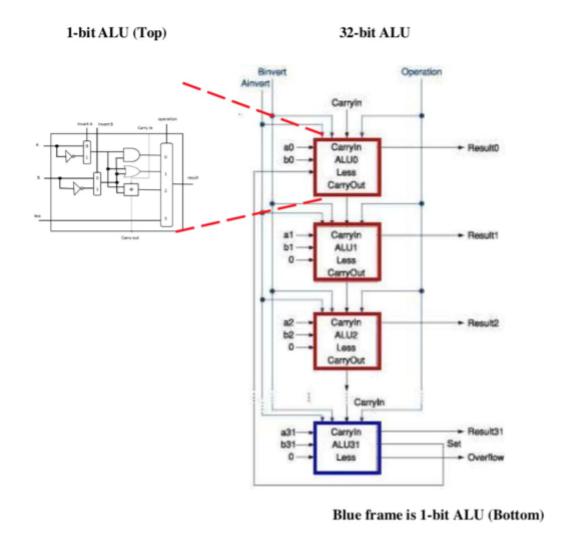
**Computer Organization**

**Architecture diagrams:**

****

如上圖,但最後一個ALU(ALU\_1bit\_last)會多出一個運算元將set的值傳回第一個ALU,重新運算一次;以及輸出一個overflow的值。

**Hardware module analysis:**

每當always@( posedge clk or negedge rst\_n )時：

result = re;

zero = (re)? 0:1;

cout = co[32]&&ALU\_control[1]&&(!ALU\_control[0]);

overflow = over;zero = (re)? 0:1;

此段程式碼會輸出答案,並且更新result值

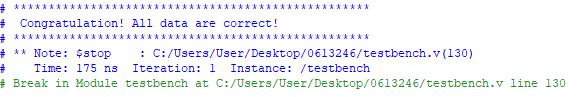
而

zero = (re)? 0:1 這段程式碼會偵測re是不是0接著把值回傳到zero

cout = co[32] &&ALU\_control[1]&&(!ALU\_control[0]);

overflow = over; 這兩段程式碼會更新carryout、overflow

**Experiment result:**

****

**Problems you met and solutions:**

其實我是轉系來資工系的,現在大二,但大一的課很多都是現在才再修,所以程式底子沒有很好,verilog也是第一次接觸,因此這次的作業我寫的蠻痛苦的QQ,很多地方都是不斷地問同學要如何寫,不過在寫cpu內部運算邏輯單位(alu)的過程中,我體會到了cpu內部複雜的運作方式。

**Summary:**

希望在下一次的lab以前我能更經濟我的verilog能力!