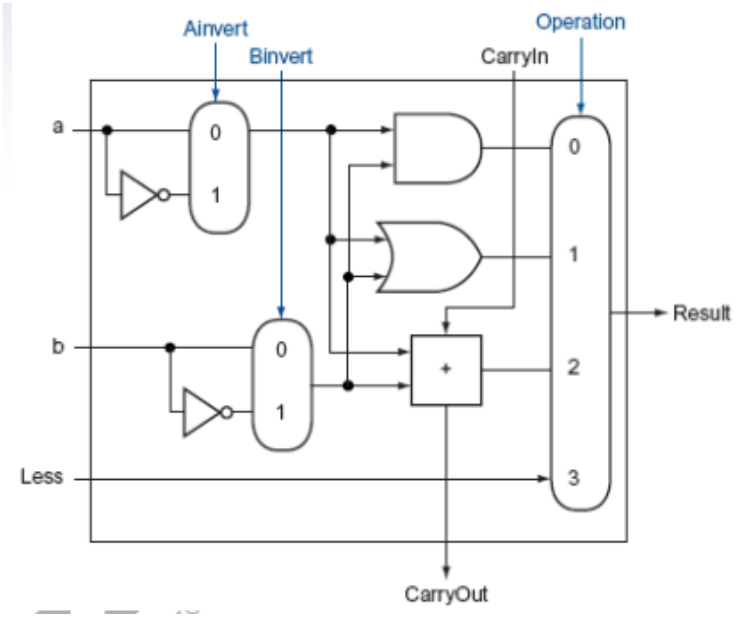
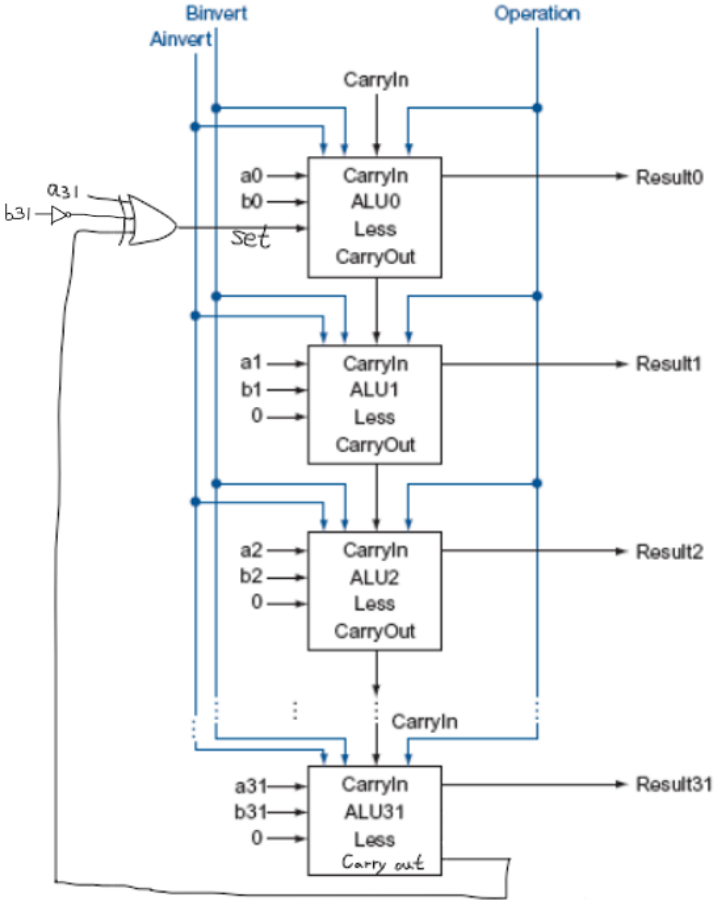
Computer Organization

Architecture diagrams:

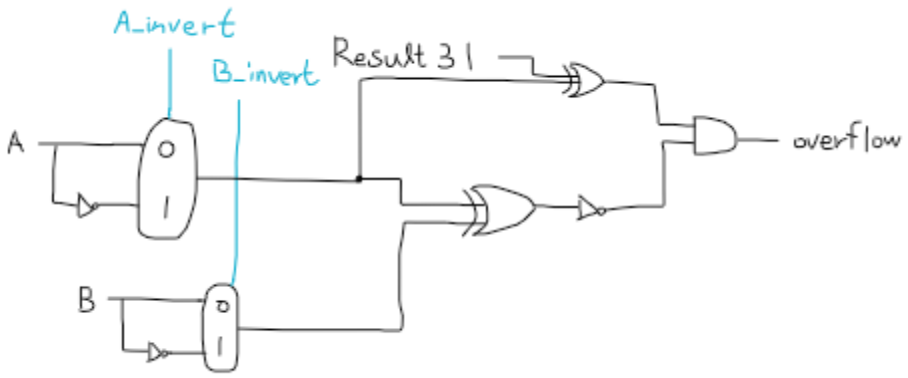
ALU_top:



32-bit ALU:



overflow:



Hardware module analysis:

ALU_top:

處理以一個 bit 為單位的資料。

因為輸入的 A 跟 B 有可能要做 inverse(sub、nor 跟 slt)，以 s1,s2 紀錄實際要運算的值(要 inverse 的 inverse，不用的就紀錄原本的 A/B)。

依據輸入的 operation 的不同，做不一樣的運算:

operation	result
00	s1 AND s2
01	s1 OR s2
10	s1 ADD s2
11	SLT

32-bit ALU:

組合了 32 個 ALU_top，可以進行 32 bits 的數值的運算。

這裡使用了 generate 的語法來呼叫全部 32 個 ALU_top。

依據輸入的 ALU_control 做運算:

Function	A_invert	B_invert	Operation
and	0	0	00
or	0	0	01
add	0	0	10
sub	0	1	10
nor	1	1	00
slt	0	1	11

其中,SLT 的 least-significant bit 是由第 32 個 ALU_top 的 carry out XOR A 的 sign bit XOR !(B 的 sign bit)。這個部分的實作是將上述 XOR 完的值拉回第一個 ALU_top 的 less(請參考 Architecture diagram 的圖)。

而第一個 ALU_top 的 carry in 取決於要做怎樣的運算:在 sub 跟 slt 時是 1，其他運算是 0，因為相減時用到的 1 補數和原數的負值差一。

最後是 ZCV 的計算:

zero: 當計算結果(result)的每一個 bit 皆為 0,zero 為 1,反之為 0。

cout: 即第 32 個 ALU_top 的 carry out。

overflow: 即實際運算的兩數同號但和結果不同號。

計算過程:

A = ALU_control[3] ? !src1[31]: src1[31];

B = ALU_control[2] ? !src2[31]: src2[31];

overflow = ((~(A^B))) & (A^result[31]);

(實際運算的兩數，ex. A - B，則實際運算的兩數為 A 和(-B))

Experiment result:

從 100ns 開始截圖(因為 rst_n 從 100ns 開始不為 0,ALU 開始運作)


```
# run 1000ns
**************************************************
 Congratulation! All data are correct!
**************************************************
$stop called at time : 175 ns : File "C:/Users/warm/Desktop/computer_organization/Lab_1/CO_LAB1/testbench.v" Line 104
xsim: Time (s): cpu = 00:00:03 ; elapsed = 00:00:08 . Memory (MB): peak = 1274.992 ; gain = 1.422
INFO: [USF-XSim-96] XSim completed. Design snapshot 'testbench_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
launch_simulation: Time (s): cpu = 00:00:05 ; elapsed = 00:00:12 . Memory (MB): peak = 1274.992 ; gain = 1.422
```

Problems you met and solutions:

```
reg [2-1:0] operation;
reg [32-1:0] result;
reg less, overflow, zero, cout;
reg A, B;
wire set;
wire [32-1:0] carry;
wire [32-1:0] result_tem;
```

Problem : 一開始不知道像[32-1:0] result 這樣的宣告方式是把資料倒過來讀取的，導致判斷 operation 那邊一直有錯誤。

Solution : 發現之後趕快改過來就好了，太感動了!

Summary:

實作 ALU 給我一種很奇妙的感覺(雖然這可能是簡化又簡化版的 ALU)，原來我也可以設計出一個頗實用的原件，真好，如果能更早發現上面敘述那個問題就更好了。