Computer Organization, Spring 2018

Lab 1: 32-bit ALU

Due: 2018/4/16

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
Addu	(unsigned) Addition	0010
Subu	(unsigned) Subtract	0110
Nor	Nor	1100
Nand	Nand	1101
Sltu	(unsigned) Set less than	0111

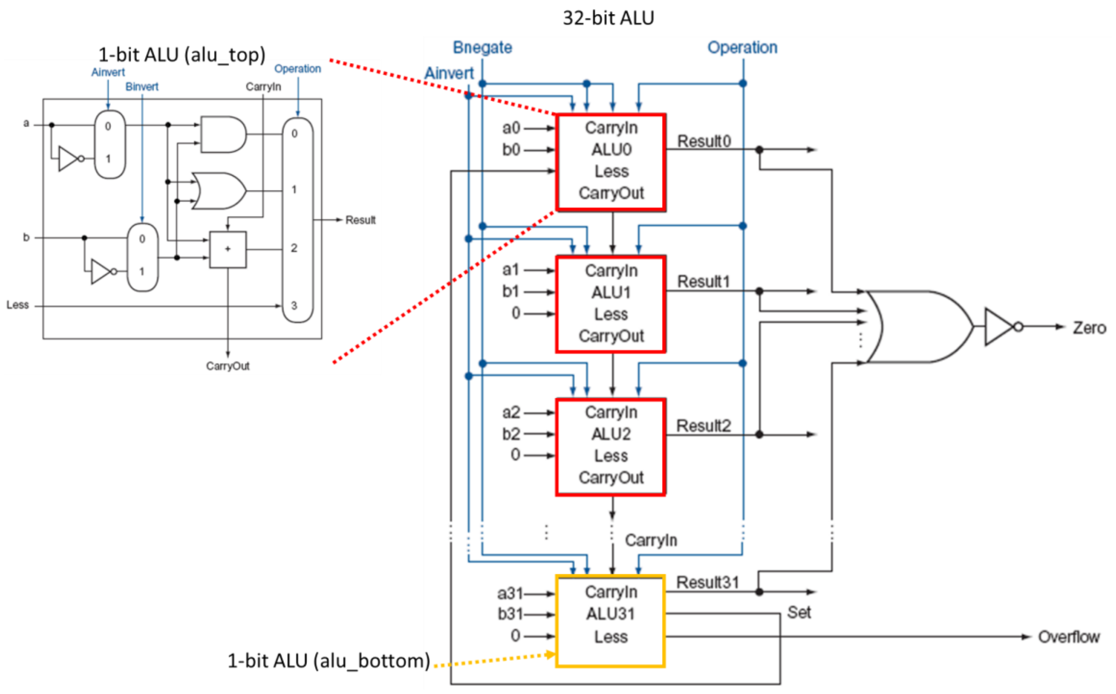
(6) ZCV three flags: zero, carry out, and overflow (30%)

zero: must be set when the output is 0

cout: must be set when carry out is 1

overflow: must be set when unsigned overflow happens

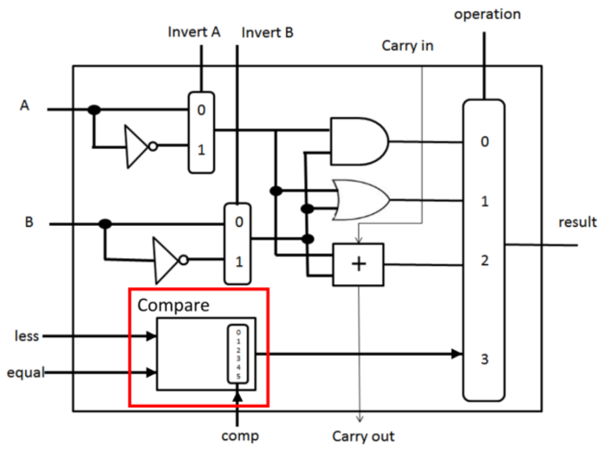
3. Architecture diagram



4. Bonus: Extra instruction set (10%)

ALU action	Name	ALU control input
Sltu	Set less than	0111_000
Sleu	Set less equal	0111_001
Sne	Set not equal	0111_010
Seq	Set equal	0111_011
Sgeu	Set greater equal	0111_111
Sgtu	Set greater than	0111_110

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



5. Grade

(1) Total: 110 points, including 10 points for a report (use CO_Report.docx)

(2) Late submission: 10 points off per day

(3) No plagiarism, or you will get 0 point.

6. Hand in

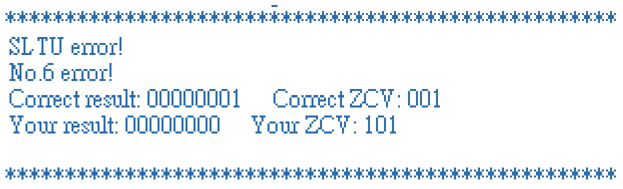
(1) Zip your folder and name it as "ID1_ID2.zip" (e.g., 0516001_0516002.zip) before uploading to e3. Other filenames and formats such as *.rar and *.7z are NOT accepted! Multiple submissions are accepted, and the version with the latest time stamp will be graded.

(2) Please include ONLY Verilog source codes (*.v) and your report (*.doc or *.pdf) in the zipped folder. There will be many files generated by the simulation tool (Xilinx) - do not include them; WE NEED ONLY VERILOG SOURCE CODES AND YOUR REPORT!

7. How to test

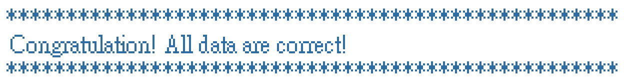
The function of testbench is to read input data automatically and output erroneous data. Please put all the .txt files and project in the same folder, after simulation finishes, you will get some information.

Partial error:



```
****************************************************
SLTU error!
No.6 error!
Correct result: 00000001    Correct ZCV: 001
Your result: 00000000    Your ZCV: 101
****************************************************
```

Or all cases pass:



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

8. Q&A

For any questions regarding Lab 1, please contact 黃甯琪 (blackitty321@gmail.com).