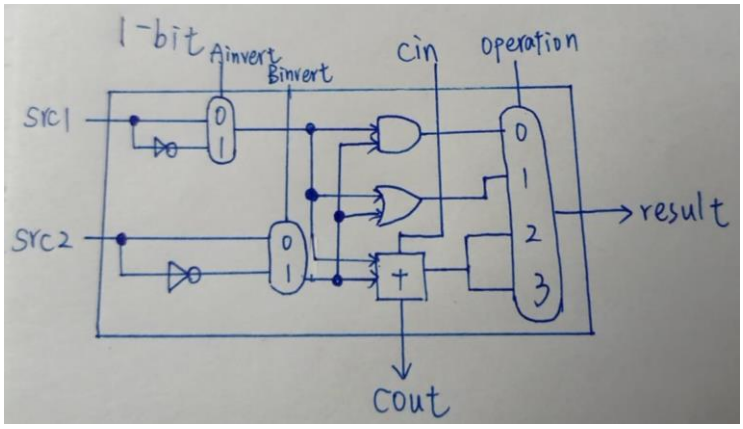
Computer Organization

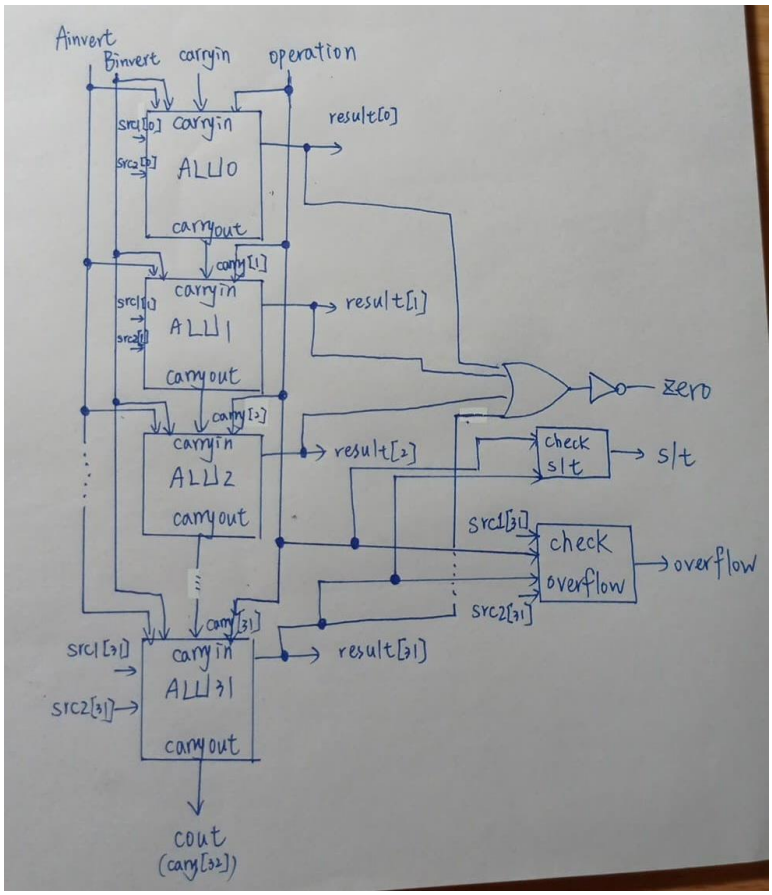
Lab2- 32-bit ALU

Architecture diagram:

1. 1-bit alu



2. 32-bit alu

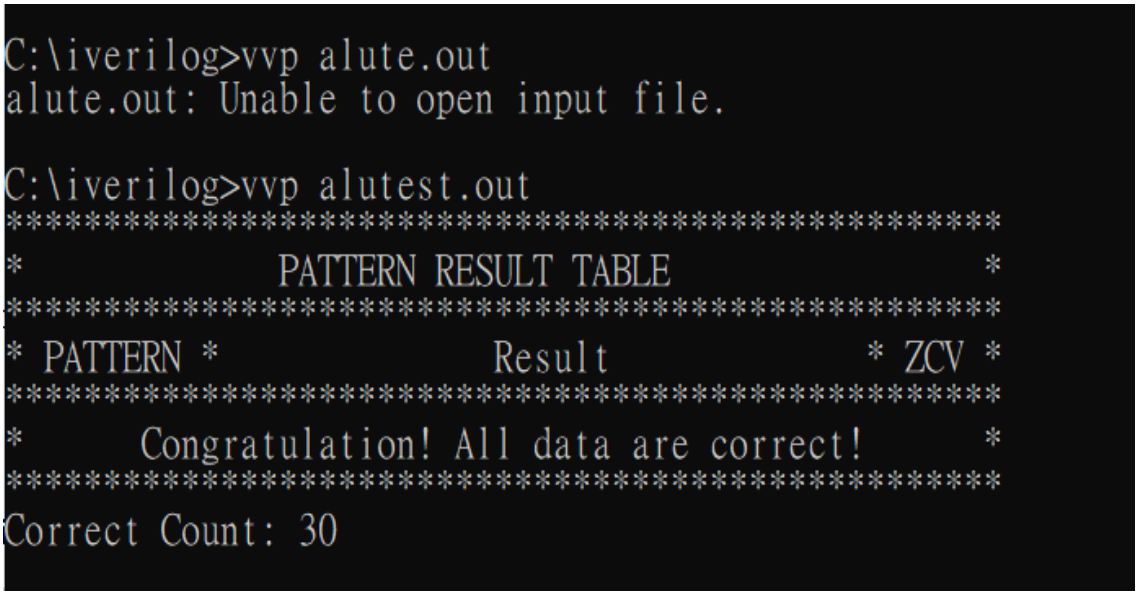


Detailed description of the implementation:

1. 先實作 **1 bit** 的 **alu**，先讓 **src1** 與 **src2** 透過 **Ainvert** 與 **Binvert** 看是否轉換，最後再判定是什麼樣的 **operation**。
2. 在 **alu.v** 的地方，先判斷 **operation** 是什麼樣的值，用了兩個新變數 **a_in**、**b_in** 作為等等宣告 **module** 的參數，但也可以直接在呼叫 **module** 的時候放進去，無需做這個判斷。
3. 因為 **cout** 也有 **32bit**，但是一開始呼叫是從 **1** 開始做，因此設定 **wire[32:0] cout**。
4. 將 **1 bit** 的 **alu** 串起來，讓上一步驟的 **cout** 成為下一步驟的 **cin**。

Implementation results:

1. 輸出結果若正確就不會有錯誤測資顯示在 **terminal** 上，以下是成功畫面



Problems encountered and solutions:

1. 在呼叫 **module** 的時候，不能直接給 **reg[0]**，像這樣一個位置的 **parameter** 進去，所以先宣告一個 **wire** 來解決這個問題，但若是參數是一整個 **reg** 的話便可以。
2. 一開始我先宣告 **wire**，並將其變數名稱設置的與 **output** 一樣，希望透過 **assign** 的方式來達到改變型態為 **reg** 的參數，但這個方式會出現 **duplicate declaration**，所以最後還是用 **always@**的方式來改 **output** 為 **reg** 型態的參數。
3. 因為 **carry in** 取決於上一個 **alu** 做出來的 **carry out**，但是第一個 **alu** 沒有之前運算完的 **carry out**，所以需要一開始要先自行設定。
4. 一開始非常不了解 **slt** 輸出的結果是甚麼意思，一直以為是比較成立輸出的答案，最後才懂原來是成立時就將結果設成 **1**，否則為 **0**。

Lesson learnt (if any):

1. **Alu** 的運作方式，以及那些情況下要考慮 **cin**。
2. **Verilog** 呼叫 **module** 的方法，以及給定參數的寫法。
3. **Overflow** 的情況。

Comment:

在實作 **slt** 時通常會有一個 **less** 去實踐，希望下次會以 **less** 去實作