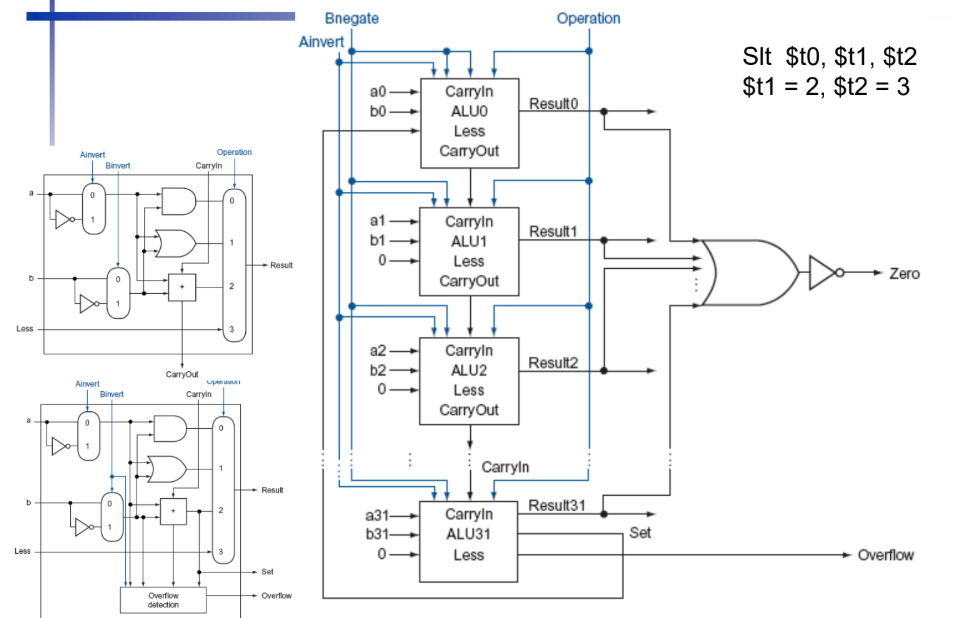
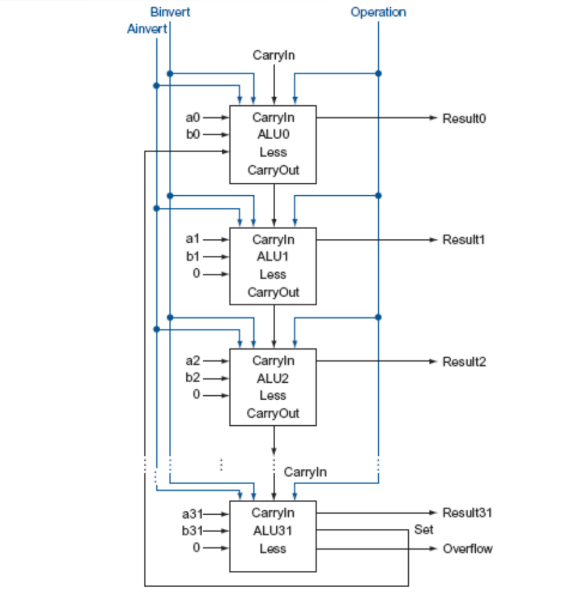
**Computer Organization**

**Architecture diagrams:**



**Hardware module analysis:**



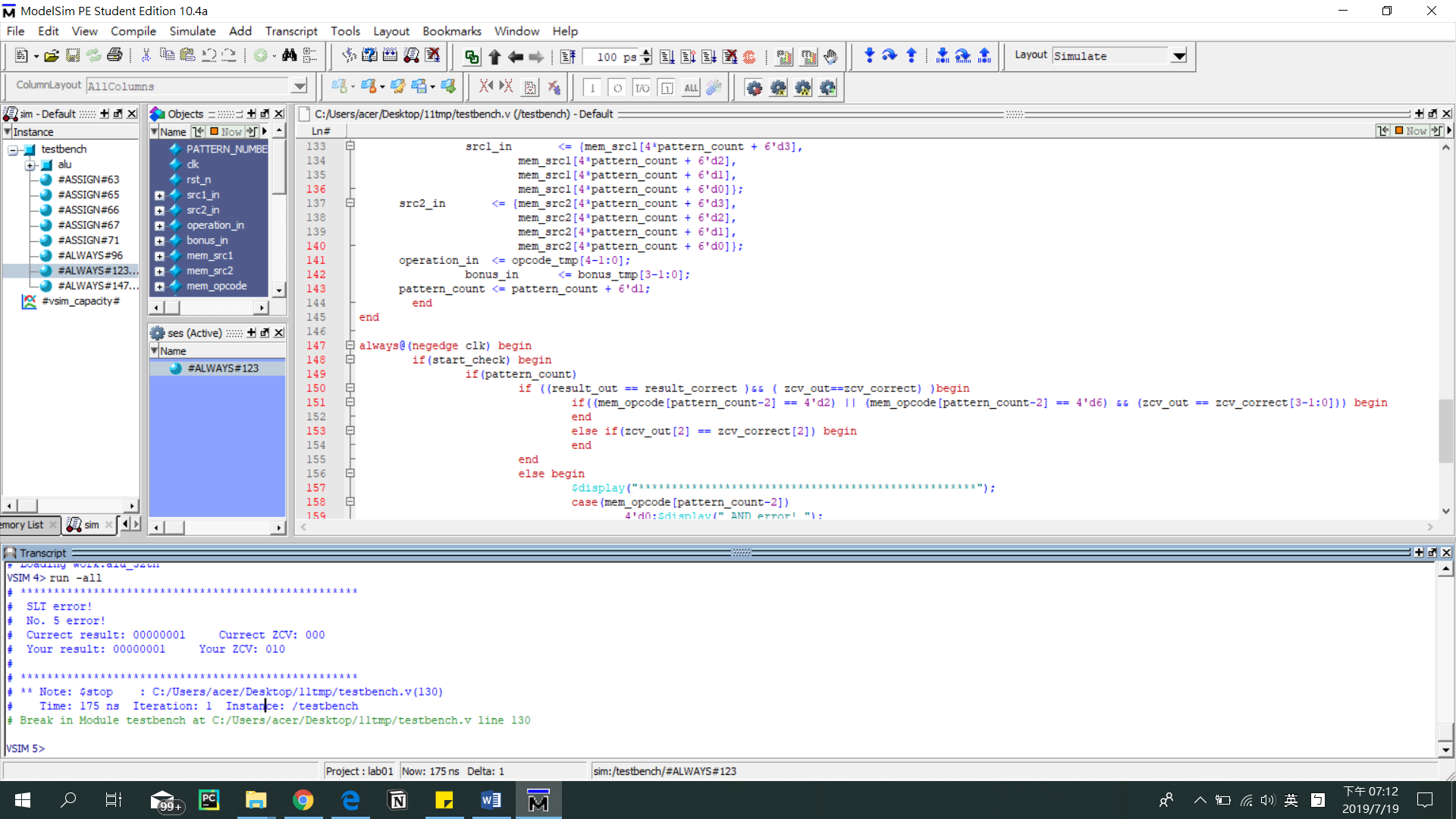
**alu\_top & alu\_32th：**

1. 透過A\_invert跟B\_invert來做src1、src2以及ALU control的判斷
2. 進入always中，先判斷要做的operation為何，再進一步去assign result的值

**alu**

1. 呼叫32次module來實現32 bit alu
2. 在always中將上面產生的結果(result/less/overflow)輸出成output

**Experiment result:**



**Problems you met and solutions:**

1. **在將第32個ALU傳回第1個ALU時發生錯誤**

Sol:將module alu\_top中的less設為input與將alu\_32th中的less設為output

1. **Overflow值錯誤**

Sol:把overflow跟carry\_out搞混。將overflow修正為第32個ALU的cin與cout的XOR。

1. **Cout的傳值有誤**

Sol:將And與Or與Nor的cout都設為0，其餘改變原本在always外去assign cout的方法，而是去判斷op code後再去做cout的給值。

1. **最後產生的result有誤**

Sol:將原本alu.v中always括號內再加入cin

1. **Slt原本完全沒有對result做處理，因此出來的值會錯誤**

Sol:更改為用less的值去做判斷，若less已被更改為1，則module中1bit的resulr輸出為1，其餘情形輸出為0。

**Summary:**

之前沒有接觸過verilog，因此一開始連架構都還搞不清楚該怎麼寫好。再更詳細了解ALU參數輸入輸出及運作原理以後，藉由摸索modele的模樣(上網google)才慢慢將ALU的雛型拼湊出來，雖然還是遇到了很多不知該從何下手的問題，總歸這次lab算是讓我們熟悉verilog的寫法以及更清楚ALU的module長什麼樣子。