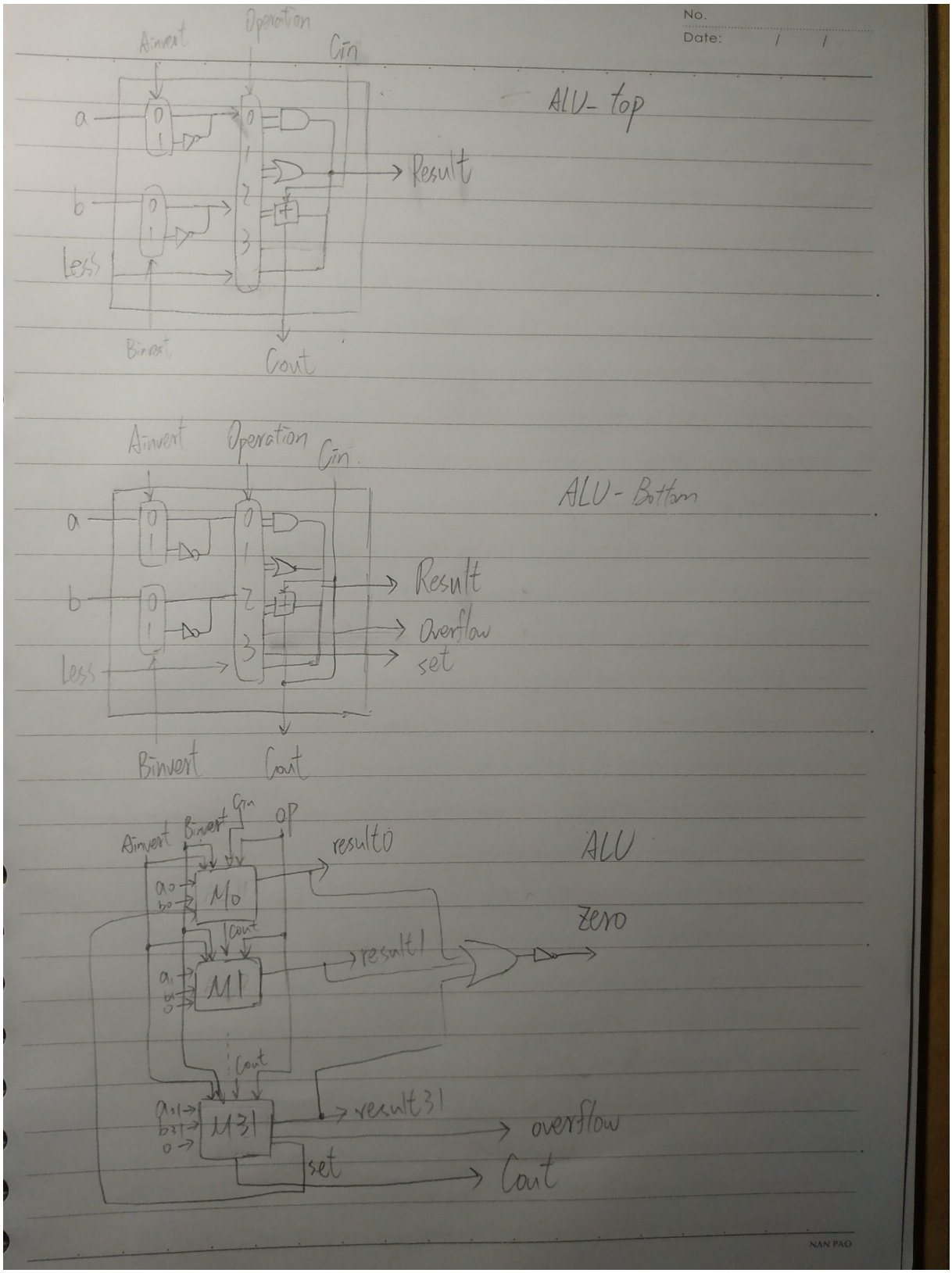
0616223 劉柏宇　　計算機組織 Lab1 報告

1.



(M0~M30 是 alu_top, M31 是 alu_bottom)

2. 用 case 的方式去選擇賦值, 當 rst_n = 1, 開始跑 接收 ALU_control 當作判斷輸出結果的選擇器

AND: 對 A,B 兩個值做&, 輸出

OR: 對 A,B 兩個值做 | , 輸出

ADD: 對 A,B 每個 bit 做加法, 包含 carry-in = 0, carry-out

SUB: 對 A,~B 每個 bit 做家法, 包含 carry-in = 1, carry-out

NOR: A NOR B = ~A & ~B

NAND: A NAND B = ~A | ~B

SLT: 做減法後判斷 sign bit, 若是 1 則代表 less than, result[0] = 1, 若不是則為 0, 其餘位數接輸出 0

Zero: 所有 result OR 後再 NOT, 輸出

Cout: 當 M31 計算完後有超出 32bit 的位數, 則 Cout 紀錄為 1

Overflow: 當正加正,負加負,結果 sign bit 卻不一樣時, overflow = 1, 我是用最後一個 bit 的 cin cout 來判斷, 若 cin cout 不相等, 則會出現 overflow

3.

iverilog -o alu.vvp testbench.v alu.v alu_top.v alu_bottom.v

4.

遇到最大的問題是在 wire 跟 reg 的宣告,在經過上網查以及詢問大家後,知道了 reg 在 always 裡面才可變動, wire 在 always 外面才可以變動

然後是 module 要用什麼去接 output 也是看了很多範例才釐清

而最大的問題是我發現一般的 1bit ALU 應該是都會跑各種 gate, 但我先用 operator 控制賦值, 導致我的輸出是根據要的東西變得,譬如跑 ADD 我就輸入 2'b10 就會跑到加法器那邊, and or 就沒有值跑出來, 跑 SLT 時候就出問題了, 因為要先做減法, 所以要先輸入 1'b0, 1'b1, cin = 1, 2'b10, 但跑完之後就沒辦法跑 2'b11 輸出各種 less 的結果, 因此我採取了輸入 2'b11, 也會跑減法, 但差別在於只跑進位, 減完的結果變成 less 輸出, 最後用 sign bit 在傳回第一個 result[0] 當輸出 (雖然我不知道這樣有沒有問題, 但我覺得我的作法好糟…, 也因此我畫的 diagram 跟給定的不一樣, 原因就是我先跑了 selector, 才跑 gate..)

5.

學到最多的當然是 verilog 的寫法跟 alu 的運作原理, 但意外的收穫也有, 就是我以後會先想清楚在開始刻電路, 這次就是因為沒太多時間想所以沒注意到我心裡規劃的跟實際給的長的有出入...