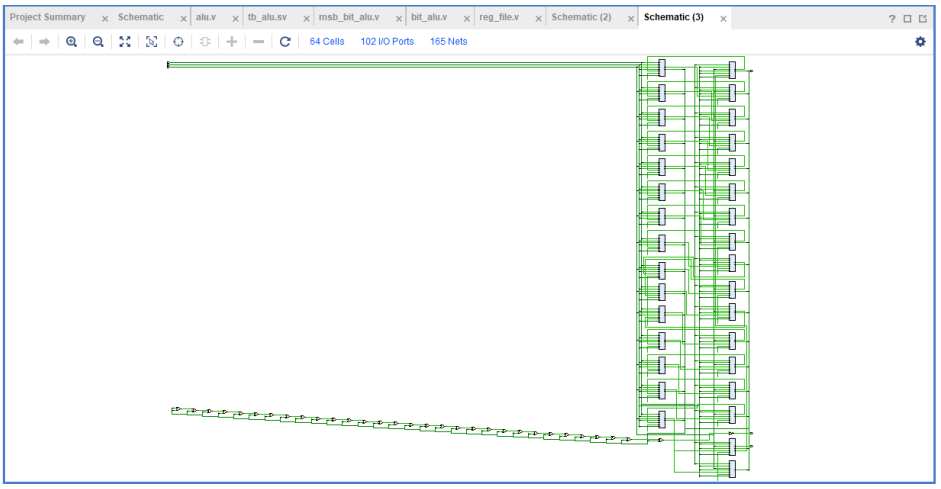
Lab 1: 32-bit ALU & Resgister File

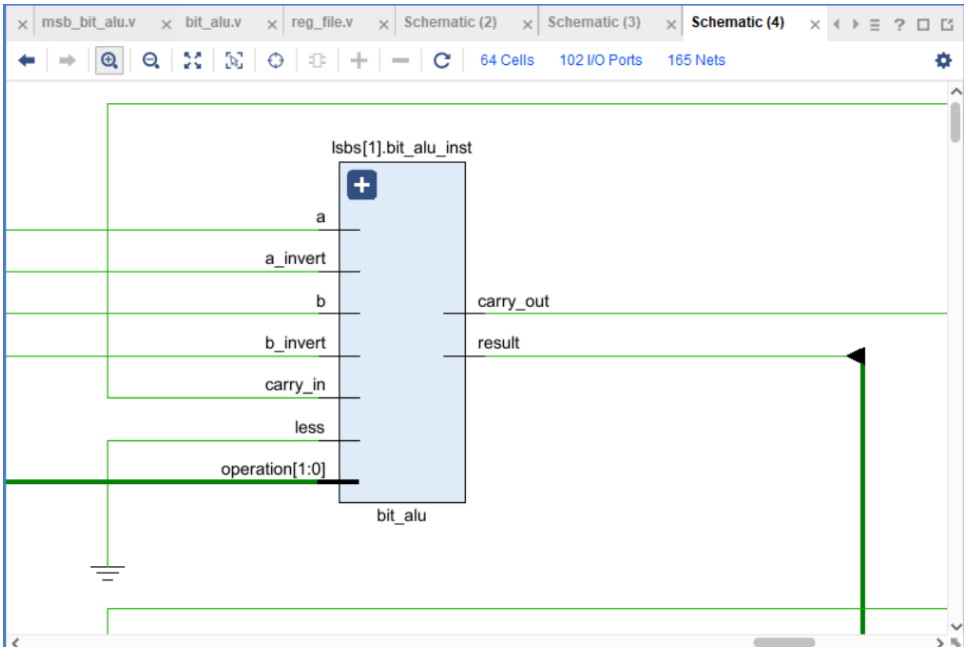
學號：111550129 姓名：林彥亨

1. Architecture Diagrams
    a. Show your 1-bit & 32-bit ALU design by “Schematic” tool in vivado.

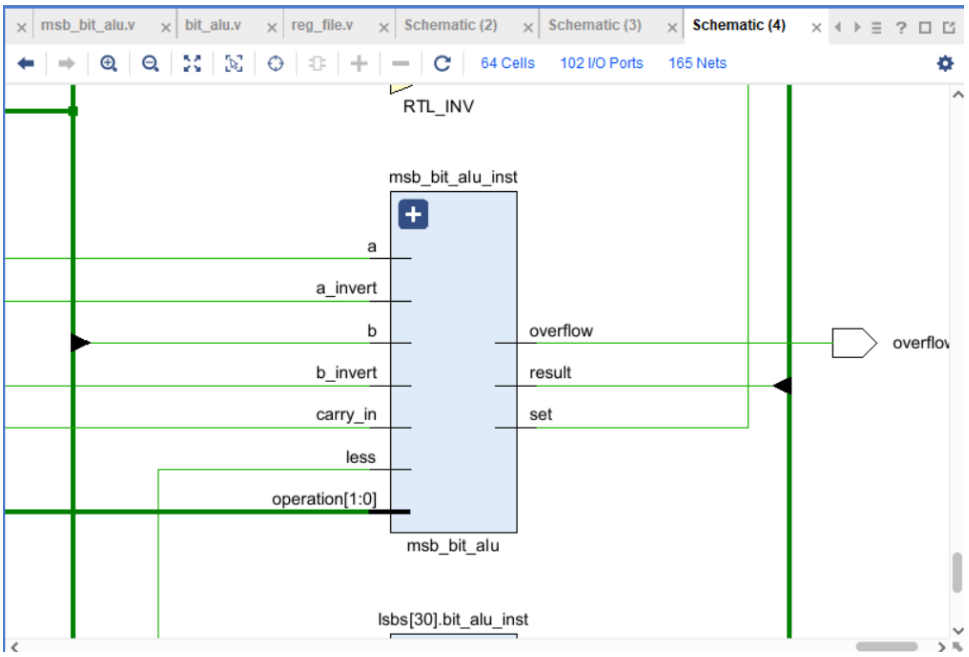
- This is the 32-bits ALU



- This is the bit_ALU



- This is the msb_bit_ALU



2. Answer the following Questions

   a. How overflow is calculated ?

      - 當最高位元的 carry out 是 1，最低位元的 carry in 是 0；或是最高位元的 carry out 是 0，最低位元的 carry in 是 1，就會發生 overflow。

   b. Explain why ALU control signal of SUB is 0110 and NOR is 1100 ?

      因為我們將operation bit 設定為第零和一位，Binvert 為第二位，Ainvert 為第三位

      - SUB: 在SUB裡會用到ADD所以在operation那2-bit 是 '10'，又因為是SUB會將Binvert 變成 '1'，Ainvert 為 '0'，所以才是0110。
      - NOR: NOR 可以將其分解成A' & B'。所以 operation 為 and "00"，又 Ainvert 和 Binvert 都為 '1'，所以NOR 為1100。

   c. (2 cont'd) If you assign different siganl to these operation, what problems you may encountered ?

      我可能遇到的問題：

      - Logical error
      - incorrect results
      - increasing complexity

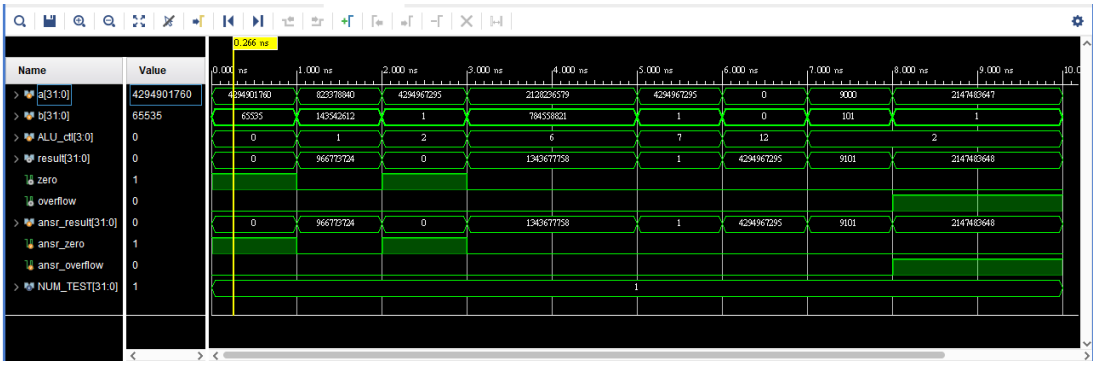
d. True or false: Because use the register file is both read and written on the same clock cycle, any MIPS datapath using edge-triggered writes must have more than one copy of the register file. Explain your answer.

   This statement is **False**. In MIPS architecture, and in many other similar architectures, the register file is designed to allow reads and writes to occur within the same clock cycle, but they do not necessitate multiple copies of the register file

3. Experimental Result

   a. Show the waveform screen shot of the testbech “tb_alu.0.txt” result.

      - The waveform of the tb_alu.0.txt.



   b. What other cases you’ ve tested? Why you choose them?

```
1   0xffff0000 0x0000ffff AND 0x00000000 1 0
2   0x3113c398 0x088e4954  OR 0x399fcbdc 0 0
3          -1          1 ADD          0 1 0
4   0x7eda5023 0x2ec36ae5 SUB 0x5016e53e 0 0
5   0x7eda5023 0x2ec36ae5 SUB 0x5016e53e
6          -1          1 SLT          1 0 0
7   0x00000000 0x00000000 NOR 0xffffffff 0 0
8   0x00002328 0x00000065 ADD 0x0000238d 0 0
9   0x7FFFFFFF 0x00000001 ADD 0x80000000 0 1
```

```
# run 1000ns
#### tb_alu.sv ####
==== Test  0 RUNNING ====
ERROR: No Format provided for this argument
[case  0] 0xffff0000 0000 0x0000ffff
[case  1] 0x3113c398 0001 0x088e4954
[case  2] 0xffffffff 0010 0x00000001
[case  3] 0x7eda5023 0110 0x2ec36ae5
[case  4] 0x7eda5023 0110 0x2ec36ae5 (ignore zv)
[case  5] 0xffffffff 0111 0x00000001
[case  6] 0x00000000 1100 0x00000000
[case  7] 0x00002328 0010 0x00000065
[case  8] 0x7fffffff 0010 0x00000001
==== Test  0 PASSED ====
#### Test Result ####
Passed  1 : 0
Failed  0 :
#### all passed!
```

the case added is in line 8 and line 9.

line 8: “2328 add 65” → the result is “238d” in hexidecimal. I want to test the add function.

line 9: “7FFFFFFF add 00000001” → the result is “80000000” in hexidecimal. I want to test the overflow function.

4. Problems Encountered & Solution
   a. List some important problem you’ ve met during this lab and these solution.
      i. carry in 和 carry out 的連接：

```
43      // Instantiate the 31 less significant 1-bit ALUs
44      genvar i;
45      generate
46          for (i = 0; i < 31; i = i + 1) begin : lsbs
47              bit_alu bit_alu_inst (
48                  .a(a[i]),
49                  .b(b[i]),
50                  .less(less[i]),
51                  .a_invert(a_invert),
52                  .b_invert(b_invert),
53                  .carry_in(carry_in[i]),
54                  .operation(operation),
55                  .result(result[i]),
56                  .carry_out(carry_out[i])
57              );
58          end
59      endgenerate
```

起初，因為沒有將各個ALU間的carry in 和 carry out 做連接妥當，以至於結果一直跑不出來，所以我利用一個迴圈去將除了最高位元(剩餘的)ALU 做連接。

      ii. set on less than 的工作原理

我參考了網路上的些資料和課本上的描述，了解 set on less than 的運作原理，並且完成set 跟 less的連接。

```
41      // Handle the 'less' signal for SLT operation
42      assign less = {31'b0, set}; // Only the least significant bit (set) defines the SLT result
```

5. Feedback
    a. Anything you want to say to TA team about this lab. How can we improve the lab？

        這份lab可以感受到助教的用心，也有在旁邊適時的提點，我覺得這樣的作業比起毫無頭緒的作業更讓人有動力去完成和學習裡面的知識！