Computer Organization, Spring 2017

Lab 1: 32-bit ALU

Due: 2017/3/23

1. Goal

The goal of this LAB is to implement a 32-bit ALU (Arithmetic Logic Unit). ALU is the basic computing component of a CPU. Its operations include AND, OR, addition, subtraction, etc. This series of LABs will help you understand the CPU architecture. LAB 1 will be reused; you will use this module in later LABs.

2. HW Requirement

(1) Please use Xilinx ISE as your HDL simulator.

(2) Please attach your names and student IDs as comment at the top of each file.

(3) Please use the Testbench we provide you.

(4) The names of top module and IO ports must be named as follows:

Top module: alu.v

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

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

In order to have a good coding style, please obey the rules below:

One module in one file.

Module name and file name must be the same.

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

(5) Basic instruction set (60%)

ALU action	Name	ALU control input
And	And	0000
Or	Or	0001
Add	Addition	0010
Sub	Subtract	0110
Nor	Nor	1100
Nand	Nand	1101
Slt	Set less than	0111

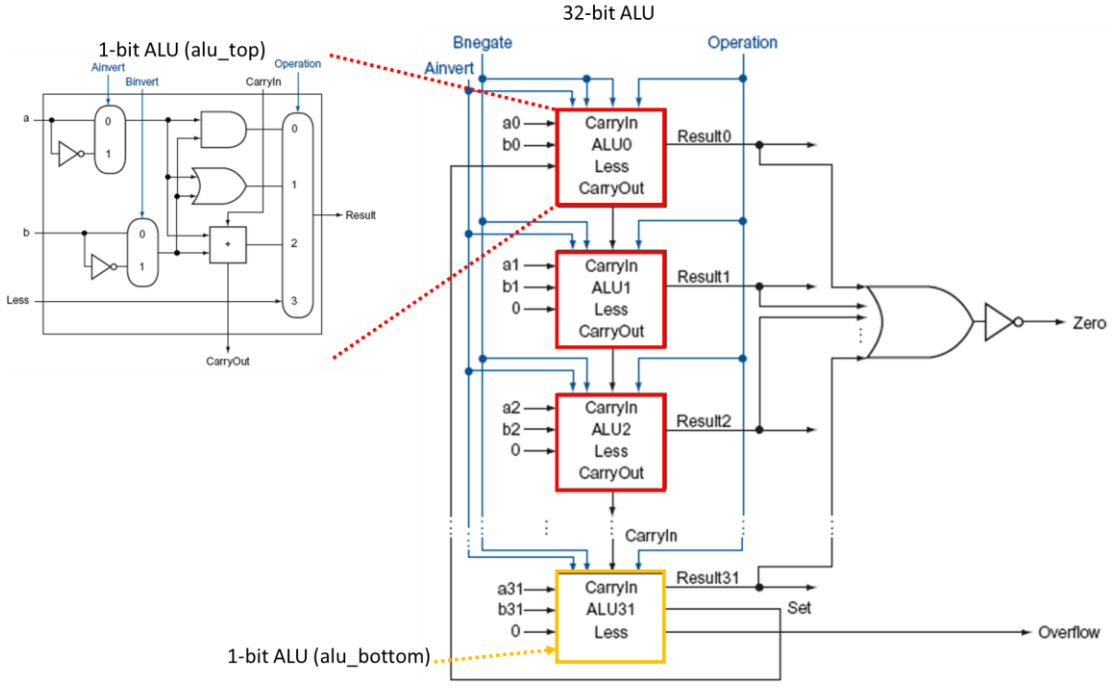
(6) ZCV three flags: zero, carry out, and overflow (30%)

zero: must be set when the output is 0

cout: must be set when carry out

overflow: must be set when overflow

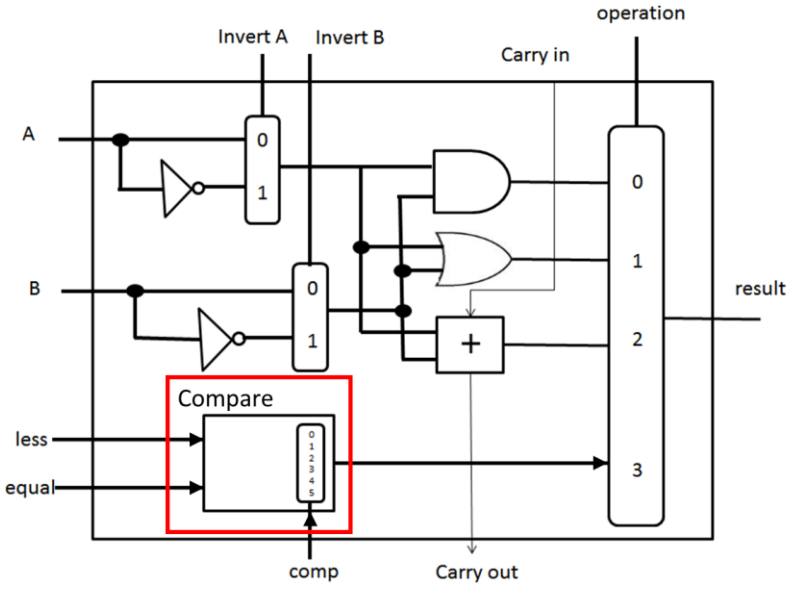
3. Architecture diagram



4. Bonus: Extra instruction set (10%)

ALU action	Name	ALU control input
Slt	Set less than	0111_000
Sgt	Set great than	0111_001
Sle	Set less equal	0111_010
Sge	Set great equal	0111_011
Seq	Set equal	0111_110
Sne	Set not equal	0111_100

Hint: Add a module named Compare in 1-bit ALU!



5. Grade

(1) Total: 110 points (plagiarism will get 0 point)

(2) Document: 10 points

(3) Late submission: 10 points off per day

6. Hand in

Please follow the following rule! Zip your folder and name it as "ID1_ID2.zip" (e.g., 0416001_0416002.zip) before uploading to e3. Multiple submissions are accepted, and the version with the latest time stamp will be graded.

7. How to test

The function of testbench is to read input data automatically and output erroneous data. Please put all the .txt files and project in the same folder, after simulation finishes, you will get some information.

Partial error:

```
# vsim -voptargs=+acc work.testbench
# Loading work.testbench
# Loading work.alu
# Loading work.bit_alu_top
# Loading work.bit_alu_bottom
VSIM 110> run -all
# ************************************************
#  No.12 error!
#  Currect result: 01111100     Currect ZCV: 000
#  Your result: 7fffffff     Your ZCV: 000
#
# ************************************************
```

Or all cases pass:

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

8. Q&A

For any questions regarding Lab 1, please contact

林恩德 <sder505022@gmail.com>,

黃睿騏 <ricky798353@gmail.com>,

陳怡霖 <ss98350131@gmail.com>

9. Requirement

Please write Verilog code according to the structure diagrams illustrated in the description of LAB1, i.e., writing "result=A+B;" or "if(A<B) result=1;" are not allowed when you implement 32-bit ALU; otherwise, you will get some points taken off!

Besides, this is the most basic LAB, the later LABs will have more modules and wire connections, so please spend some time reviewing how to write Verilog code.