Computer Organization
Lab 2 - 32-bit ALU

教授:蔡文錦

TAs:單宇晟 吳年茵 曾偉杰

Objectives

In Lab2, you are going to implement an **32-bit ALU (Arithmetic Logic Unit)** by Verilog.

Through this lab, you will learn how to design the basic function unit and get quick review of Verilog.

Note that you should design these circuits in gate level as **sequential logic** instead of combinational logic.

※ The ALU designed in this lab may also be used in the succeeding Labs.

Lab 2 Description

- Tools
- Attached Files
- Arithmetic Logic Unit (ALU)
  - Overview
  - 1-bit ALU Architecture Diagram
  - 32-bit ALU Architecture Diagram
  - ALU Control Signals Table
- Grading Policy
- Submission

Tools

- **Icarus verilog (to compile and simulate Verilog code)**
  - Mac OSX
    - brew install icarus-verilog
  - Windows
    - https://bleyer.org/icarus/iverilog-v11-20210204-x64_setup.exe
  - Linux
    - sudo apt install verilog
- **GTKWave (unnecessary)**
  - Easy to debug

Attached Files

- **You need to do:**
  - MUX_2to1.v
  - MUX_4to1.v
  - ALU_1bit.v
  - ALU.v

```
module ALU_1bit(
    input               src1,       //1 bit source 1  (input)
    input               src2,       //1 bit source 2  (input)
    input               less,       //1 bit less      (input)
    input               Ainvert,    //1 bit A_invert  (input)
    input               Binvert,    //1 bit B_invert  (input)
    input               cin,        //1 bit carry in  (input)
    input       [2-1:0] operation,  //2 bit operation (input)
    output reg          result,     //1 bit result    (output)
    output reg          cout        //1 bit carry out (output)
    );

/* Write down your code HERE */

endmodule
```

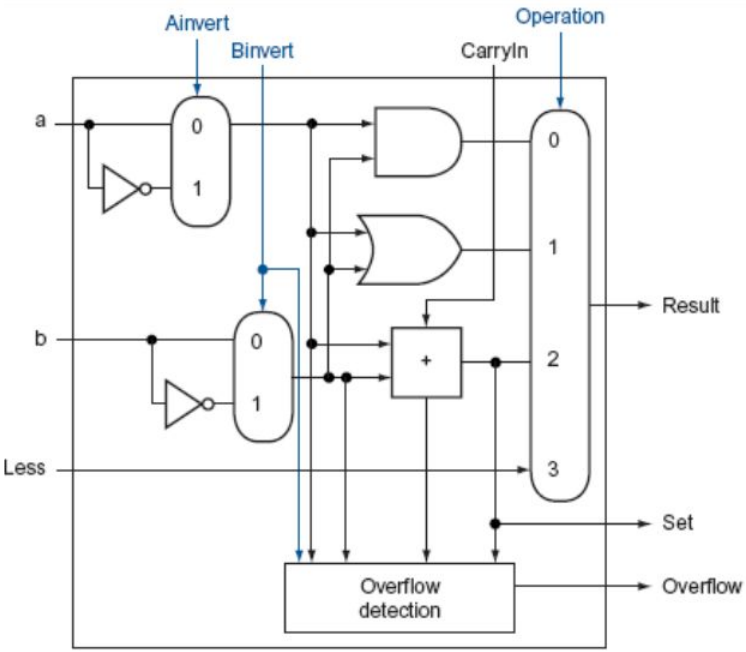
- **File to validate the correctness of your implementation:**
  - testbench.v (for 32-bit ALU)
- **Testcase:**
  - *.txt

DO NOT modify these files

※ **Basically, you don't need to add any additional .v file.** We'll test your code using our own testbench.

Arithmetic Logic Unit (ALU) Overview

The block diagram of the ALU is shown in the following figure

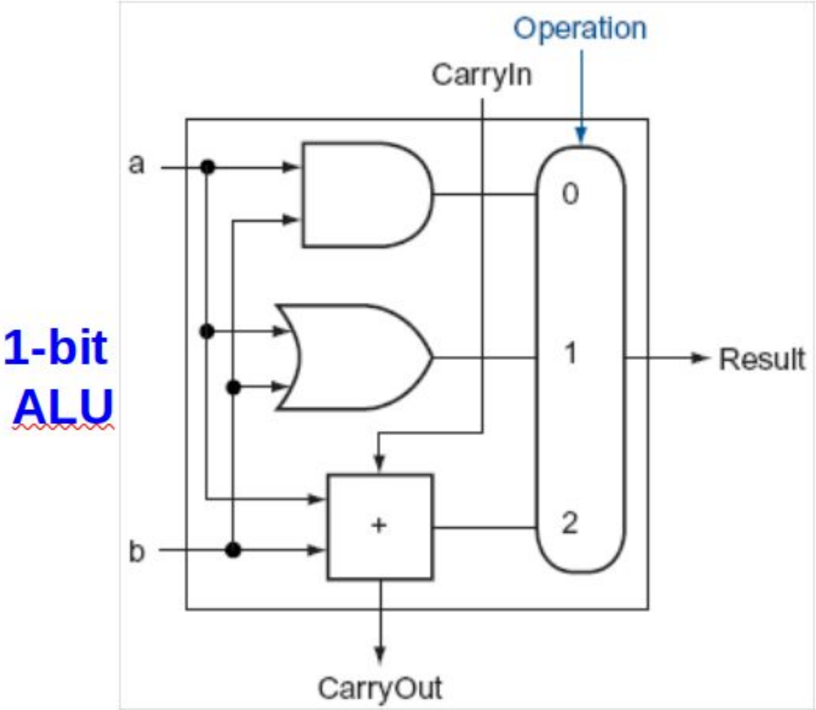


You should generate three outputs of the ALU:

- **result**: the computation result
- **zero**: a 1-bit output control signal.
  - set to 1 when the result is 0.
  - set to 0 otherwise.
- **overflow**: a 1-bit output control signal.
  - set to 1 when the result overflows (add, sub).
  - set to 0 otherwise.
- **set**: a 1-bit output control signal.
  - set to 1 if a < b
  - set to 0 otherwise

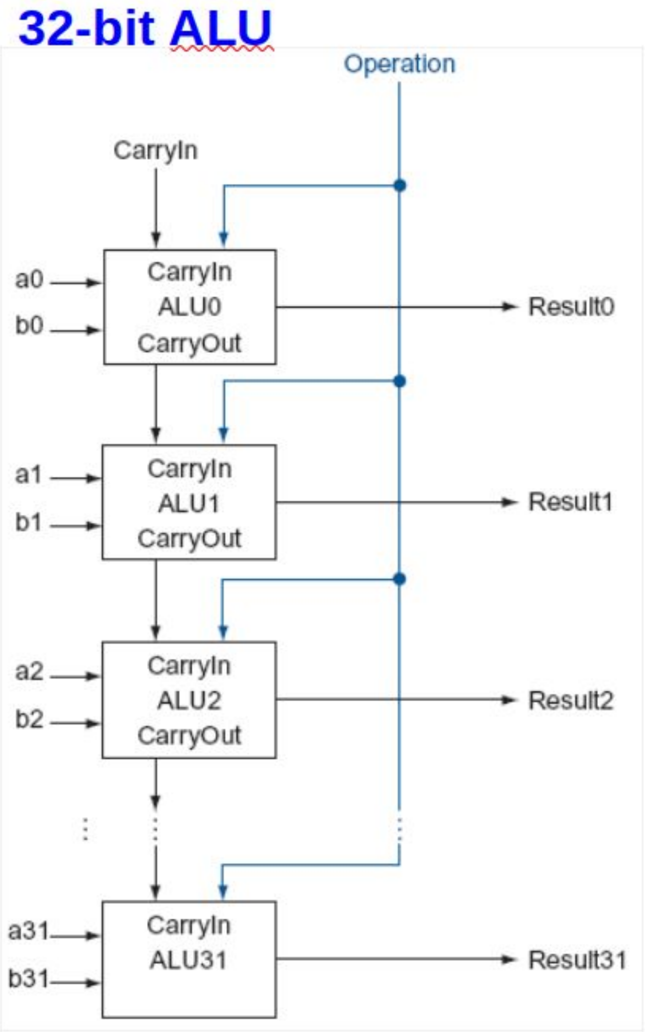
1-bit ALU

The block diagram of the 1-bit ALU for this lab is shown in the right figure.



32-bit ALU

The block diagram of the 32-bit ALU for this lab is shown in the right figure.



ALU Control Signals Table

The operations and corresponding control signals for the ALU included in Lab2 are described in the following table.

Note that all operations in the table are **32-bit** operations.

Therefore, you should implement a typical 1-bit ALU first and then build the 32-bit ALU using 32 1-bit ALUs.

Function	ALU control
AND	0000
OR	0001
add	0010
sub	0110
slt	0111
NOR	1100
NAND	1101

Compile & Run

- **Compile**
  - $ iverilog -o lab2 testbench.v ALU.v

- **Run**
  - $ ./lab2

Wrong results:

```
VCD info: dumpfile alu.vcd opened for output.
**********************************************************
*               PATTERN RESULT TABLE                     *
**********************************************************
* PATTERN *               Result                 * ZCV *
**********************************************************
* No. 1 error!                                           *
* Correct result: eeeeeeee      Correct ZCV: 000         *
* Your result: xxxxxxxx         Your ZCV: xxx            *
**********************************************************
* No. 2 error!                                           *
* Correct result: 00000000      Correct ZCV: 100         *
* Your result: xxxxxxxx         Your ZCV: xxx            *
**********************************************************
* No. 3 error!                                           *
* Correct result: 9bf5fea6      Correct ZCV: 000         *
* Your result: xxxxxxxx         Your ZCV: xxx            *
**********************************************************
```

All testcase PASS:

```
$ ./lab2
VCD info: dumpfile alu.vcd opened for output.
**********************************************************
*               PATTERN RESULT TABLE                     *
**********************************************************
* PATTERN *               Result                 * ZCV *
**********************************************************
*      Congratulation! All data are correct!             *
**********************************************************
Correct Count: 30
testbench.v:91: $finish called at 415000 (1ps)
```

Grading Policy

- There are **10 hidden cases**, and you will get 10 points for each correct testcase, totally 100 points.
- **Any assignment work by fraud will get a zero point !**
- **No late submission !**

Submission

- **Please attach student IDs as comments at the top of each file.**
- The files you should hand in include:
  - **MUX_2to1.v**
  - **MUX_4to1.v**
  - **ALU_1bit.v**
  - **ALU.v**
- Compress the above file into one zip file, and name your zip file as **HW2_{studentID}.zip (e.g. HW2_123456789.zip)**
  - Make sure not to add an extra folder layer.
- **Deadline: 2025/04/09 23:55**
- **Wrong format will have 20% penalty !**