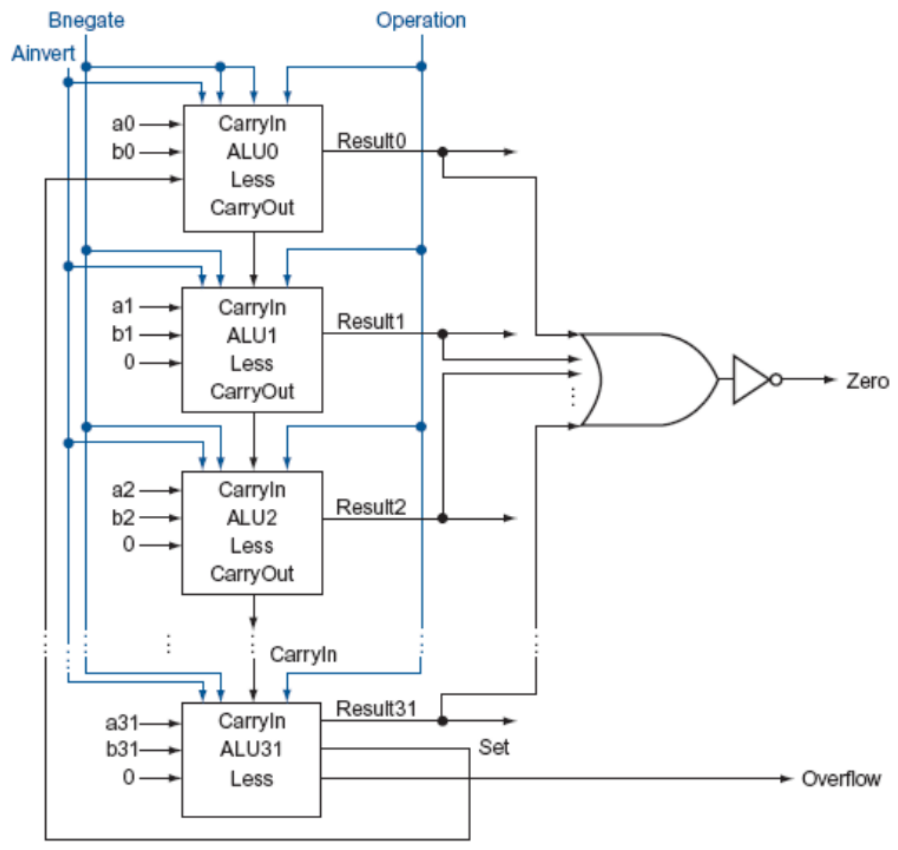
k. About alu_top.v

alu_top.v is an 1-bit ALU module. We will not force you to use alu_top.v to implement this homework. However, we still strongly recommend you to implement this homework by using alu_top.v as submodule, as in the architecture diagram below. The designs in future labs will become more complex, so implementing your design by connecting multiple modules together is an essential skill.

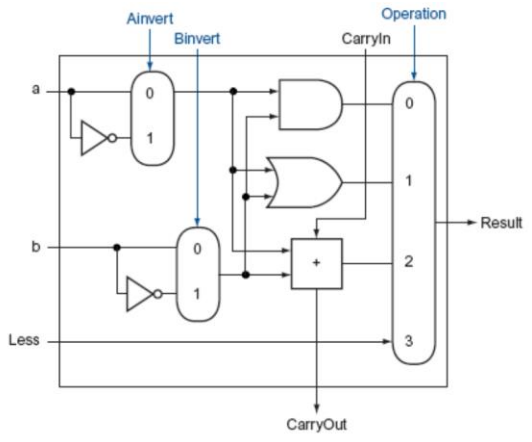
(Note: You can add any module into your design if you want, but the top module must be alu in the alu.v as defined in Section 2-f.)

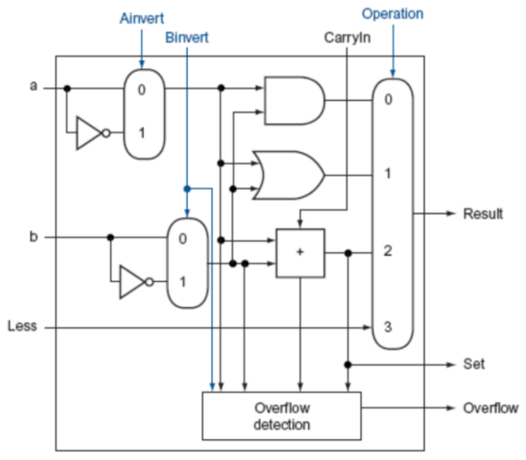
3. Architecture Diagram

- 32-bit ALU



- 1-bit ALU (ALU00~ALU30)
- 1-bit ALU (ALU31)





(Hint: reference for ‘overflow detection’, https://blog.csdn.net/qq_43121830/article/details/110305726)

4. Grade

a. **Total: 100 points (plagiarism will get 0 point)**

b. **Report: 10 points** (name it as "Lab1_ID.pdf") (ex. Lab1_110550000.pdf)

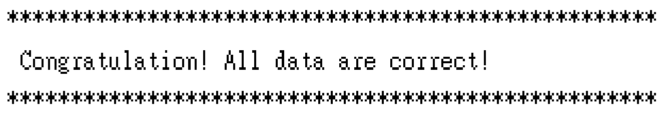
c. Late submission: Score * 0.8 before 4/9. After 4/9, you will get 0.

d. Please zip the following files and upload:
   1. alu.v
   2. alu_top.v
   3. Lab1_ID.pdf

e. Verify your design

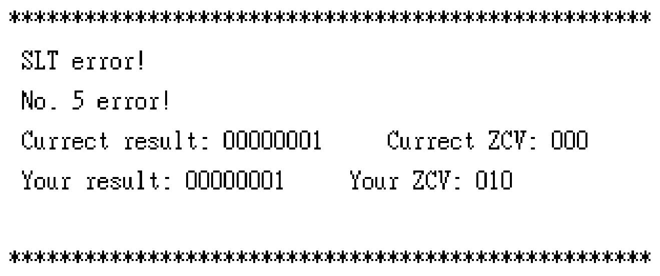
   Please add all the.txt files into the project (refer to Section 5), after simulation finishes, you will get the simulation result.

   - Pass all patterns



```
**************************************************
 Congratulation! All data are correct!
**************************************************
```

   - Fail



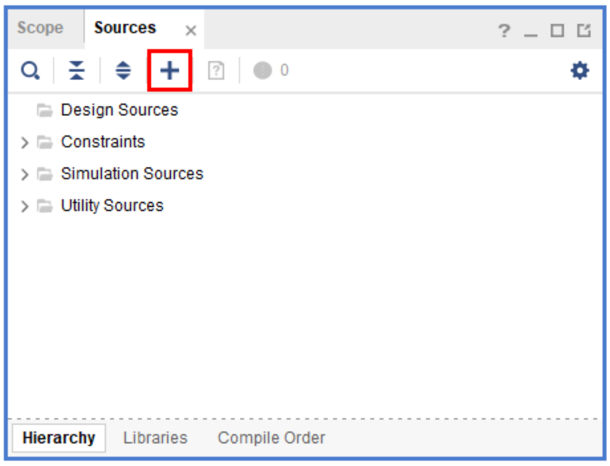
```
**************************************************
 SLT error!
 No. 5 error!
 Currect result: 00000001     Currect ZCV: 000
 Your result: 00000001     Your ZCV: 010

**************************************************
```

   Beside given testcases, several hidden testcases will also be adopted to evaluate your design.

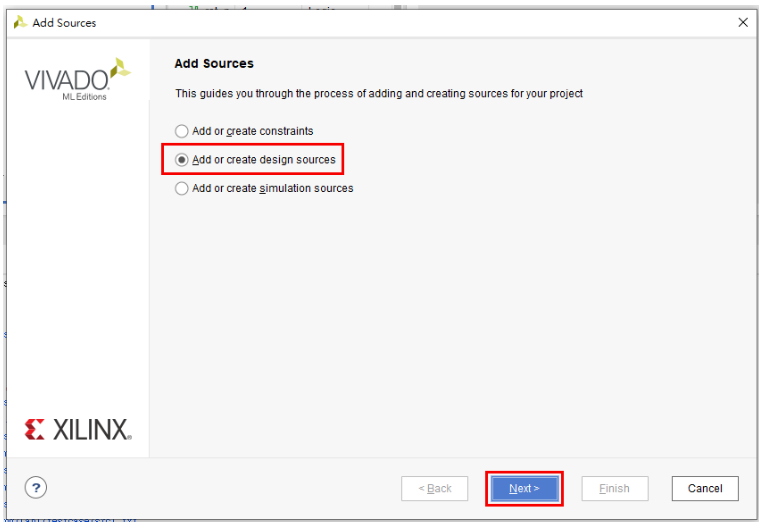
5. Add Design and Simulation Source

Before running the simulation, you need to add testcases to your project.

1. Add sources.



2. Add design sources.

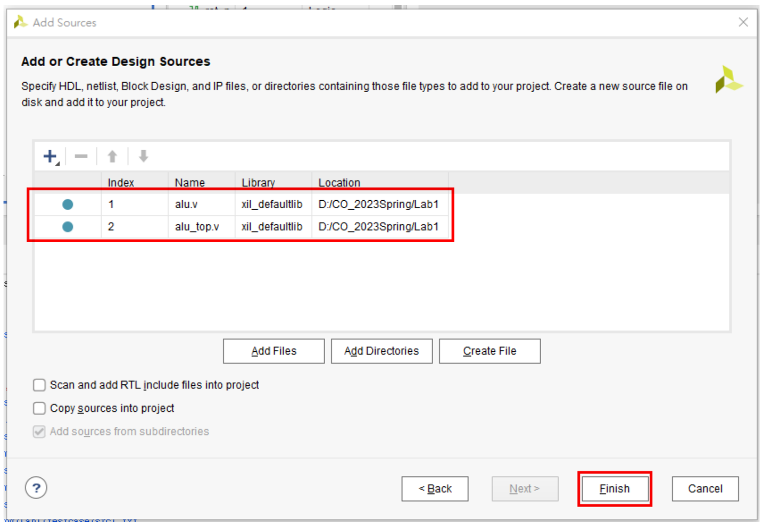

Add Sources

Add Sources

This guides you through the process of adding and creating sources for your project

- Add or create constraints
- [✓ Add or create design sources]
- Add or create simulation sources

3. Add files.


Add Sources

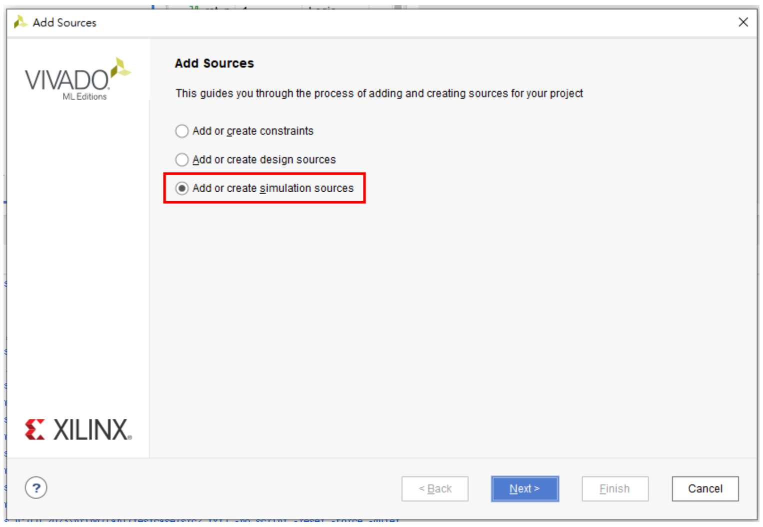
Add or Create Design Sources

Specify HDL, netlist, Block Design, and IP files, or directories containing those file types to add to your project. Create a new source file on disk and add it to your project.

	Index	Name	Library	Location
●	1	alu.v	xil_defaultlib	D:/CO_2023Spring/Lab1
●	2	alu_top.v	xil_defaultlib	D:/CO_2023Spring/Lab1

- ☐ Scan and add RTL include files into project
- ☐ Copy sources into project
- ☑ Add sources from subdirectories

4. Add simulation sources.



Add Sources

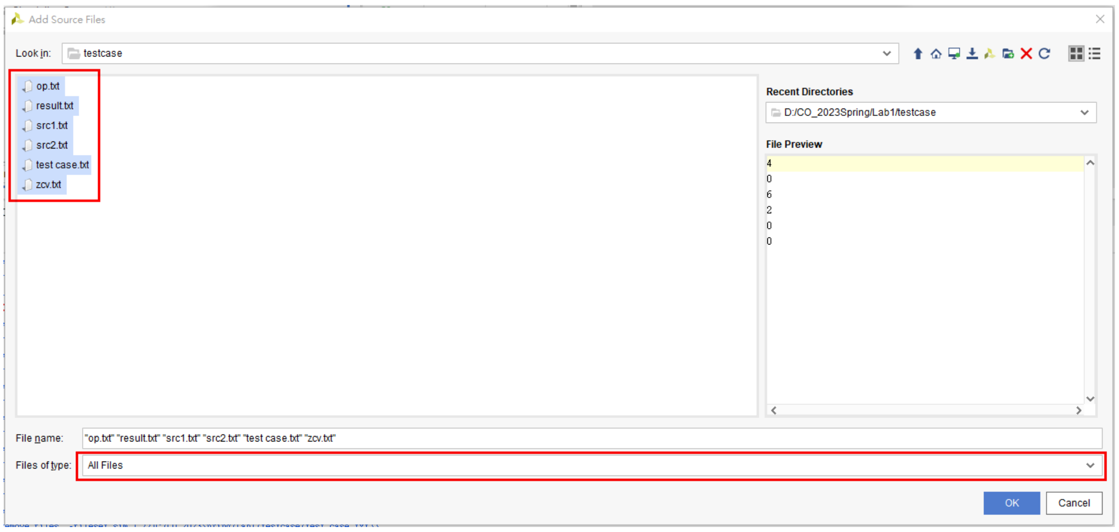
Add Sources

This guides you through the process of adding and creating sources for your project

- ○ Add or create constraints
- ○ Add or create design sources
- [✓ Add or create simulation sources]

< Back | Next > | Finish | Cancel

5. Add testbench.v and all files in the /testcase directory.



Add Source Files

Look in: testcase

- op.txt
- result.txt
- src1.txt
- src2.txt
- test case.txt
- zcv.txt

Recent Directories: D:/CO_2023Spring/Lab1/testcase

File Preview

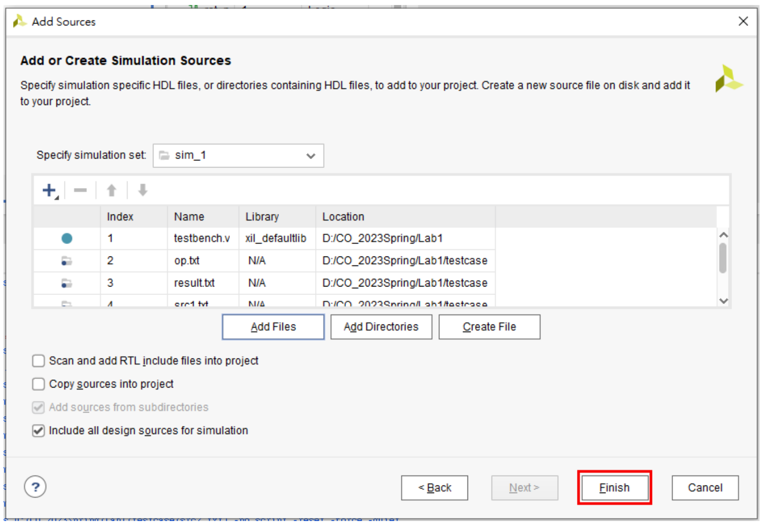
```
4
0
6
2
0
0
```

File name: "op.txt" "result.txt" "src1.txt" "src2.txt" "test case.txt" "zcv.txt"

Files of type: All Files

OK | Cancel

6. Finish and start the simulation.



Add Sources

Add or Create Simulation Sources

Specify simulation specific HDL files, or directories containing HDL files, to add to your project. Create a new source file on disk and add it to your project.

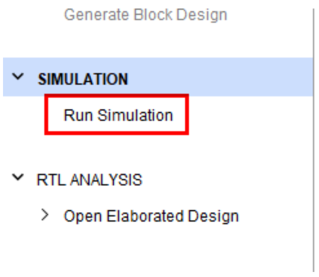
Specify simulation set: sim_1

	Index	Name	Library	Location
●	1	testbench.v	xil_defaultlib	D:/CO_2023Spring/Lab1
	2	op.txt	N/A	D:/CO_2023Spring/Lab1/testcase
	3	result.txt	N/A	D:/CO_2023Spring/Lab1/testcase
	4	src1.txt	N/A	D:/CO_2023Spring/Lab1/testcase

Add Files | Add Directories | Create File

- ☐ Scan and add RTL include files into project
- ☐ Copy sources into project
- ☑ Add sources from subdirectories
- ☑ Include all design sources for simulation

< Back | Next > | Finish | Cancel



Generate Block Design
SIMULATION
Run Simulation
RTL ANALYSIS
Open Elaborated Design