DCP3362 Computer Organization Lab 1

∗ Name: 石育瑋 ID: A073708 Email: stoneonetwo1203@gmail.com

1 Architecture diagrams

總共設計了 2 個 module，分別是 alu 與 alu_top。

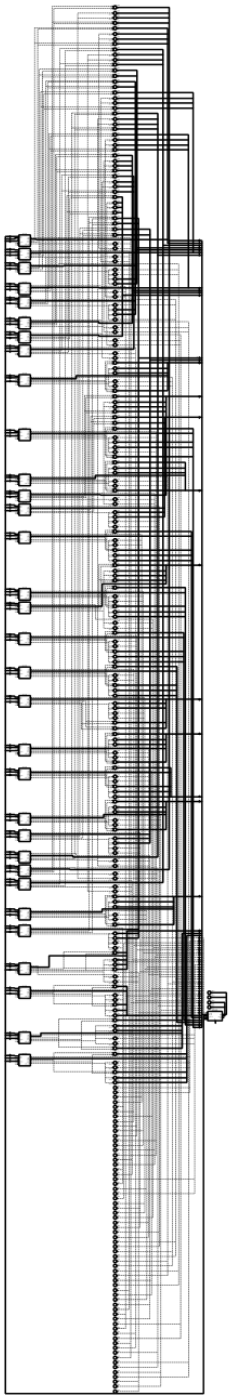

Fig. 1.1: Data flow of alu

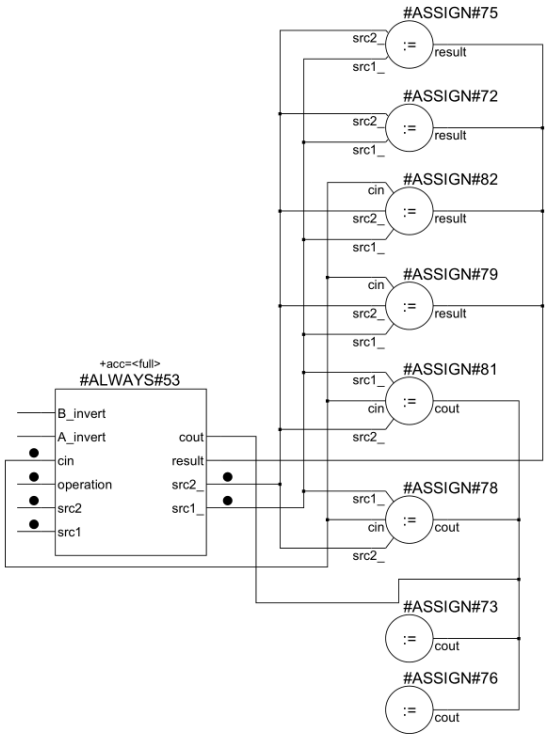

Fig. 1.2: Data flow of alu_top

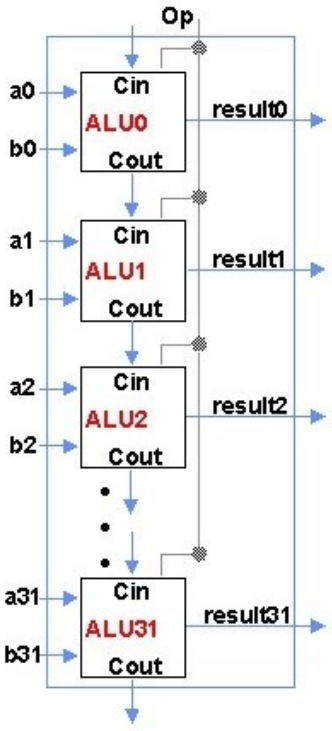

Fig. 1.3: Diagram

2 Hardware module analysis

```
module alu_top(
                src1,       //1 bit source 1 (input)
                src2,       //1 bit source 2 (input)
                less,       //1 bit less     (input)
                A_invert,   //1 bit A_invert (input)
                B_invert,   //1 bit B_invert (input)
                cin,        //1 bit carry in (input)
                operation,  //operation      (input)
                result,     //1 bit result   (output)
                cout,       //1 bit carry out(output)
                );
```

Fig. 2.1: alu_top module

在 alu_top 的設計中:

1. 2 個 2-1 MUX 控制信號:

   - (input) A_invert
   - (input) B_invert

   當 A_invert 或 B_invert 為 1 時，對 src1 或 src2 取 NOT 值輸出；而當 A_invert 或 B_invert 為 0 時，則輸出 src1 或 src2。

2. 1 個 4-1 MUX 控制信號:

   - (input) operation

   當 operation 為 00 時，進行 AND 運算。
   當 operation 為 01 時，進行 OR 運算。
   當 operation 為 10 時，進行 ADD 運算。
   當 operation 為 11 時，進行 SLT 運算，搭配 (input) less 信號。

3. 輸出運算結果

   - (output) result

4. 輸出進位

   - (output) cout

```
module alu(
            clk,            // system clock              (input)
            rst_n,          // negative reset            (input)
            src1,           // 32 bits source 1          (input)
            src2,           // 32 bits source 2          (input)
            ALU_control,    // 4 bits ALU control input  (input)
              //bonus_control, // 3 bits bonus control input(input)
            result,         // 32 bits result            (output)
            zero,           // 1 bit when the output is 0, zero must be set (output)
            cout,           // 1 bit carry out           (output)
            overflow        // 1 bit overflow            (output)
            );
```

Fig. 2.2: alu module

在 alu 的設計中:

1. 2 個 ALU 控制信號:
   - (input) clk
   - (input) rst_n

   當 clk 上沿時，才對運算結果等進行更新。
   當 rst_n 為 1 時，ALU 才會運作。

2. 2 個輸入 srouce:
   - (input) src1
   - (input) src2

3. ALU 控制信號
   - (input) ALU_control

   控制關係在表5.1中。

4. 4 個輸出結果
   - (output) result
   - (output) zero
   - (output) cout
   - (output) overflow

在 alu 中呼叫了 32 個圖2.1 alu_top，其中 alu_top 負責 AND、OR、NOT、ADD 的運算，圖2.2 alu 則是將所有的控制值 (ex: 運算的 controll、invert 等) 傳進 alu_top 以及對最後輸出的結果進行處理。

3 Experiment result

實驗透過助教提供的 Test Bench 與 ModelSim 進行模擬測試，表3.1為輸入的測試數據表格，分別對 6 種不同的運算測驗，模擬出來得到的結果在表3.2中，圖3.1可以看到最終結果正確無偏差。

Tab. 3.1: 實驗數據

operation	ALU code	src1	src2
AND	0000	ffff0000	0000ffff
OR	0001	3113c398	088e4954
ADD	0010	fffffff	00000001
SUB	0110	7eda5023	2ec36ae5
SLT	0111	fffffff	00000001
NOR	1100	00000000	00000000

Tab. 3.2: 實驗結果

result	zcv
00000000	100
399fcbdc	000
00000000	110
5016e53e	010
00000001	000
fffffff	000

```
# ****************************************************
# ****************************************************
#  Congratulation! All data are correct!
# ****************************************************
# ** Note: $stop    : C:/Modeltech_pe_edu_10.4a/examples/testbench.v(130)
#    Time: 175 ns  Iteration: 1  Instance: /testbench
```

Fig. 3.1: testbench 測試結果

4 Problems you met and solutions

在寫 LAB 時，主要有遇到以下 2 個問題:

1. 分不清楚 verilog 裡"A <= B;" 與"assign A = B;" 之間的差異，導致在寫 lab 時不知什麼情況該用哪一種。

   Solution.

   (a) assign a = b (Blocking assignment) : 執行順序不一定，

   (b) a <= b (Nonblocking assignment) : 所有可同時值行的東西都要執行完一次後，才會前進到下一個時間點。

   □

2. 在連接 alu_top 時，不知道如何生成 32 個 alu_top modules。

   Solution.

先創建初始第一個 alu_top alu_0，再利用 for 循環創建剩餘的 31 個 alu_top，最後將 less wire 連接。

```
// construct ALU from alu_top
alu_top  alu_0(
    .src1(src1[0]),
    .src2(src2[0]),
    .less(!carry_wire[31]),
    .A_invert(ALU_control[3]),
    .B_invert(ALU_control[2]),
    .cin(1'b0),
    .operation(ALU_control[1:0]),
    .result(result_wire[0]),
    .cout(carry_wire[0])
);
for(i=1; i<32; i=i+1) begin: generate_alu_top
    alu_top  alu_k(
        .src1 (src1[i]),
        .src2 (src2[i]),
        .less (1'b0),
        .A_invert (ALU_control[3]),
        .B_invert (ALU_control[2]),
        .cin (carry_wire[i-1]),
        .operation (ALU_control[1:0]),
        .result (result_wire[i]),
        .cout (carry_wire[i])
    );
end
```

Fig. 4.1: Construct ALU from alu_top

□

5 Summary

這個 Lab 是我第一次接觸 Verilog，學到很多東西，對計算機組織中 ALU 的架構也更加理解，最終實現表5.1中的 6 種指令。

Tab. 5.1: Supported Instructions

ALU Action	Name	ALU Control Input
And	And	0000
Or	Or	0001
Add	Addition	0010
Sub	Subtraction	0110
Nor	Nor	1100
Slt	Set less than	0111