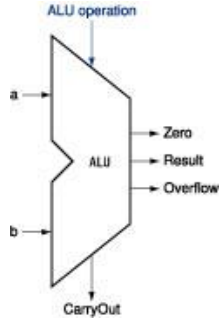
Computer Organization 0716074

■ Architecture diagram:

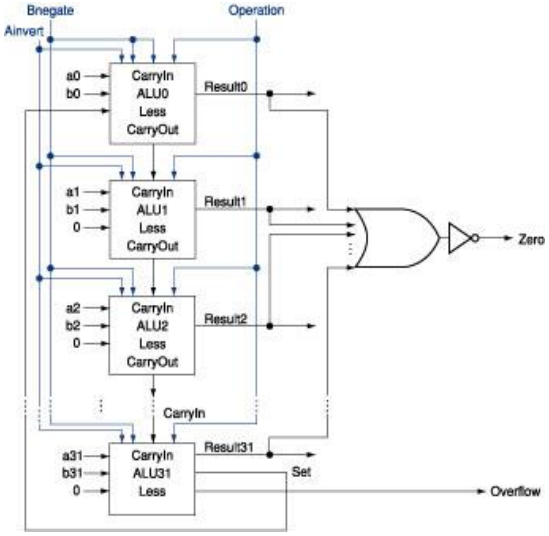
1. The structure of 32-bit ALU



- 本次實作的 32 位元 ALU，有：

Input: **a, b、op code**
Output: **result**
Flag: **Zero, Carryout, Overflow**

2. alu.v

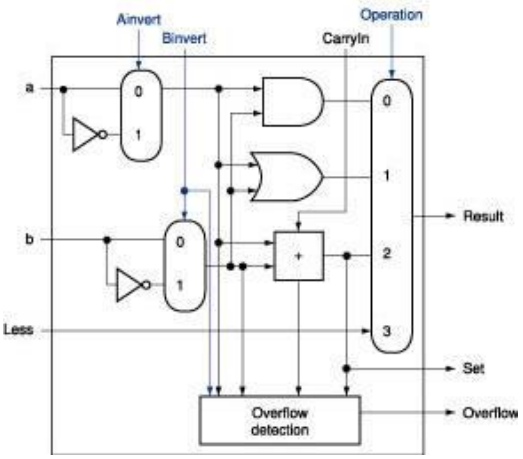


- ALU 內部的 32bits 實作上分成兩大部分：

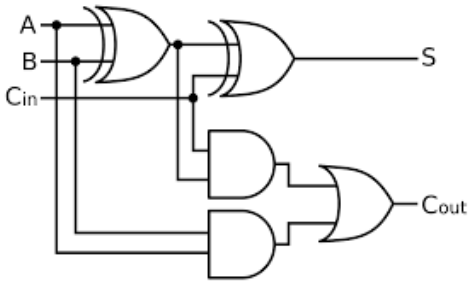
(1) 最後一個 bit 的 1-bit ALU (File name is ***alu_last.v***)

(2) 其他的 1-bit ALU (File name is ***alu_1bit.v***)

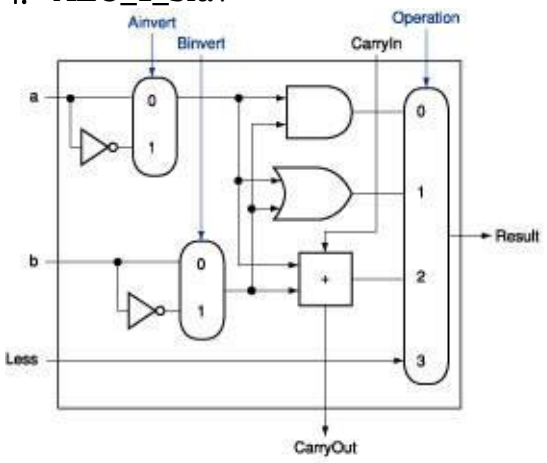
3. alu_last.v



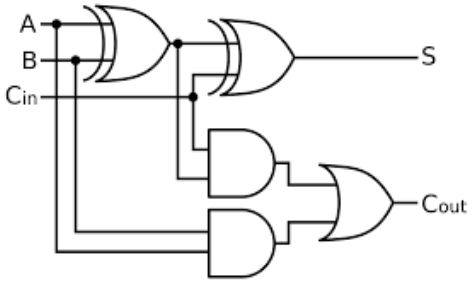
- 需要實作一個 **full adder (full_adder.v)**



4. **ALU_1_bit.v**



- 使用到相同的 **full adder (full_adder.v)**



■ Detailed description of the implementation:

A. Opcode

ALU action	Function	ALU control input
and	AND	0000
or	OR	0001
add	Addition	0010
sub	Subtract	0110
slt	Set less than	0111
nor	NOR	1100
nand	NAND	1101

- Opcode 分成兩大部分：
  i. 前 2 bits 用來決定是否將 input a, b 做 invert:

	A_invert	**B_invert**
00	0	0
01	0	1
10	1	0
11	1	1

  ii. 後 2 bits 用來決定 4x1 MUX 哪一個要當 result:

operation[2-1:0]	action
00	AND
01	OR
10	Addition
11	Set less than

B. Detail of *ALU_1bit.v*

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

- 除了上述的 ***input A_invert, B_invert, operation[2-1:0]***，還加入了 addition 和 subtraction 會用到的 ***input Cin, output cout***，以及輸出該 bit 結果的 ***output result***，並且另外加入 ***set***(預設為 0)，給 set less than 使用，當 ***opcode = 11*** 時，***result*** 都設為 ***set***

C. Detail of *ALU_last.v*

```
module ALU_last(
        input     src1,      //1 bit source 1  (input)
        input     src2,      //1 bit source 2  (input)
        input     A_invert,   //1 bit A_invert  (input)
        input     B_invert,   //1 bit B_invert  (input)
        input     Cin,       //1 bit carry in  (input)
        input     [2-1:0] operation,  //2 bit operation
        input     set,
        output reg    result,     //1 bit result
        output wire   cout,       //1 bit carry out
        output wire   overflow,
        output wire   less_out
  );
```

- 最後 1bit 的 ALU 多了兩個 output:
- ***overflow***
  最後一個 bit 的 ***Cin*&*cout***
- ***less_out***
  用在 set less than，把最高位加減的結果(也就是 signed bit)作為判斷 a<b 的依據，並將 ***less_out*** 拉到最低位 bit 的 set，即：若 a<b，則 ***less_out=1***，拉到第一位的 ***set*** = ***1***，最後此 32bits 的 ***result*** 就會輸出 **000…0001**

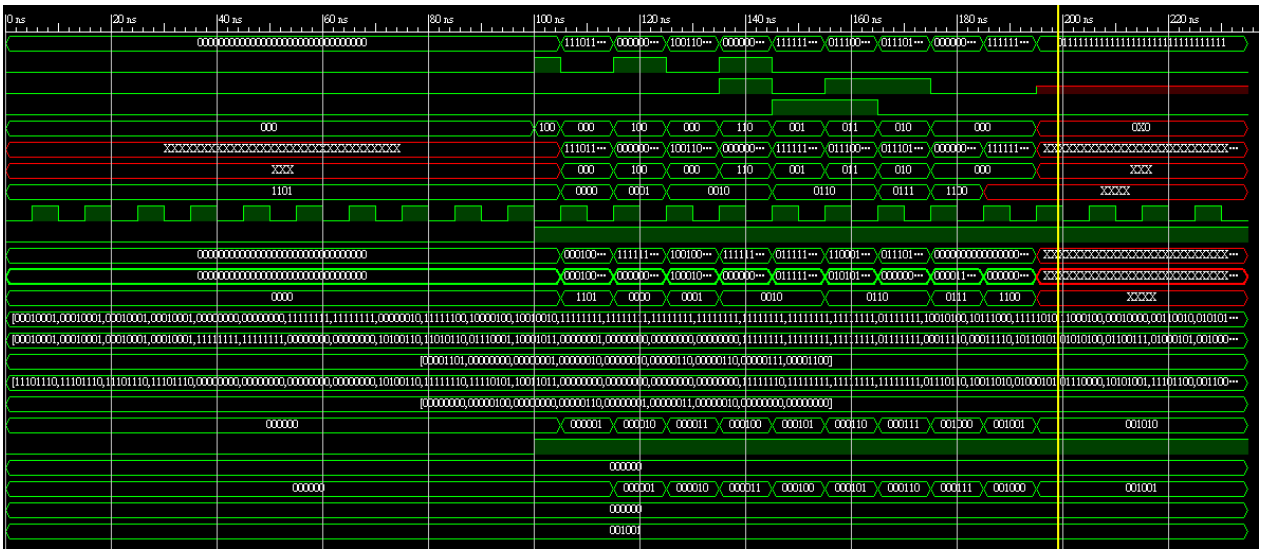
D. Detail of *alu.v*

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

- 本次實作的 32 位元 ALU：
- Input: ***src1, src2***、***ALU_control***
- Output: ***result***
- Flag: ***zero, cout, overflow***

■ Implementation results:

I. 波形圖



II. 結果

```
Console
ISim P.20131013 (signature 0x7708f090)
WARNING: A WEBPACK license was found.
WARNING: Please use Xilinx License Configuration Manager to check out a full ISim license.
WARNING: ISim will run in Lite mode. Please refer to the ISim documentation for more information on the differences between the Lite and the Full version.
This is a Lite version of ISim.
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.
*****************************************************
*         PATTERN RESULT TABLE            *
*****************************************************
*PATTERN *          Result          *ZCV *
*****************************************************
*     Congratulation! All data are correct!     *
*****************************************************
Correct Count: 9
Stopped at time : 205 ns : File "C:/Users/Cool Jacky/Desktop/wake_course/Organization/ChenTienFu/lab/Computer Organization Lab2/testbench.v" Line 146
```

■ Problems encountered and solutions:

我花了大部分的時間在 debug，我唯一錯誤的是第八筆測資，是關於 set less than 的，不過經過了超久的 debug，最後錯誤竟然在要設定 B_invert 那裏少加入了 ALU_SLT 這個 ALU control。

■ Lesson learnt (if any):

I. Divide & Conquer

讓自己能夠更輕鬆 debug 的方法就是把每一個物件都排列整齊，利用各種 module 去實作，而不是直接用基本邏輯式開始硬幹，像是有些 selection 有時候你把它用 mux 去做會比每次遇到都用 always block 更清楚一些。

II. Learn all of ALU

經過本次實作，我已經完全了解並且記得所有關於 ALU 的架構。

III. Verilog coding

以前老師只有教紙本的 code，現在自己實際寫一次就真的不用硬記了，很自然地就可以看懂 verilog 並使用。

■ Comment:

真心感謝老師和 TA 精心準備這次的 lab，我獲益匪淺，也克服了以前對於機器語言頗難的想像！