國立臺灣科技大學
電機工程系

計算機組織
作業報告
HW1

四電機三乙 | B10707128 | 劉杰閎

各模組程式碼截圖並說明

一、RF.v

```
29  module RF(
30      input [4:0]Addr1,
31      input [4:0]Addr2,
32      output [31:0]Src1,
33      output [31:0]Src2
34  );
35
36      /*
37       * Declaration of inner register.
38       * CAUTION: DONT MODIFY THE NAME AND SIZE.
39       */
40      reg [31:0]R[0:31];
41
42      // Reg MUXs
43      assign Src1 = R[Addr1];
44      assign Src2 = R[Addr2];
45
46  endmodule
47
```

關於ALU的這個模組，除了一開始提供的模板以外，我做的部分就是將Addr2的部份給讀入Src2。

二、ALU.v

C: > Users > jieho > Documents > HDL > Computer_Organization > HW1 > ALU.v

```
1  `define SLL      6'b100001
2  `define AND      6'b010001
3  `define subu     6'b001010
4  `define addu     6'b001001
5  module ALU
6  (
7  input [31:0] Src1,
8  input [31:0] Src2,
9  input [4:0] Shamt,
10 input [5:0] Funct,
11 output reg [31:0] Result,
12 output Zero,
13 output reg Carry
14 );
15
16 assign Zero = !(Result);
17
18 // operation  triggered by Inputs
19 always@(Src1 or Src2 or Shamt or Funct)
20 begin
21     case (Funct)
22         `SLL:   {Carry,Result} <= (Src1 << Shamt);
23         `AND:   {Carry,Result} <= (Src1 & Src2);
24         `subu:
25             begin
26                 if(Src1 < Src2)
27                 begin
28                     Carry = 1'b1;
29                 end
30                 else begin
31                     Carry = Carry;
32                 end
33                 {Carry,Result} <= (Src1 - Src2);
34             end
35         `addu:  {Carry,Result} <= (Src1 + Src2);
36         default:{Carry,Result} <= 0;
37     endcase
38
39 end
40
41
42 endmodule
```

關於 ALU 的這個模組，除了一開始提供的模板以外，我做的部分就是將 ADD，AND，unsigned 減法的功能的實作。還有 zero flag 及 carry flag。這裡要稍微說明一下 Carry flag，我的作法是在減數比被減數大的時候就令 carry flag = 1，接著才做減法。還有就是 Zero flag 的部分，我是利用 wire 的 assign 方式來確保 Zero flag 會在任何情況下，只要 result 為 0，zero flag 就一定會跳 1。

三、CompALU.v

```
29  module CompALU(
30      //  Outputs
31      Result, Zero, Carry,
32      //  Inputs
33      Instr
34  );
35  // Ports declaration
36  input Instr;
37  output Result,Zero, Carry;
38
39  // Type declaration
40
41  wire [31:0] Instr;
42  wire [31:0] Result;
43  wire [31:0] inner_Src1;
44  wire [31:0] inner_Src2;
45  /*
46  * Declaration of Register File.
47  * CAUTION: DONT MODIFY THE NAME.
48  */
49  RF Register_File(
50      //Inputs
51      .Addr1(Instr[25:21]),
52      .Addr2(Instr[20:16]),
53      //Outputs
54      .Src1(inner_Src1),
55      .Src2(inner_Src2)
56  );
57
58  ALU Arithemetic_Logical_Unit(
59      //Inputs
60      .Src1(inner_Src1),
61      .Src2(inner_Src2),
62      .Shamt(Instr[10:6]),
63      .Funct(Instr[5:0]),
64      .Zero(Zero),
65      .Carry(Carry),
66      //Outputs
67      .Result(Result)
68  );
69
70
71  endmodule
```

而最後，關於 CompALU 這個模組，我做的則是去呼叫 RF 跟 ALU 這兩個模組，再把 Zero，Carry，Result 等等藉由 wire 把他們從內部連接起來。

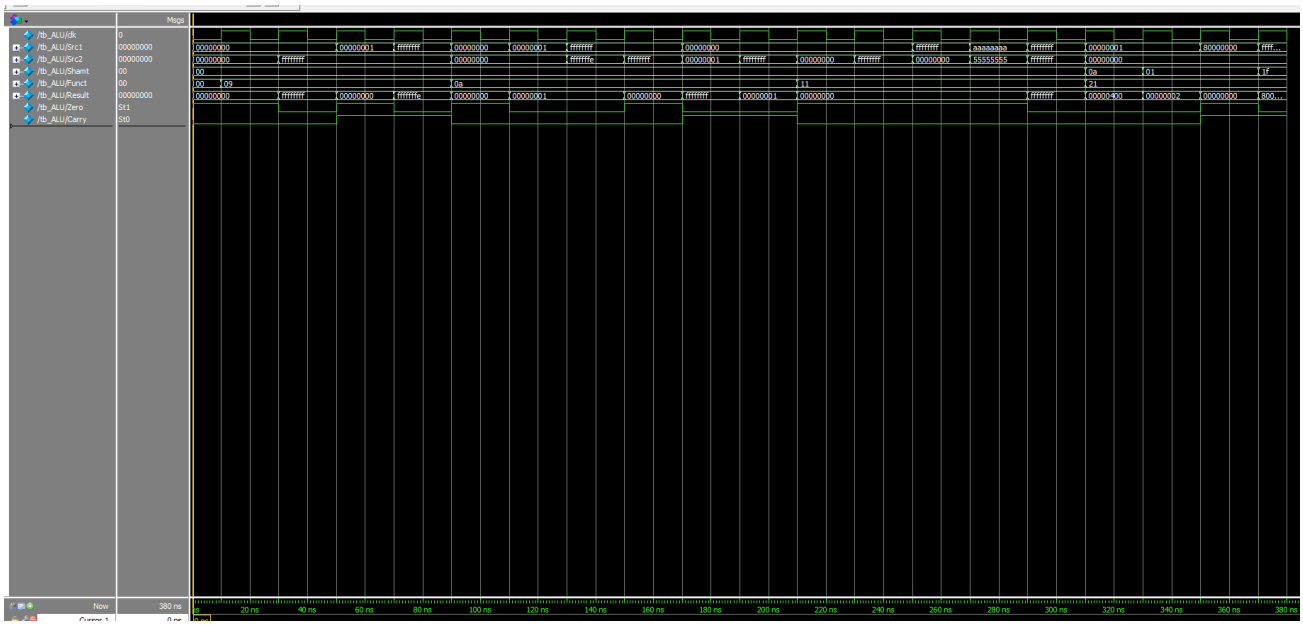
各模組測試指令（.in檔）截圖及說明

tb_ALU.in

testbench > tb_ALU.in

```
1  00000000000000000000000000000000_00000000000000000000000000000000_00000_001001
2  00000000000000000000000000000000_11111111111111111111111111111111_00000_001001
3  00000000000000000000000000000001_11111111111111111111111111111111_00000_001001
4  11111111111111111111111111111111_11111111111111111111111111111111_00000_001001
5
6  00000000000000000000000000000000_00000000000000000000000000000000_00000_001010
7  00000000000000000000000000000001_00000000000000000000000000000000_00000_001010
8  11111111111111111111111111111111_11111111111111111111111111111110_00000_001010
9  11111111111111111111111111111111_11111111111111111111111111111111_00000_001010
10 00000000000000000000000000000000_00000000000000000000000000000001_00000_001010
11 00000000000000000000000000000000_11111111111111111111111111111111_00000_001010
12
13 00000000000000000000000000000000_00000000000000000000000000000000_00000_010001
14 00000000000000000000000000000000_11111111111111111111111111111111_00000_010001
15 11111111111111111111111111111111_00000000000000000000000000000000_00000_010001
16 10101010101010101010101010101010_01010101010101010101010101010101_00000_010001
17 11111111111111111111111111111111_11111111111111111111111111111111_00000_010001
18
19 00000000000000000000000000000001_00000000000000000000000000000000_01010_100001
20 00000000000000000000000000000001_00000000000000000000000000000000_00001_100001
21 10000000000000000000000000000000_00000000000000000000000000000000_00001_100001
22 11111111111111111111111111111111_00000000000000000000000000000000_11111_100001
23
```

這個 ALU 的 in 檔，從上到下分別是 add, sub, AND, shift，由三行 enter 來區隔。在 add 的部分，我分別測試了 0+0, 0+最大, 1+最大, 及最大+最大。在 sub 的部分，我分別送了正常的減法(也就是沒有負數的部分)，及會造成負數的減法(也就是減數大於被減數)，在 and 的部分就只有做單純的測試，而在最後的 shift 我也有分別測試會造成 carry 為 1 及 zero 為 1 的部分。

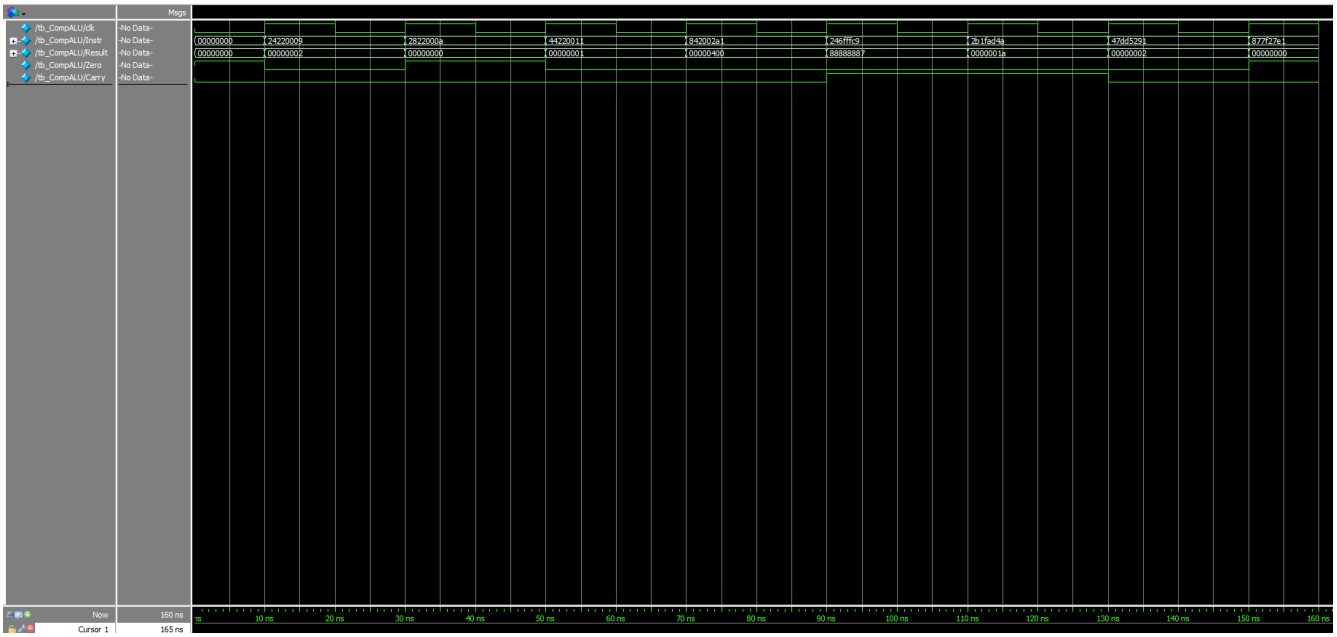


Tb_CompALU.in

testbench > tb_CompALU.in

```
1   001001_00001_00010_00000_00000_001001
2   001010_00001_00010_00000_00000_001010
3   010001_00001_00010_00000_00000_010001
4   100001_00001_00000_00000_01010_100001
5
6   001001_00011_01111_11111_11111_001001
7   001010_11000_11111_10101_10101_001010
8   010001_11110_11101_01010_01010_010001
9   100001_11011_11111_00100_11111_100001
```

這個是 Complete ALU 的 in 檔，上面的部分是一開始 template 所提供的測資。從上到下分別是 add, sub, AND, shift， 而在下面的部分，我也分別測試了 add, sub, and, shift。理論上在經過上面 ALU 的多重測資驗證後應該不會有什麼樣的問題，於是在最大的這個 complete ALU 下，我主要是在 dest reg 與 shamt 放一些不需要的數字，想試試看在不需要這些 reg 的情況下，會不會造成甚麼 BUG，而在經過我的測試以後，確實不會造成問題。



作業總結與心得

這學期很幸運有機會加簽到這堂課，原本我是沒有選到的，好在第一次上課有爭取到加簽的機會。而對於這次作業，在歷經整個寒假沒有碰到Verilog的狀況下，第一次寫。實在是有點生疏，好在這次作業開始之前，助教有花了一點時間帶我們複習一下Verilog，有稍微抓回一些感覺。除此之外，我也有藉由moodle上所提供的template稍微改變了一下關於我在撰寫verilog 宣告wire, reg方式的風格。原本我會宣告在一起，但後來在看到template那樣寫之後，好像也能漸漸了解到這樣子的好處。

而此次作業與之前數位系統設計所學到verilog最大的不同就是，以往我們都是講所有模組寫在同一個.v檔，但是這次是把每個模組分開成各個檔案。我在上學期的數位系統期末專題原本就想這樣子做，但是後來不知道如何將所有模組合在一起而作罷，這次學習到了這個技能，相信對於以後關於verilog的程式設計，也能把我的code整理得更加簡潔，加強我的READABILITY。

希望以後這堂課的不論是考試或是project，都可以順利度過!