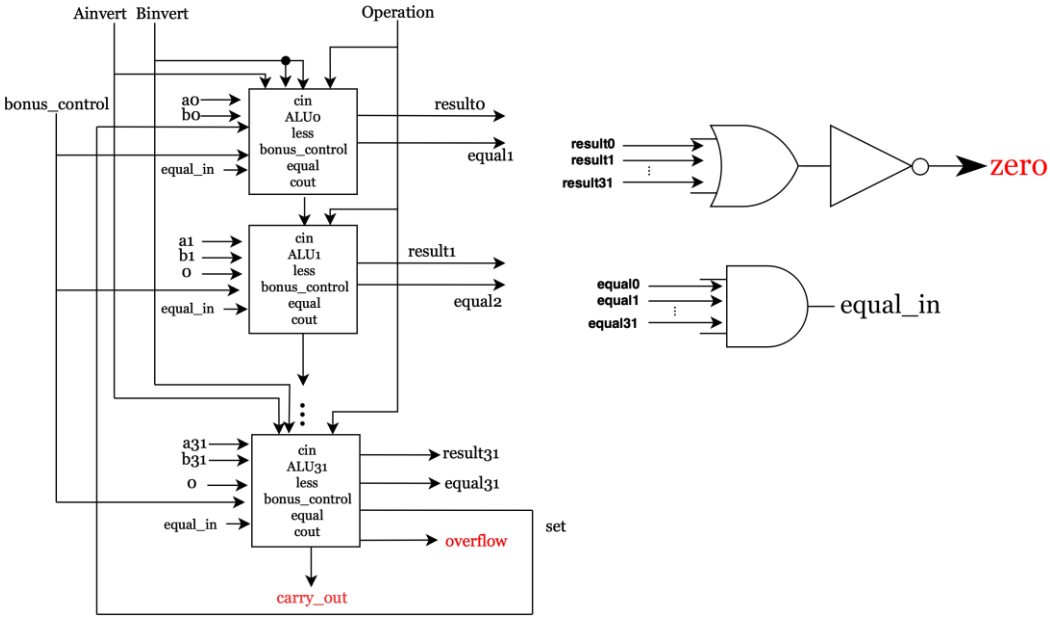
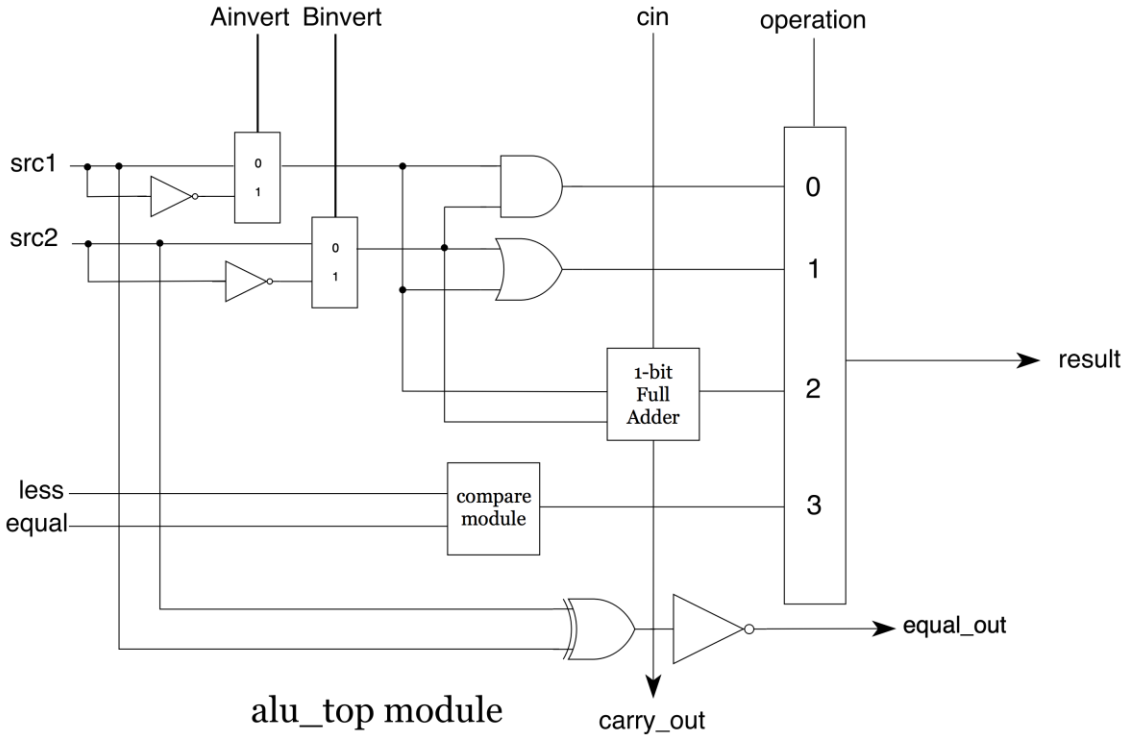
0416005張彧豪、0416315王定偉

Computer Organization

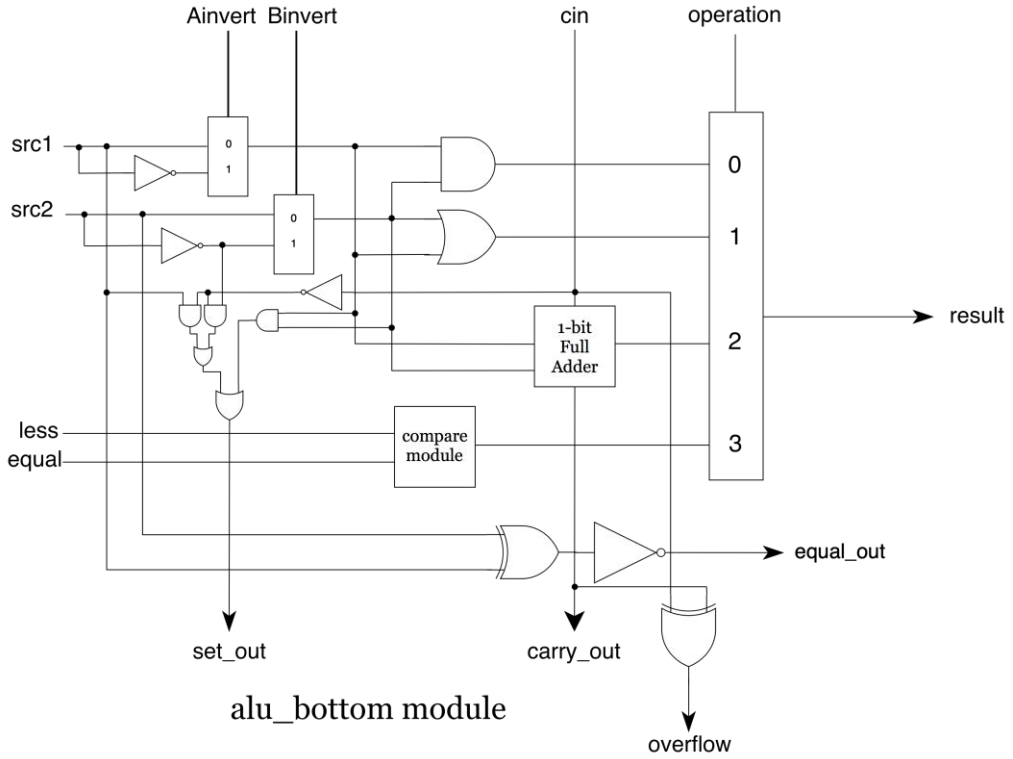
Architecture diagram:



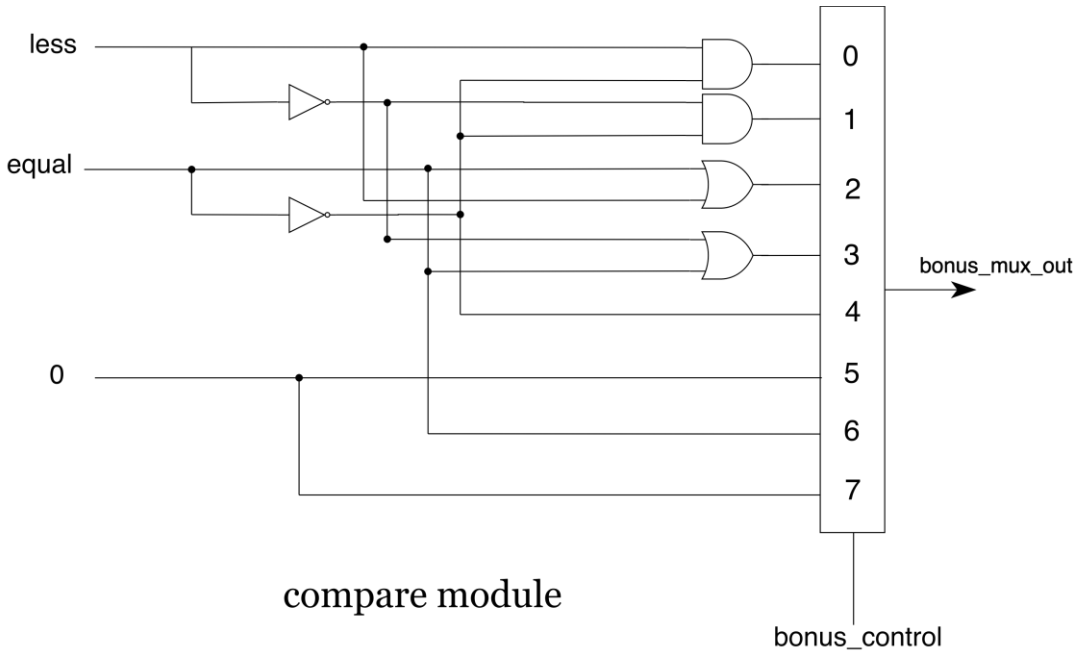
▲圖 1：ALU MODULE



▲圖 2：ALU_TOP MODULE



▲圖 3：ALU_BUTTOM MODULE



▲圖 4：COMPARE MODULE

Detailed description of the implementation:

一、以 alu 為主要模組，此模組會去呼叫 alu_top 及 alu_bottom 模組，top 的模組用在 bit1~31，bottom 模組用在 bit32，會有這樣的差別是在於 set 訊號及 overflow 訊號的計算只需在 MSB 處進行即可，所以將 1-bit ALU 分成兩種來做。在 conditional operation 的部份我們是將 equal 及 set 訊號回接 LSB 的 alu，用傳入的 op code 及

bonus_control 去決定最終的結果。alu 此模組要做的事情就是將各 alu 傳回來的 result 接到整個 32-bit alu 的輸出、設定 cout 以及判斷 result 是否為 0。

二、1-bit ALU 內部實作及部分訊號判斷方法：

1. overflow 的判斷方法：觀察兩個在做加法的運算元之 MSB 的 carry_in 跟 carry_out 可以發現，當這兩個訊號一個為 0 一個為 1 時，表示有 overflow 發生，觀察可由下表得出：

Src1	Src2	cin	Result’s MSB	cout	Overflow?
0	0	1	1	0	Yes
0	0	0	0	0	No
0	1	1	1	1	No
0	1	0	0	0	No
1	0	1	1	1	No
1	0	0	0	0	No
1	1	1	1	1	No
1	1	0	0	1	Yes

一正一負相加絕對不會造成 overflow

由上表可以得知，判斷 overflow 可以用 MSB 的 cin 及 cout 判斷，overflow=cin⊕cout。

2. Invert 的判斷方法：觀察所有需要有 invert 訊號的 control code，發現到它們都有一個共通點，就是由右數過來第三個 bit 皆為 1。因為這個特性，我們以此當作 invert 訊號，1 代表需要 invert，0 代表不需 invert。
3. 減法的實作：減法轉成加法最重要的就是二補數的轉換，目前已經有 invert 訊號，差的是 invert 完後的加 1 這個動作，而這個動作我們是利用 LSB 的 cin 去達成的，將 LSB 的 cin 設為 1 並傳入 ALU。要將 cin 設為 1 也是利用觀察法得到的，所有需要用到減法的運算的 control code 的中間兩個 bit 皆為 1，故我們將 LSB 的 cin 輸入設為 ALU_control[2]&&ALU_control[1]。
4. 判斷相等的方法：

   我們判斷相等是採用一個 bit 一個 bit 判斷，每個 1-bit ALU 皆會輸出送入該 ALU 的兩 bit 是否相等，接著再將這 32 個取 AND 運算，將得到的結果指派到 equal_in 上，若為 1，表示兩數相等，反之為 0。

   equal_in=src1⊕src2。
5. Slt 運算實作方法：

   這部份我們利用的減法的概念去實作 slt，而要判斷 A 是否小於 B 時，只要用 result 的 sign bit 即可判斷，但要注意如果當原本兩運算元就不同號時，set 訊號就必須強制為 0 或 1，以此避免因為溢位而產生錯誤的 set 訊號。這個 set 訊號我們是用 MSB 的 cin 與 MSB 的 src1 與 src2 計算出來，因為是用減法所以會將 src2 取 invert，把取過 invert 的 src2 稱作 real_src2。

   下表是我們如何判斷 set 訊號的方法：

cin(MSB)	src1(MSB)	real_src2(MSB)	Set	說明
0 or 1	0	0	0	正數比負數大,set=0
0	0	1	1	Result sign bit=1,set=1
1			0	Result sign bit=0,set=0;
0	1	0	1	Result sign bit=1,set=1;
1			0	Result sign bit=0,set=0;
0 or 1	1	1	1	負數比正數小,set=1

將上表經由 K-map 化簡後可以得到下式：

set=((~cin)&real_src2)|((~cin)&src1)|(src1&real_src2)。

6. 其他的 condition instruction 的實作

這部份我們是利用 less 訊號與 equal_in 還有 bonus_control 這三個訊號，因為 less 訊號是指 A 有沒有小於 B，而 equal_in 是指 A 有無等於 B，利用這兩個資訊並搭配 bonus_control 就可以將算出對應的 result。實際作法是將該三個訊號傳入 LSB 對應的 ALU 中並用 compare 模組去選出我們需要的 result。

下表為 bonus_control 與對應的答案算法：

Operation	bonus_control	Result 接法
SLT	000	less&(~equal_in)
SGT	001	(~less)&(~equal_in)
SLE	010	less|equal_in
SGE	011	(~less)|equal
SEQ	110	equal
SNE	100	~equal

7. Compare module

此模組是將 equal、less 及 bonus_control 訊號傳入並傳出相對應的訊號，內部實作是用一個 8-1 mux，bonus_control 為選擇線，八個輸入，以下表表示蕭對應的選擇線與輸出：

Select	Output
000	less&(~equal)
001	(~less)&(~equal)
010	less|equal
011	(~less)|equal
100	equal
101	0
110	~equal
111	0

Problems encountered and solutions:

1. 對於 K-map 以及化簡部分較生疏，查了一下以前數電的課程才找回那種感覺。
2. 一開始其實不太知道要怎麼設定 conditional operation 的訊號跟 set 訊號是要幹嘛的，只知道應該跟減法有關，最後是在網路上看到關於 slt 的文章，看了一下原來只要判斷 MSB 的 cin 與兩個 MSB 運算元之間的關係就可以得出 set 訊號了。
3. 原本判斷兩數的相等的方法是用減法的 result 判斷，但發現這樣有點不合理，如果它的運算根本不是減法或是只是利用的減法但結果卻不是減法的運算就不能用了，最後只好在每個 1-bit alu 多加 equal_out 的訊號輸出，再將 32 個 equal_out 訊號 AND 起來看是不是真的相等。

Lesson learnt (if any):

1. Overflow 的判斷方法。
2. Conditional operation 的實作方法。
3. 重溫 ISE 模擬的環境，並再次熟悉該環境。