Computer Organization, Fall, 2013

Lab 1: 32bits ALU

Due: 2013/10/14

1. Goal

The goal of this LAB is implement a 32bits ALU (Arithmetic Logic Unit). ALU is the computing component of CPU. It's operation are such as AND, OR, add and subtract. This LAB will help you to understand the CPU architecture. LAB1 will be reused; you can use this module in later LABs. The function of testbench is read input data automatically and output which data is not pass. Please unzip the files in the same folder.

2. HW Requirement

a. Please use ModelSim PE Student Edition 10.2c as your HDL simulator.
(Here is the guide written by TA https://sites.google.com/site/nctuco/lecture-notes)

b. One people form a group. Please attach your name and student ID as comments in the top of each file.
The assignment you upload on e3 must have the form of "student ID.rar ".

c. Testbench modules have supplied.

d. Any assignment work by fraud will get a zero point.

e. The names of top module and IO ports must naming as following:

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".

f. instruction set: basic operation instruction。(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**

zcv three control signal : zero、carry out、overflow(30%)

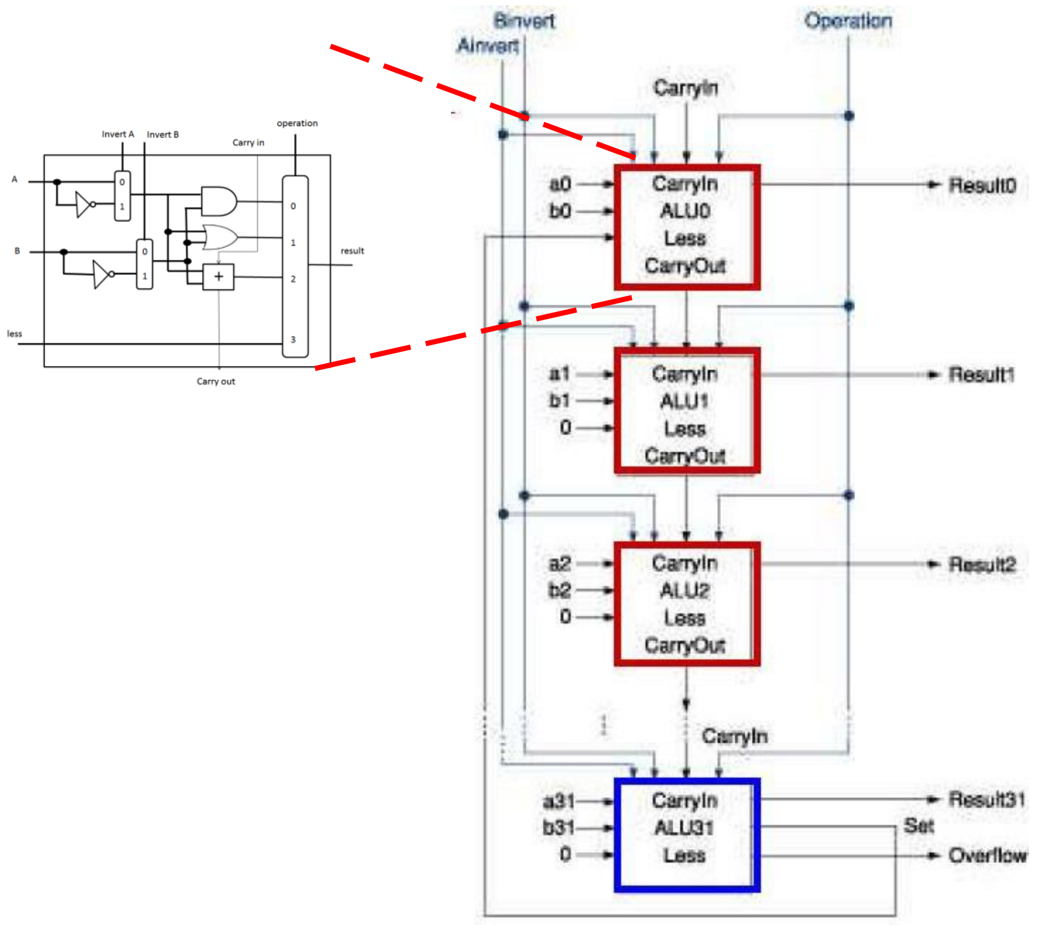
zero must be set when the result is 0.

cout must be set when carry out.

overflow must be set when overflow.

3. Architecture diagram

1- bit ALU (Top) **32-bits ALU**



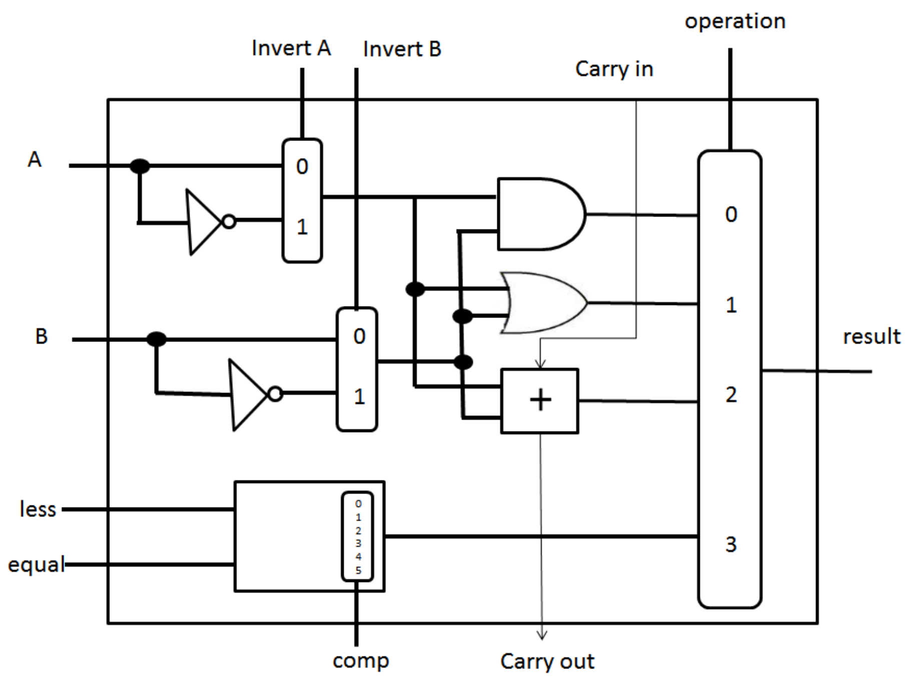
Blue frame is 1-bit ALU(Bottom)

4. Bonus: Extra instruction set, will get extra score if you do successfully。(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 non equal	0111_100

Hint: Add a module named Compare in 1-bit ALU, it needs extra 3 bits control input – Compare_sel

There is a reference architecture diagram.



5. Grade

a. Total: 110 point, **copy will get 0 point!**

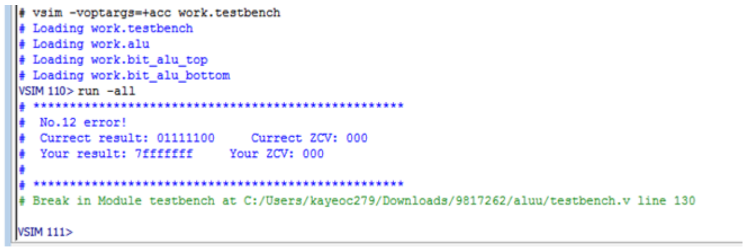
b. Document: 10 point

c. **Delay: 10% off/day**

6. Hand in

E3 : http://dcpc.nctu.edu.tw/

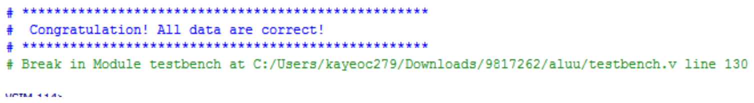
Please put all the .txt files and project in the same folder, after simulation finishes, you will get some information.



```
# 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
#
# **************************************************
# Break in Module testbench at C:/Users/kayeoc279/Downloads/9817262/aluu/testbench.v line 130

VSIM 111>
```

Partial error



```
# **************************************************
#  Congratulation! All data are correct!
# **************************************************
# Break in Module testbench at C:/Users/kayeoc279/Downloads/9817262/aluu/testbench.v line 130

VSIM 114>
```

All case pass