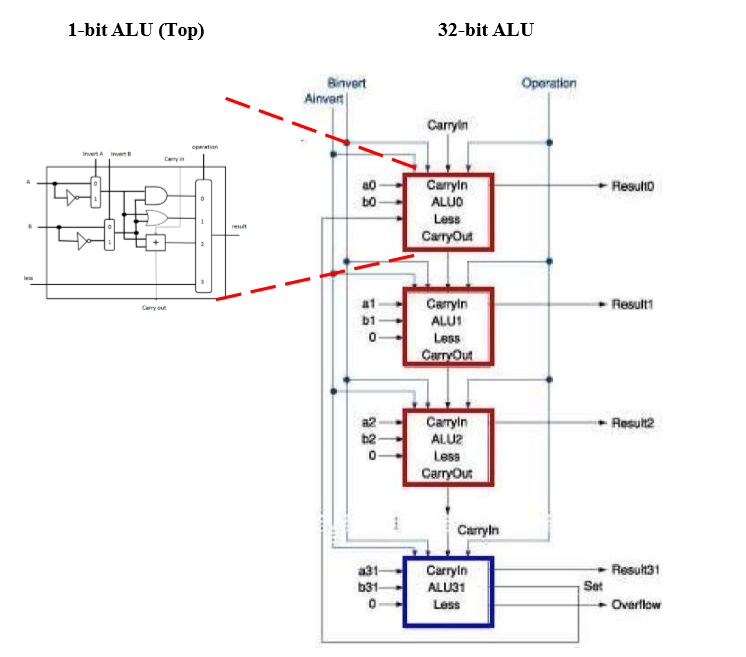
**Computer Organization**

**0516303 余采彧**

**0516208 黃郁恬**

**Architecture diagrams:**

****

**32-bit的alu是用32個1-bit alu完成，不過我們的方法並非寫32個function，而是用generate-for語句來實現**

**Hardware module analysis:**

**alu\_top:**

**src1\_temp: src1 跟 A\_invert 經過多工器(MUX)出來的結果**

**src2\_temp: src2 跟 B\_invert 經過多工器(MUX)出來的結果**

**cout: 在and和or時候，不需考慮進位🡪0**

**在add和比較大小時，則需考慮**

**🡪 (cin&src2\_temp) | (cin&src1\_temp) | (src1\_temp&src2\_temp)**

**(畫真值表後化簡出來的結果)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **src1** | **src2** | **cin** | **plus** | **cout** |
| **0** | **0** | **0** | **0** | **0** |
| **0** | **0** | **1** | **1** | **0** |
| **0** | **1** | **0** | **1** | **0** |
| **0** | **1** | **1** | **0** | **1** |
| **1** | **0** | **0** | **1** | **0** |
| **1** | **0** | **1** | **0** | **1** |
| **1** | **1** | **0** | **0** | **1** |
| **1** | **1** | **1** | **1** | **1** |

**Operation:**

**2'b00: and**

**2'b01: or**

**2'b10: add**

**2'b11: slt**

**alu:**

**cin[0]=ALU\_control[1]&ALU\_control[2]:**

**SUB和SLT(用到減法) ，都要取二補數(轉成1's complement然後再加1)**

**用generate-for語句來實現32個1-bit alu:**

**less=(src1[31]^ALU\_control[3]) ^ (src2[31]^ALU\_control[2]) ^ cin[31]**

**(這個運算在高位元的31個bit都是一樣的，都為0，所以只有在第一個位元需要另外設值)**

**第一個bit:**

**Less就是A加上B的invert，所以只要最後一位的A , B’, cin有奇數個1，就表示做減法之後的結果是1**

**.A\_invert(ALU\_control[3]):**

**A\_invert在做NOR時是1(因為NOR是把A和B取invert再AND起來)，其餘都是0**

**.B\_invert(ALU\_control[2]):**

**B\_invert在做NOR, SUB, SLT(相減後的結果來判斷)時會是1**

**.operation(ALU\_control[1:0]):**

**AND: 0000 🡪 and**

**OR: 0001 🡪 or**

**ADD: 0010 🡪 add**

**SUB: 0110 🡪 相當於add src\_2的補數**

**NOR: 1100 🡪 相當於把輸入的invert and起來**

**SLT: 0111 🡪 slt**

**overflow<=cin[31]^cin[32]:**

**會發生overflow的狀況:**

1. **正加正/正減負:**

**正加正必為正，也就是最高位元會是0，但是若有進位的話，最高位元就會是1(cin[31])**

1. **負加負/負減正:**

**負加負必為負，也就是最高位元會是1，如果carry in是0的話，最高位元就會是1+1 = > 0**

* + **結論: 最高位元的carry in (cin[31])和carry out (cin[32])不同時**

**zero <= (tmp == 32'b0) ? 1'b1 : 1'b0:**

**zero為1時，表示result(32-bit)等於0 ;當zero 為0時，表示result(32-bit)不全為0**

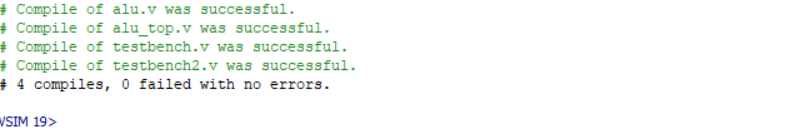
* **簡單地判斷兩暫存器的值究竟相等不相等**

**cout <= (ALU\_control == 4'b0111)? 1'd0 : cin[32]:**

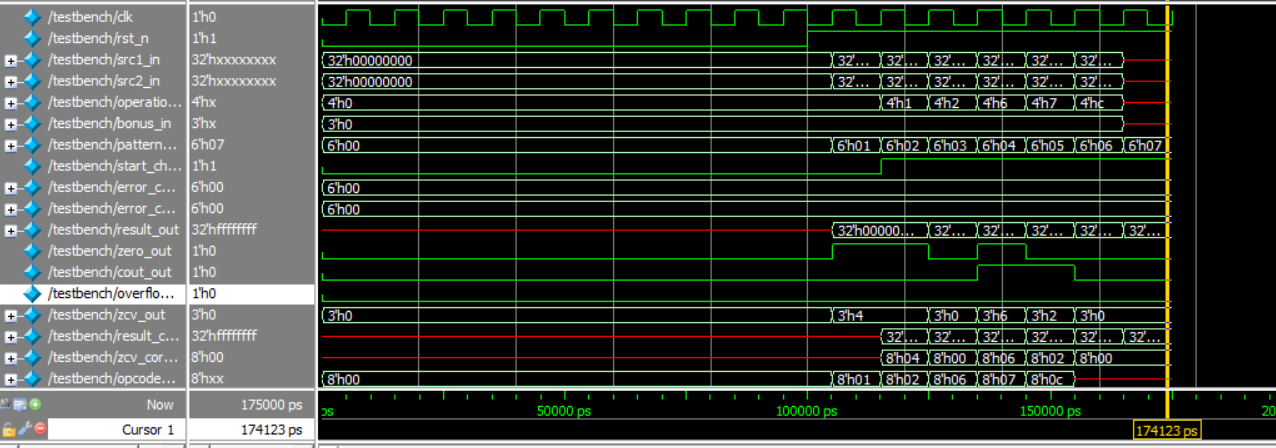
**在做SLT時(src1比src2小時，設為1; 否則設為0)，最高的31位元都是0**

**Experiment result:**

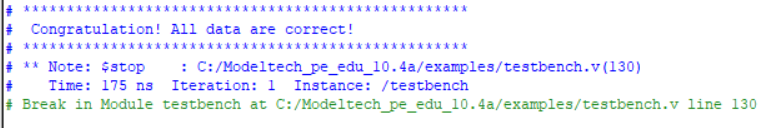
**成功編譯:**

****

**波形圖:**

****

**成功模擬:**

****

**Problems you met and solutions:**

1. **分不清楚何時要宣告成wire，何時要宣告成reg**

* **Sol: 找助教求救，得知放等式左邊的要是reg，因為要把線路(wire)的值寫進reg中，線路是用來連接電路，而非儲存值的。**

1. **從1-bit轉成32-bit時，zero和overflow的設值以及sub**

* **Sol: 參考網站<https://chi_gitbook.gitbooks.io/personal-note/content/alu.html>**

1. **編譯出現error時，看不懂，因而不知道怎麼解決**

* **Sol: 問助教，或上網查遇到相同問題的人怎麼解決問題**

1. **讀檔案時，路徑問題**

* **Sol: 上網查，跟組員討論**

1. **模擬結果不正確時，不知道哪裡有誤**

* **Sol: 跟助教和同學討論，並且去試各種可能(真的是試到天荒地老，花的時間是寫的三倍)**

**Summary:**

**心得**

**以前數位電路作業時就有碰過verilog，作業都可以在一兩天內完成，沒想到寫計組作業時要花那麼多精力跟時間，有很多之前沒看過的用法、符號，期間也遇到各種意想不到的狀況，明明都一步一步trace了，明明都知道問題在哪裡了，卻是想破頭也不知道原因出在哪裡，該怎麼改正，真的上網查了很多很多資料，看了很多教學影片，助教跟同學也幫了很多忙，才終於可以成功完成，覺得心累(但確實有學到東西)!!!**

**建議**

**Modelsim真的不太好用(而且有些其他編譯器可以過的，modelsim都會出問題，然後又有版本問題，寫好了才發現因為沒有更新所以不能用)，希望可以開放用ubuntu的環境，另外推薦iverilog+gtkwave。**