**Computer Organization**

**Architecture diagrams:**

**1.設計1bit 的top-ALU。**

**2.透過設計好的1bit的top-ALU串接成32bit的ALU。**

**3.設計1bit的 bottom-ALU。**

**4.設計辨識zero、overflow及求出carry out的架構。**

**Hardware module analysis:**

**1.top-ALU:**

**透過四種不同的operation，分別進行不同的運算**

* **AND:使用&**
* **OR:使用|**
* **ADD: Result用 a^b^carry\_in 表示**

**Carry out 用(a&b) | (a&carry\_in) | (b&carry\_in)表示，化簡後變成 (a&b) | ((a^b)&carry\_in)**

* **SLT: 變相的加法，result用less表示**

**2.ALU**

* **rst\_n:**

**用一個temp的register來儲存運算完的result，再將temp放到輸出。**

* **fisrt carryin control:**

**除了有使用到減法的運算，其他運算都不需要first carryin的值，因此只有SUB和SLT的fisrt carryin為1。**

* **overflow control:**

**分成兩個部分討論**

**加法時overflow發生情況為**

**1.正數(scr1[31]==0)加正數(scr2[31]==0)結果得到負數(result[31]==1)**

**2.負數(scr1[31]==1)加負數(scr1[31]==1)結果得到正數(result[31]==0)因此用( src1[31] == src2[31] ) && ( alu\_result[31] != src1[31] )來判斷**

**減法時overflow發生情況為**

**1.正數(scr1[31]==0)減負數(scr1[31]==1)結果得到負數(result[31]==1)**

**2.負數(scr1[31]==1)減正數(scr1[31]==0)結果得到正數(result[31]==0)**

**因此用( src1[31] != src2[31] ) && ( alu\_result[31] != src1[31] )來判斷**

* **carry out:**

**分成兩部分討論**

**加法時可以將最後一個bit的cout當作carry out表示**

**減法時則是要考慮src1和src2的正負**

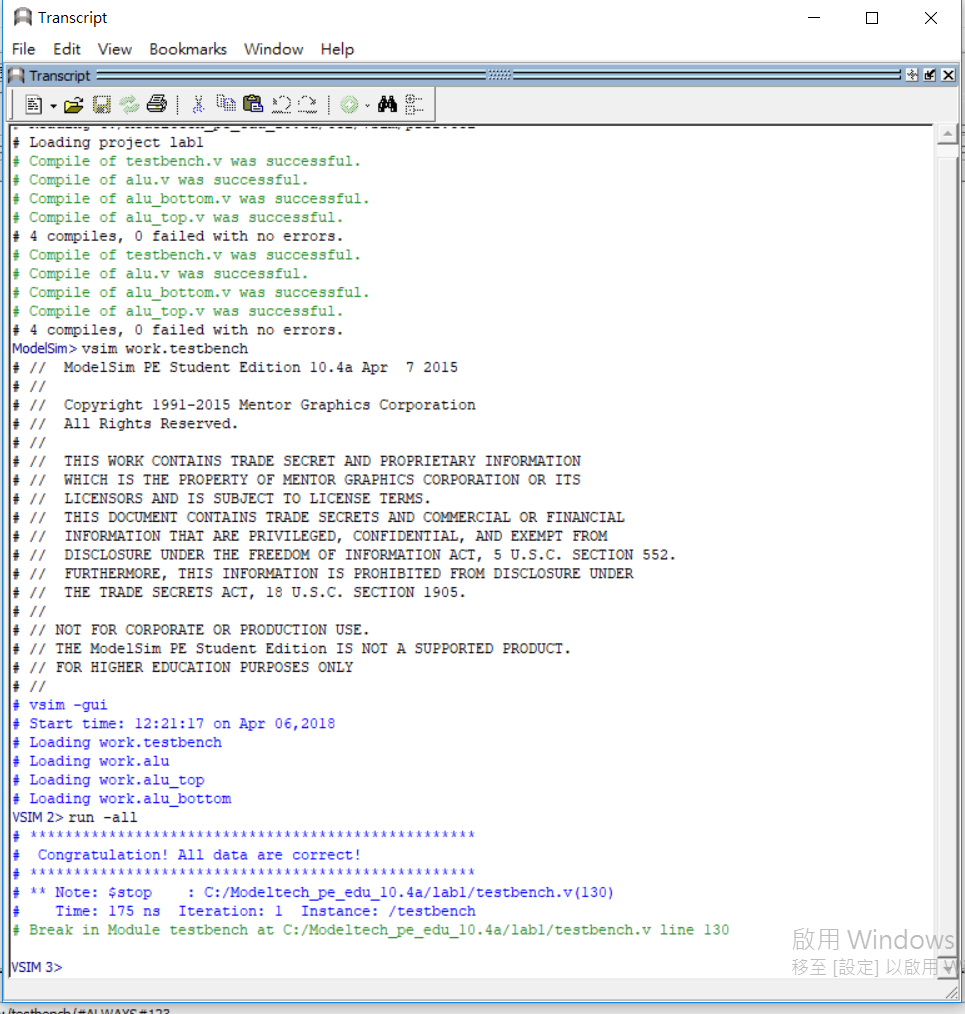
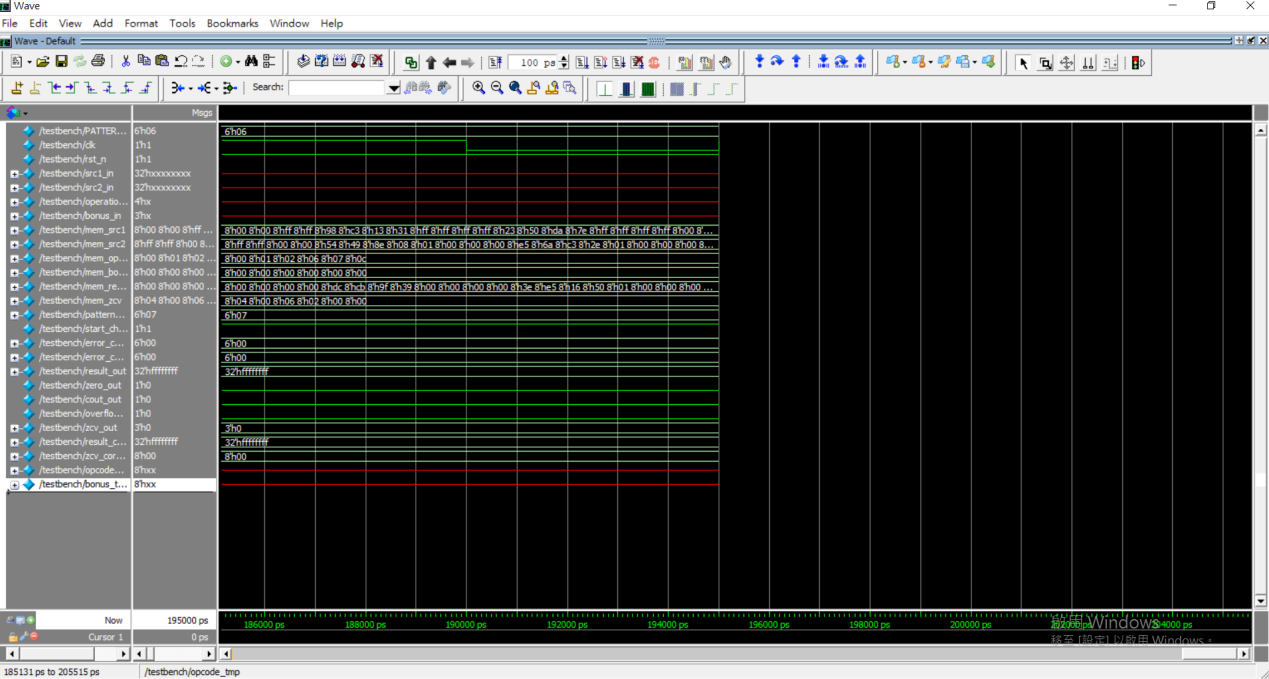
* **zero:**

**設條件判斷result是否為32’b0**

* **32bit ALU design :**

**除了第一個bit的carry\_in是透過上方fisrt carryin control給定外，其他bit均是使用前一個bit的carry out當作carry in，並串接成一個32位元的ALU**

**Experiment result:**

****

**Problems you met and solutions:**

**1.top-ALU:**

**Module括號內最後一項cout後方有一個”,”，進行simulate後通知我應該要有10個port，可是全部項目只有九個port，因為那個”，”讓系統認定後方還有一項，所以發生錯誤**

**2.32bit ALU:**

**原本carry out是學https://github.com/uttu357/alu-8bit/blob/master/adder\_8\_bit.v 裡面用ripple1、ripple2....來傳，後來發現會出現問題，因此將ripple改成wire的型態。**

**First carry in design有卡了一下，去找一下的數位電路設計的資料後才確定SUB的first carry in是1。**

**Overflow和carry out老師上課有說過，所以沒有什麼問題。**

**Zero判斷打反了，所以error上顯示剛好全部反過來，把條件後的zero\_control改過來就對了。**

**3.bottom-ALU**

**新增檔案時檔名bottom打成buttom，所以當ALU呼叫時顯示找不到檔案**

**Set一開始擺錯位置，和cout放反，改過來後就沒有錯了**

**Summary:**

**對於verilog語法的不熟悉，讓我在一開始摸索時花了很多時間，後來上網查了需多關於ALU的資料，才開始進行作業。**

**Reg與Wire:** **http://www.cnblogs.com/oomusou/archive/2007/10/10/919339.html**

**ALU基本概念:**

[**https://chi\_gitbook.gitbooks.io/personal-note/content/alu.html**](https://chi_gitbook.gitbooks.io/personal-note/content/alu.html)**(最詳細)**

**http://ccckmit.wikidot.com/ve:alu**

**ALU實作範例**

[**https://github.com/uttu357/alu-8bit**](https://github.com/uttu357/alu-8bit)

**我覺得之後出作業可以附上一些參考資料或網站，讓學生能更快找到方向。**

**Verilog使用的開發環境相對於c語言來的複雜許多，也更具有挑戰性，因此使我花了更久時間熟悉。**