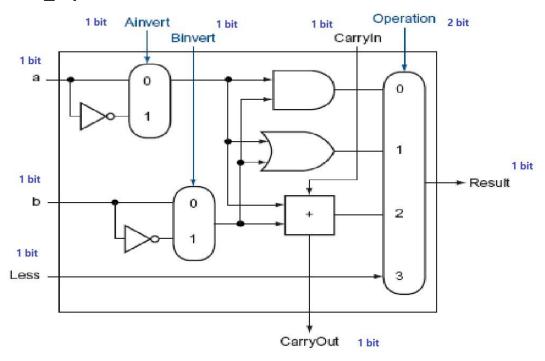
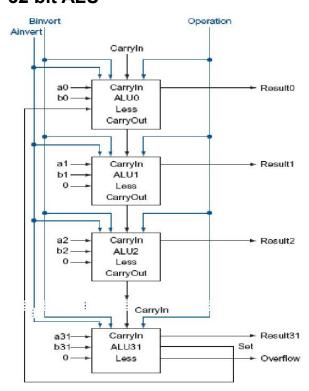
# Computer Organization\_0516244

## **Architecture diagrams:**

### ALU\_top



#### 32 bit ALU



### Hardware module analysis:

#### ALU\_top 的設計

負責處理 1bit 的 input,使用 module 如下

1) 2個 2-1 MUX

- A\_invert: 判斷 src1 的 input 是否需要 inverse

- B\_invert: 判斷 src2 的 input 是否需要 inverse

2) 1個 4-1 MUX

- Operation: 針對 operation code 的值執行對應的操作

當 operation code = 00, 把 src1 及 src2 使用 AND gate 接起,獲得 result

當 operation code = 01, 把 src1 及 src2 使用 OR gate 接起, 獲得 result

當 operation code = 10, 把 src1, src2 及 cin 使用 Full Adder 接起, 獲

得 result

當 operation code = 11, 把 less 值直接接去 result, 獲得 result

#### ALU 32 bit 的設計

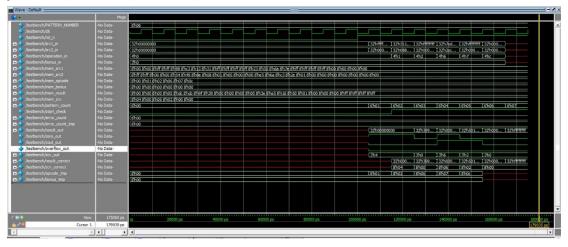
	ALU Control lines		
Function	Ainvert	Binvert	Operation
and	0	0	00
or	0	0	01
add	0	0	10
subtract	0	1	10
slt	0	1	11
nor	1	1	00

- 1) 通過把 32 個 ALU\_top 接起來實現 32 bit 的 ALU
- 2) 關於 **ZCV** 三個 **flag** 的設計

- zero: 若 result = 0, zero = 1, 反之 zero = 0;
- cout: 若 carry\_out = 1, cout = 1, 反之 cout = 0;
- overflow: 若 msb 的 cin 和 cout 不一樣, overflow = 1, 反之 overflow = 0
- 3) 最後一個 alu\_top 的 set 會被拉到第一個 alu\_top 的 less input 中
- 4) 比較需要注意的是第一個 alu\_top module,也就是負責 0 bit 的 alu\_top, 他的 carry in input 是由 ALU\_operation 來賦值,而且他的 less input 是由最後一個 alu\_top 的 output 拉回來的,所以需特別處理。
- 5) 剩餘的 31 個 alu\_top 便可通過 generate for loop 來產生

### **Experiment result:**

```
# Reading D:/Modeltech_pe_edu_10.4a/tcl/vsim/pref.tcl
# Loading project sfdfsddf
# Compile of testbench.v was successful.
# Compile of alu.v was successful.
# Compile of alu_top.v was successful.
# 3 compiles, 0 failed with no errors.
```



```
VSIM 7> run -all
# ************************
# Congratulation! All data are correct!
# *****************
# ** Note: $stop : D:/ashl3/Downloads/CO_LABI/testbench.v(131)
# Time: 175 ns Iteration: 1 Instance: /testbench
# Break in Module testbench at D:/ashl3/Downloads/CO_LABI/testbench.v line 131
```

## **Problems you met and solutions:**

一開始看不明白 Architecture Diagram,一直向助教詢問關於 ALU 的運作,上網看教學影片,花了很多時間才明白

## **Summary:**

很感動,寫出來了,建議可以給多點邏輯上的提示