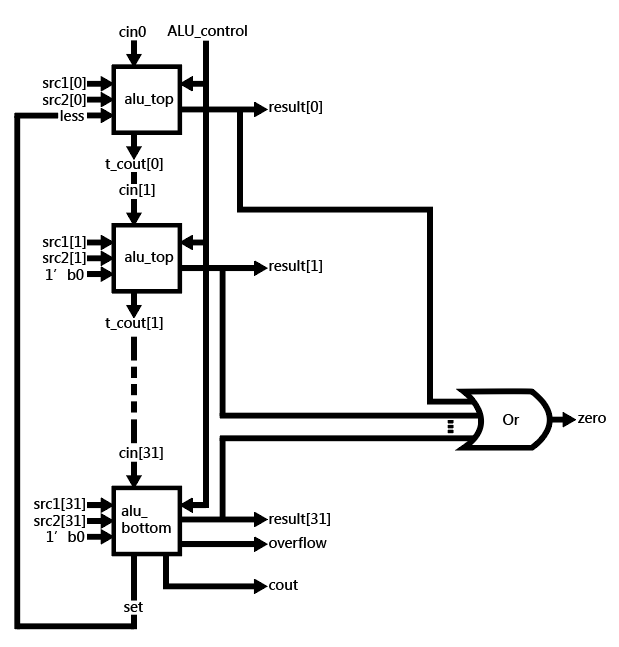
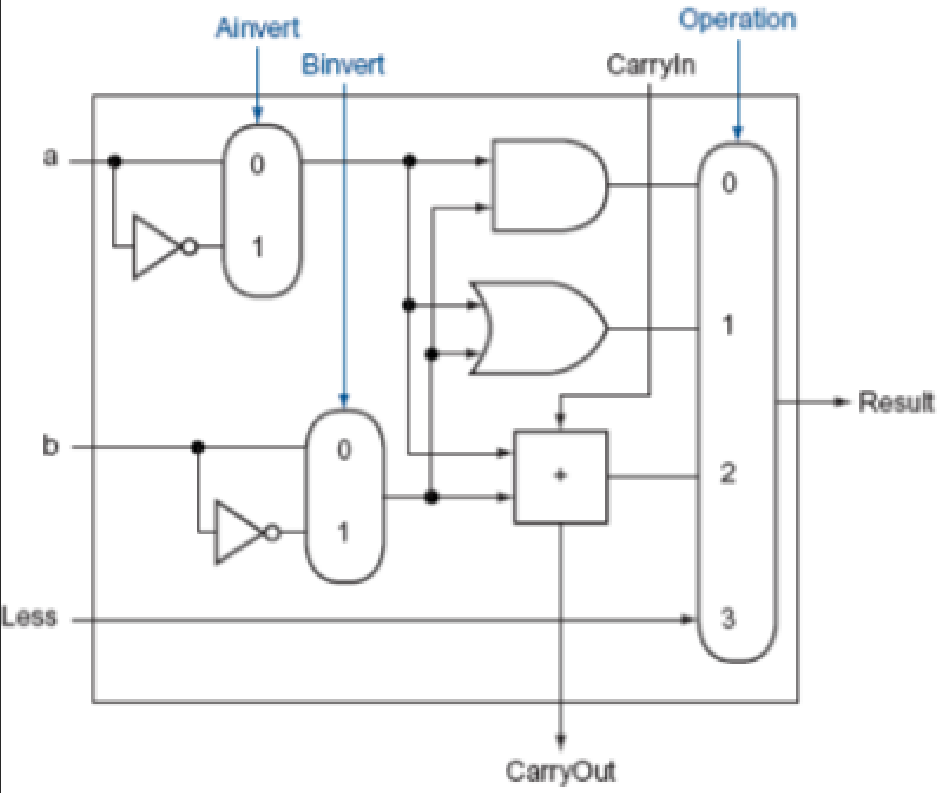
**Computer Organization**

**Architecture diagrams:**

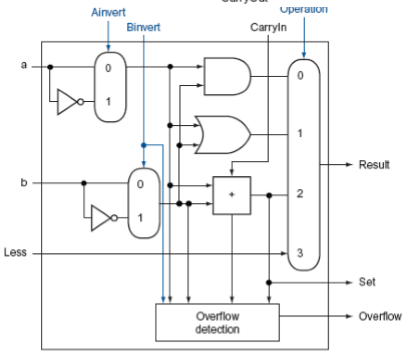
ALU:

****

ALU\_TOP:



ALU\_BOTTOM:



cout

**Hardware module analysis:**

ADD:

將scr1和scr2互相對應的bit做add，將結果存進result。Cout、Set、Overflow都設為0

OR:

將scr1和scr2互相對應的bit做or，將結果存進result。Cout、Set、Overflow都設為0

ADD:

將scr1、scr2和cin做exclusive or，結果存進result

先做src1 and src2。再做cin and (src1 or src2)。最後將兩者做or，結果存進cout

在最後一個alu單元(alu\_bottom)，決定overflow的方法是:先將src1和src2做exclusive or，判斷src1和src2是不是同樣是正數(或負數)。如果確定src1和src2同正負號，在判斷result和src1是不是同正負號，就可得overflow的值

Set設為0。

SUB:

將scr1、scr2\_invert和cin做exclusive or，結果存進result

先做src1 and scr2\_invert。再做cin and (src1 or scr2\_invert)。最後將兩者做or，結果存進cout

Set設為0。

在最後一個alu單元(alu\_bottom)，決定overflow的方法是:先將src1和scr2\_invert做exclusive or，判斷src1和scr2\_invert是不是同樣是正數(或負數)。如果確定src1和scr2\_invert同正負號，在判斷result和src1是不是同正負號，就可得overflow的值

SLT:

SLT的演算法是將兩者相減，如果結果為負，則SLT是1；結果為0或正數，則SLT為0。也就是說，SLT等於結果的signbit，因此會將Set值輸入給第一位alu\_top。除了第一位alu\_top的result由set決定。其餘result值為0

Cout為0

Overflow為0

Set的值等於相減結果的signbit。但為了避免overflow，所以當src1和src2不同正負號時，直接用正負號決定Set的值。反之，即可將src1和src2相減，獲得相減結果的signbit

NOR:

將scr1和scr2先做not。再將互相對應的bit做or，將結果存進result。Cout、Set、Overflow都設為0

**Experiment result:**

AND:

Currect result: 00000000 Currect ZCV: 100

My result: 00000000 My ZCV: 100

OR:

Currect result: 399fcbdc Currect ZCV: 000

My result: 399fcbdc My ZCV: 000

ADD:

Currect result: 00000000 Currect ZCV: 110

My result: 00000000 My ZCV: 110

SUB:

Currect result: 5016e53e Currect ZCV: 010

My result: 5016e53e My ZCV: 010

SLT:

Currect result: 00000001 Currect ZCV: 000

My result: 00000001 My ZCV: 000

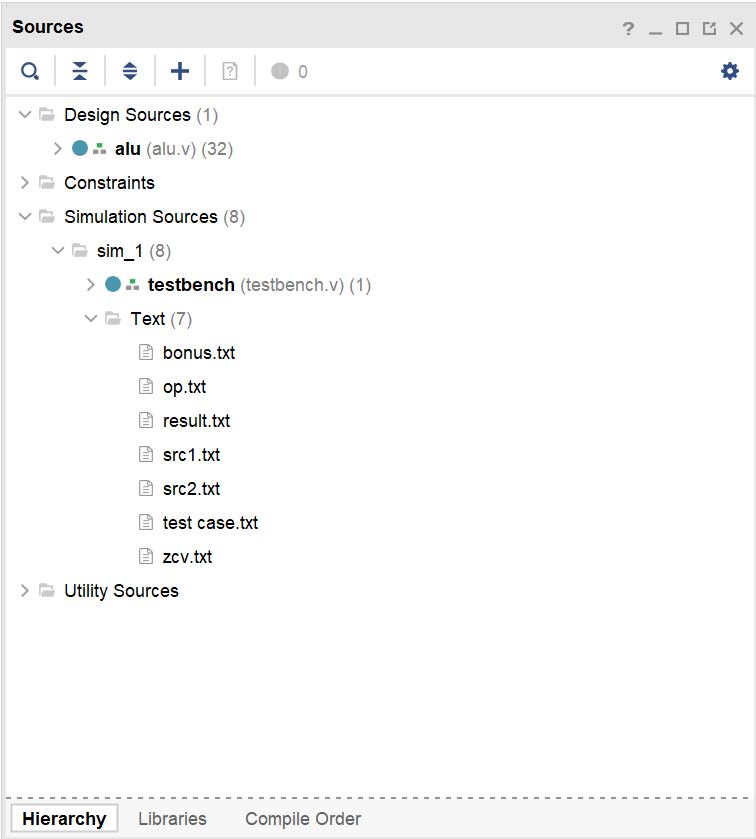
NOR:

Currect result: 00000000 Currect ZCV: 000

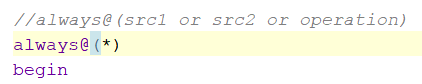
My result: 00000000 My ZCV: 000

**Problems you met and solutions:**

1.一開始我跑模擬時，讀不到測資。後來發現除了要把檔案放在同一個資料夾內，還需要加進project裡



2.我的加法器Cout總是算錯。為此我特地把alu\_top的src1,src2,cin,cout的波形都列出來。明明src1,src2,cin都對，cout就是會錯。我花三天debug，最後終於發現一開始的always block trigger只有src1,src2, operation，沒有cin。所以cin變化不會啟動always block。我把trigger改成 \* 就成功了

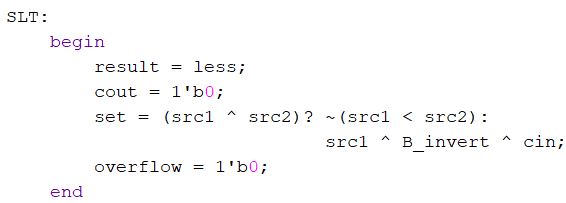


3.接著是我的減法器出錯。我是採用-src2 = + (2’complement of src2)的概念。但結果出來會差1。後來發現我只有將src2 invert，忘了加1。所以我將減法器的cin0預設為1，問題就解決了



4.起初的架構是用32個alu\_top。但我發現當我在算slt時，我發現在最後一個運算單元，不能同時獲得set和less。所以最後一個運算單元，我另外寫成alu\_bottom

5.當我在驗證答案時，發現slt涉及減法器，所以有overflow的問題。因此我有加入判斷



當src1和src2的signbit不一樣時，相減可能會overflow，所以我直接做判斷。反之兩者一樣時，就必須用減法做判斷(不會有overflow的風險)

**Summary:**

這個實驗目標是完成一個32bit的ALU。這個ALU能執行and,or,add,sub,slt,nor等指令。實驗目的是讓人更理解ALU的架構，以及ALU能在之後的實驗使用。過程中我遇到5個問題，分別是讀取測資、進位、互補數轉換、alu\_bottom和slt的相關問題。最後順利通過助教給的測資