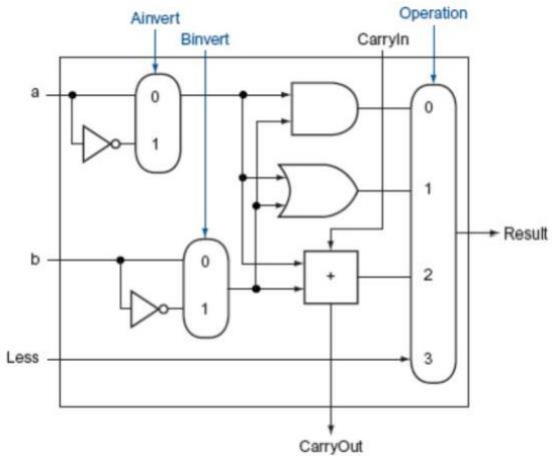
Computer Organization

Lab 1: 32-bit ALU

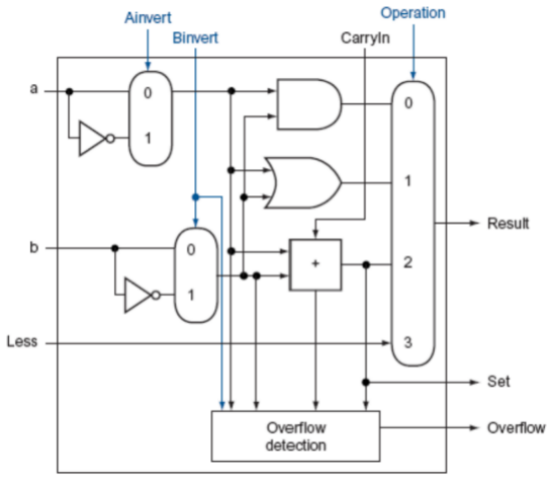
Student ID: 110550085 Name: 房天越

1. Architecture diagrams

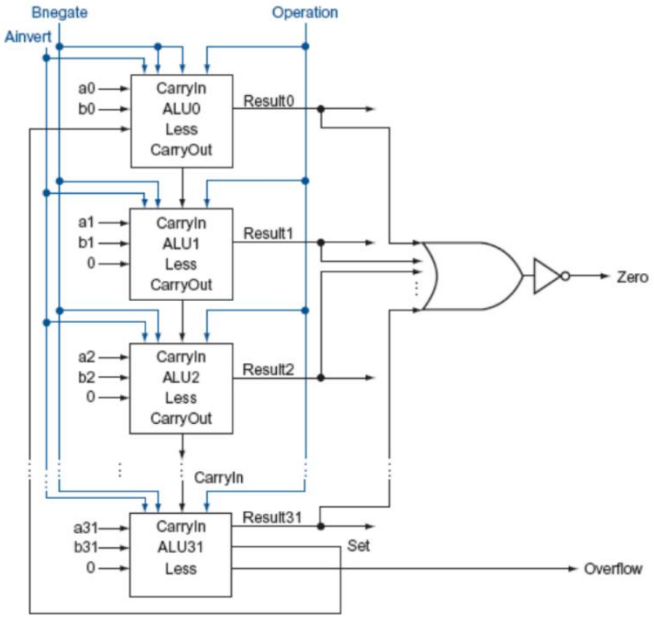
1-bit ALU (first 31 ones):



1-bit ALU (the last one):



the whole 32-bit ALU:



2. Hardware module analysis

There are three modules in my design, which are alu.v, alu_top.v, alu_top_overdetect.v:

alu.v:

This module contains the main 32-bit ALU design, with the first 31 bit and the last special one. This module split the opcode into 4 pieces, and send each pieces to the 1-bit ALUs. Also, this module has the functionality to determine the ZCV outputs by the result and carry bits.

alu_top.v:

This module implements the functionality of 1-bit ALU, and determines the result and cout by the operation code. First, it set src11 to A_invert xor src1, and src21 to B_invert xor src21.

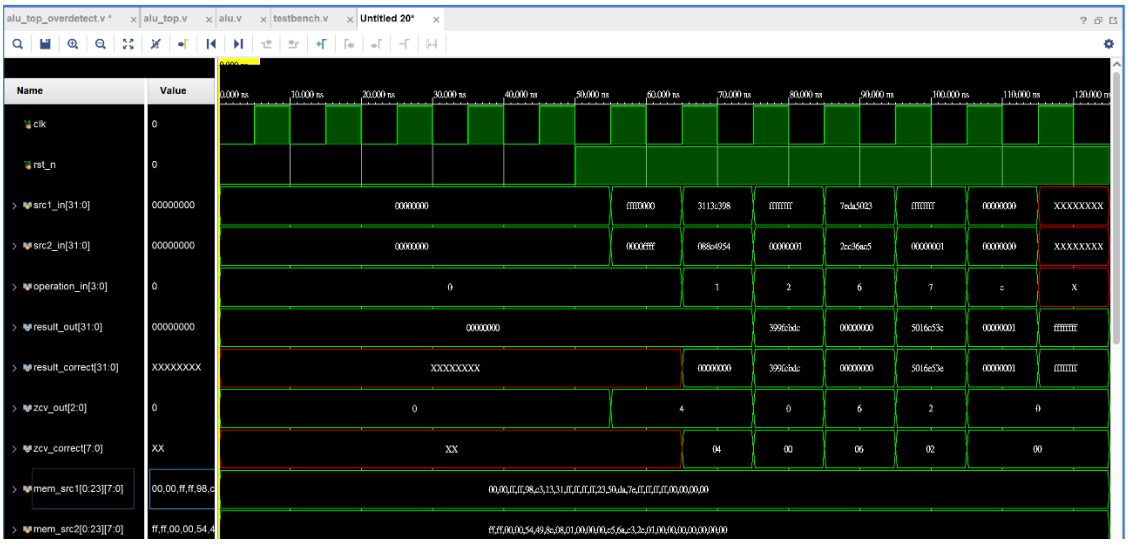
Then, when operation code is 00, do and operation. When operation code is 01, do or operation. When operation code is 10, do the full adder operation. When the operation code is 11, set the result equals to less, and also do the full adder operation to determine cout.

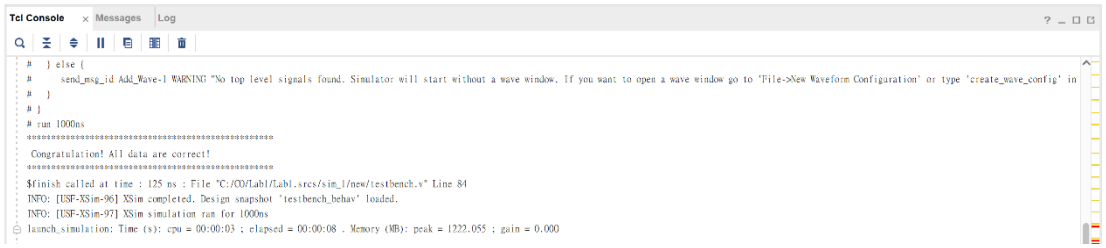
alu_top_overdetect.v:

This module is almost the same with alu_top.v, the only difference is that it has a set operation, which equals to src11 xor src22 xor cin. This is used to determine the result of set less than function.

I originally tried to detect overflow with this module, so I gave it this name, however, after I understand more I realized that I can use carry[30] and carry[31] to determine it.

3. Experimental result





As the wave figure and the Console shows, my results are correct.
For example, when src1_in is ffff0000, src2_in is 0000ffff, and operation_in is 0, I do and operation, so result is 00000000 and zcv is 4 (100) because the result is zero, no cout and no overflow.
For another example, when src1_in is 3113c398 and src2_in is 088e4954, I do or operation, so the result is 399fcbdc, and zcv is 0 (000) because output is not zero, no cout and no overflow.

4. Problems you met and solutions

Since I took the DCLab course in the last semester, didn't know how to debug, got only 10 points in midterm exam because of a dumb multidriven mistake and had to withdraw, I am kind of scared of Verilog now, so this time the biggest problem is to break through the fear.
Luckily, it went quite well this time, I found that for all the problems I met, their solutions are right there on the spec pdf file. For example, I didn't know what zcv meant, and I didn't know how to make the text files into Vivado, and the solutions are right there.

5. Summary

After this Lab, I have a better comprehension of how the 32-bit ALU runs and know how to combine the 1-bit units to make a larger device.
Also, with this experience, I am now less afraid of Verilog, and I hope in the future labs, I would have the ability to deal with the challenges and regain my confidence in this field.