Computer Organization Lab1

0816146 韋詠祥

Architecture diagrams:

先設計 1-bit ALU（alu_top.v）作為基底，支援 AND、OR、ADD、SLT 操作

主程式 32-bit ALU（alu.v）再使用 32 個 1-bit ALU 模組，整合功能並延伸出 SUB 及 NOR 操作

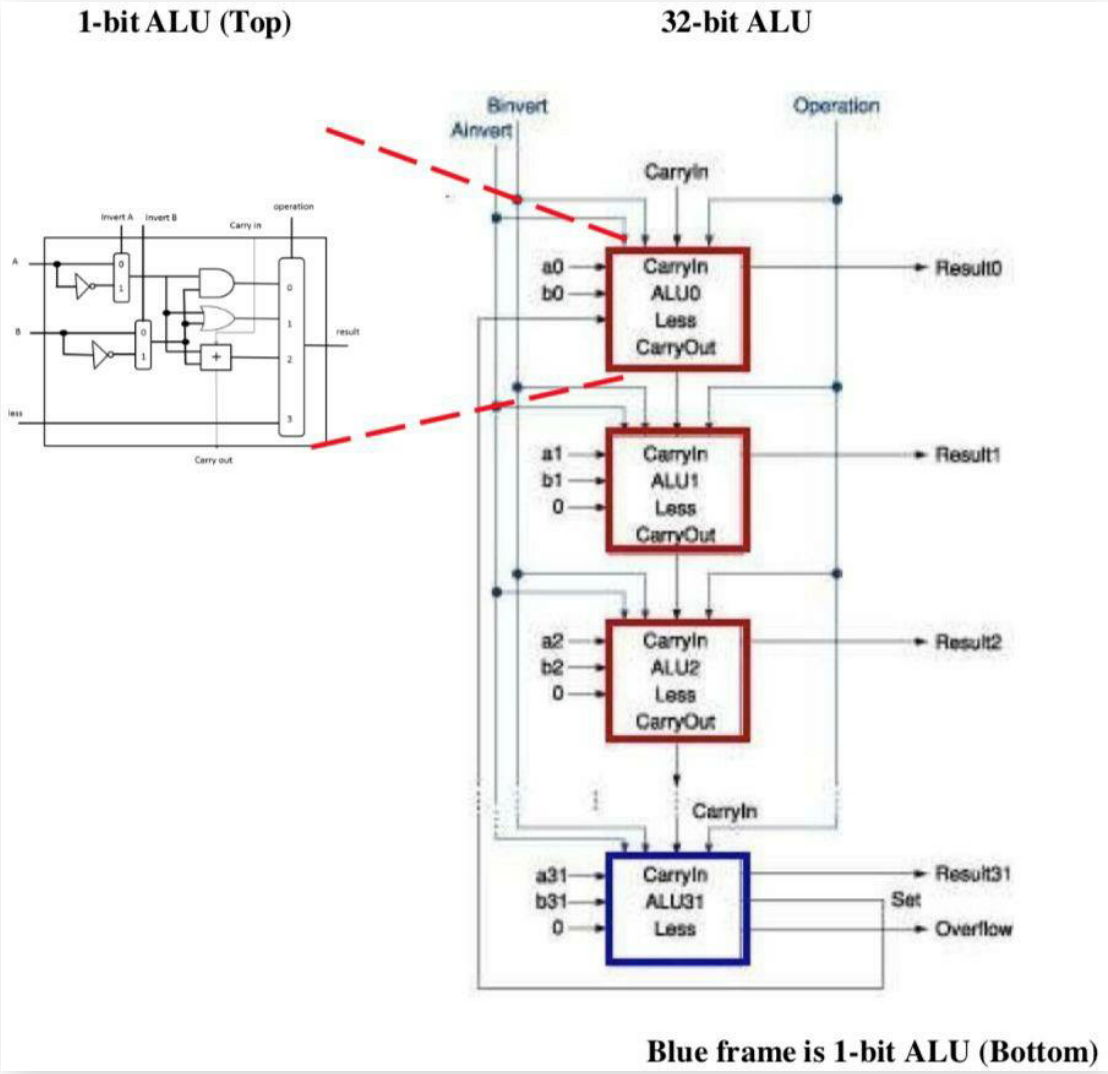

Fig 1. 架構圖，擷取自作業規格

Hardware module analysis:

用到了 1 個 alu( ) 模組、32 個 alu_top( ) 模組、
32 個 fa( ) 模組、64 個 ha( ) 模組

Experiment result:

不論是使用助教提供的測資或是自己撰寫的 gen.py 產生的測資，
testbench( ) 輸出都是「Congratulation! All data are correct!」正確訊息

```
bash-5.1$ make
iverilog -o alu ha.v fa.v alu_top.v alu.v testbench.v
python3 gen.py
./alu
VCD info: dumpfile debug.vcd opened for output.
****************************************************
 Congratulation! All data are correct!
****************************************************
```

Fig 2. 實驗結果

Problems you met and solutions:

1. 不清楚 carry bit 跟 overflow bit 差異

一開始被兩種 flag 混淆，經過搜尋後才想起來 carry bit 是從 unsigned 無號數的角度來看、overflow bit 則關係到 signed 有號數的運算

2. 不肯定自己是否有正確實作

因為怕改了 A 造成 B 壞掉，自己寫了產生測資及答案的 Python 腳本，讓我在撰寫 Verilog 模組時可以更有信心的修改原先正確的程式碼

3. 時間未對齊

一開始不管是 always @(posedge clk) 或是 always @(*) 的結果都出現錯誤，經過幾次嘗試後才終於找到哪些 always 區塊要被哪些事件觸發

Summary:

經過了這次的 Lab，我更了解 ALU 運作的原理，也藉由實作過程中釐清了自己對 zcv 三個 flag 的疑惑

這次雖然有自己寫測試用腳本，但仍然缺乏對於邊際測資的覆蓋，是未來可以考慮改進的地方